# Password Cracking: The Effect of Bias on the Average Guesswork of Hash Functions

Yair Yona, and Suhas Diggavi, *Fellow, IEEE*



**Abstract**

In this work we analyze the average guesswork for the problem of hashed passwords cracking (i.e., finding a password that has the same hash value as the actual password). We focus on the following two cases: Averaging over all strategies of guessing passwords one by one for any hash function that has effective distribution (i.e., the fraction of mappings to any bin) which is i.i.d. Bernoulli$(p)$, and averaging over all hash functions whose effective distribution is i.i.d. Bernoulli$(p)$ for any strategy of guessing passwords one by one.

For the case where the hash function is adaptively modified based on the passwords of the users we first find the average guesswork across users when the number of bins is $2^m$ and the number of users equals $\lfloor 2^{H(s)\cdot m-1} \rfloor$, where $1/2 \leq s \leq 1$ and $m \gg 1$. It turns out that the average guesswork increases (as a function of $m$) at rate that is equal to $H(s) + D(s||p)$ when $(1-p) \leq s \leq 1$, and $2 \cdot H(p) + D(1-p||p) - H(s)$ when $1/2 \leq s \leq (1-p)$. We then show that the average guesswork of guessing a password that is mapped to any of the assigned bins (an offline attack) grows like $2^{D(s||p)\cdot m}$. We also analyze the effect of choosing *biased passwords* on the average guesswork and characterize the region in which the average guesswork is dominated by the guesswork of a password as well as the region in which the average guesswork is dominated by the above results. Moreover, we provide a *concentration* result that shows that the probability mass function of the guesswork is concentrated around its mean value.

We also analyze the more prevalent case in which hash functions *can not* be modified based on the passwords of the users (i.e., users are mapped to bins randomly). We derive a lower and an upper bounds for the average guesswork both under offline and online attacks and show that the rate at which it increases under offline attacks is upper bounded by $D(s||p)$, and lower bounded by $D(1-s||p)$ when $1-p \leq s \leq 1$ as well as $0$ for $1/2 \leq s \leq 1-p$, whereas under an online attack the rate is upper bounded by $H(s) + D(s||p)$ when $(1-p) \leq s \leq 1$, and $2 \cdot H(p) + D(1-p||p) - H(s)$ when $1/2 \leq s \leq (1-p)$, and lower bounded by $H(s) + D(1-s||p)$. In addition, we show that the most likely average guesswork when passwords are drawn uniformly increases at rate $H(p) - H(s)$ under an offline attack and at rate $H(p)$ when cracking the password of any user. These results give quantifiable bounds for the effect of bias as well as the number of users on the average guesswork of a hash function, and show that increasing the number of users has a far worse effect than bias in terms of the average guesswork.

Furthermore, we show that under online attacks the average guesswork is upper bounded by $H(s) + D(s||p)$ when $(1-p) \leq s \leq 1$, and $2 \cdot H(p) + D(1-p||p) - H(s)$ when $1/2 \leq s \leq (1-p)$, and lower bounded by $H(s) + D(1-s||p)$

For keyed hash functions (i.e., strongly universal sets of hash functions) we show that when the number of users is $\lfloor 2^{m-1} \rfloor$ and the hash function is adaptively modified based on the passwords of the users, the size of a uniform key required to achieve an average guesswork $2^{\alpha \cdot m}$, $\alpha > 1$, is $\alpha$ times larger than the size of a key that is drawn Bernoulli$(p_0)$ that achieves the same average guesswork, where $p_0$ satisfies the equality $1 + D(1/2||p_0) = \alpha$.

Finally, we present a "backdoor" procedure that enables to modify a hash function efficiently without compromising the average guesswork. This work relies on the observation that when the mappings (or the key) of a hash function are biased, and the passwords of the users are mapped to the least likely bins, the average guesswork increases significantly.

## I. Introduction

A password is a means of protecting information by allowing access only to an authorized user who knows it. A system that provides services to multiple users, usually stores their passwords on a server; the system accesses the passwords stored on the server in order to validate a password that is provided by a user. Protecting passwords that are stored on a server is extremely important since servers are likely to be attacked by hackers; once a server is successfully attacked, all user accounts are compromised when the passwords are not properly protected, [1] ,[2] (e.g., if the passwords are stored in plain-text, then once the server is hacked the attacker gets a hold of all passwords that are stored on it, and therefore can break into any account).

The most prevalent method of protecting passwords on a server is using one-way hash functions which can be computed easily but can not be inverted easily [3], [4], [5], [1], [2]. Essentially, cryptographic hash functions are the "workhorses of modern cryptography" [6] and enable, among other uses, to securely protect passwords against offline attacks. Hashing has many applications in various fields such as compression [7], search problems [8], as well as cryptography [9]. When it comes to message authentication codes (MAC) [10] it is meaningful to consider keyed hash functions. Keyed hash functions can be viewed as a set of hash functions, where the actual function which is used is determined by the value assigned to the key which is a secret [11]. A practical keyed hash functions that enable to securely use off-the-shelf computationally secured hash functions, is a keyed-hash message authentication code (HMAC) [10].

The two main scenarios of passwords cracking are online guessing attack and offline attack [2]. In online guessing attack an attacker tries to login under a certain user name by guessing passwords one by one until he finds a password that is hashed to the same value as the original password. Offline attack occurs when an attacker gets a hold of the hashed passwords stored on the server; for any hash value that is stored on the server, the attacker searches for a password that is hashed to it.

In this work we derive the average guesswork for passwords cracking in both online and offline attacks. Guesswork is the number of attempts required to guess a secret. The guesswork is a random variable whose probability mass function depends on the statistical profile according to which a secret is chosen, along with the strategy used for guessing. It was first introduced and analyzed by Massey [12] who lower bounded the average guesswork by the Shannon entropy. Arikan showed that the rate at which any moment of the guesswork increases is actually equal to the Renyi entropy [13], and is larger than the Shannon entropy unless the secret is uniformly distributed in which case both are equal. Guesswork has been analyzed in many other scenarios such as guessing up to a certain level of distortion [14], guessing under source uncertainty with and without side information [15], [16], using guesswork to lower bound the complexity of sequential decoding [13], guesswork for Markov chains [17], guesswork for multi-user systems [18] as well as guesswork for the Shannon cipher system [19].

A hash function is a mapping from a larger domain to a smaller range, and therefore in this work we consider a slightly different version of guesswork in which an attacker should guess *a password* that has the same hash value as the original password, and not necessarily *the password*. The problem that we consider, the tools that we use for analysis, and the theoretical results are all very different from the ones in [14] where guessing with distortion is considered, because in [14] a single-letter distortion measure is considered, whereas in our problem a single letter-distortion measure can not capture the event of success in passwords cracking for every hash function. Furthermore, in [19] the authors analyze the guesswork of the Shannon cipher system with a guessing wiretapper. In this case the attacker captures a cryptogram, which is a function over a key and a message; the attacker tries to guess the message, which is protected by the cipher system. The main difference between this setting and our problem lies in the fact that the cryptogram is invertible, whereas a keyed hash function is not invertible. This in turn requires different analytical approach and leads to different results. Moreover, in [20] the Shannon cipher system is analyzed for the case where the attacker produces a list of possible messages, and secrecy is measured by the minimum distortion over the entire list; however, in their work the authors consider a single-letter distortion measure, which does not capture our problem.

We derive the average guesswork for two scenarios: The average guesswork for any hash function given the fractions of mappings to every bin, and the average guesswork of a keyed hash function when the key is independent and identically distributed (i.i.d.) Bernoulli$(p)$, where $0 < p \leq 1/2$. In both cases we assume that the number of users is smaller than the number of bins, and we consider both the case when the hash function or the key can be adaptively modified according to the passwords of the users (i.e., a procedure decides which bin is allocated to which user, and modifies the hash function such that the password of the user is mapped to this bin), and the case when the hash function *can not* be adaptively modified according to the passwords. Note that the procedure of modifying a hash function is not as hard as cracking the passwords (and as a result breaking the hash function); in fact, we present in this paper a "backdoor" mechanism that enables to modify a hash function efficiently without decreasing the average guesswork, as long as the number of users is not significantly larger than the number of bins.

We show that when a hash function can be adaptively modified according to the passwords, the more "unbalanced" the hash function is, the larger is the average guesswork (i.e., for a balanced hash function the fraction of mappings to each bin is identical; the smaller the fraction of mappings to a certain subset of bins is, the more "unbalanced" the hash function becomes). The underlying idea behind this result is that when the fraction of mappings to a bin is small, it is much harder to find *a password* that is mapped to this bin (since there is a small number of mappings to this bin). Therefore, when users are coupled to the subset of bins which are least likely, the average guesswork increases compared to the case where a hash function is balanced. This result holds under the assumption that either the attacker does not know the actual hash function (he may know the fractions of mappings to each bin), or alternatively when considering keyed hash functions, that he does not know the key (but may know the probability mass function according to which the key is drawn).

For example, when the fraction of mappings to a certain bin equals $1/100$, the average guesswork of finding a password that is mapped to this bin grows like $100^m$, where $m \gg 1$ is the size of the output of the hash function (the size of the bins); where the average is over all strategies of guessing passwords one by one. When a user is assigned to this bin, the average guesswork required to crack his password also grows like $100^m$. This term is significantly larger than the average guesswork of any user when the hash function is balanced in which case the average guesswork grows like $2^m$.

Another motivation for adapting a hash function according to the passwords, is in case a service (e.g., a bank) that has multiple user accounts (e.g., checking accounts), stores hashed passwords on an online server. However, this service provides stronger protection to some of its users (e.g., customers who have very large bank accounts). A regular user chooses his own password which is hashed; whereas a "premium" user receives a security token that contains a password (that may change over time), such that this password is mapped to one of the least likely bins by some "backdoor" procedure.

Furthermore, for the case when hash functions *can not* be adaptively modified, we provide expressions that enable to quantify the effect of bias and the number of users on the average guesswork. It turns out that increasing the number of users has a far worse effect than bias in terms of the average guesswork.

The first main result in this paper is the derivation of the average guesswork when either averaging over all strategies of guessing passwords one by one for any hash function that has effective distribution (i.e., the fraction of mappings to any bin, which is the number of inputs that are mapped to a bin divided by the total number of inputs) which is i.i.d. Bernoulli$(p)$, or averaging over all hash functions whose effective distribution is i.i.d. Bernoulli$(p)$ for any strategy of guessing passwords one by one; and assuming that the passwords of the users are mapped to the least likely bins, that is, the bins that have the smallest fractions of mappings. While the attacker may know the fractions of mappings (i.e., the statistical profile of the mappings), we assume that he does not know the actual mappings. Based on this assumption we show that when passwords are drawn uniformly, the number of users is $\lfloor 2^{m \cdot H(s)-1} \rfloor$ where $1/2 \leq s \leq 1$, and $m \gg 1$ is the size of the output of the hash function, the average guesswork increases at rate (as a function of $m$) that is equal to $H(s) + D(s||p)$ when $(1-p) \leq s \leq 1$, and $2 \cdot H(p) + D(1-p||p) - H(s)$ when $1/2 \leq s \leq (1-p)$, where $D(\cdot||\cdot)$ is the Kullback-Leibler divergence, and $H(\cdot)$ is the binary Shannon entropy [7]. Furthermore, we also analyze the effect of choosing *biased passwords* on the average guesswork and characterize the region in which the average guesswork is dominated by the guesswork of a biased password as well as the region in which the average guesswork is dominated by the above results. Furthermore, when considering an offline attack in which the attacker obtains the hash values that are stored on the server, we show that the average guesswork of the event where the attacker finds *a password* that is mapped to any of the hash values stored on the server, grows like $2^{D(s||p) \cdot m}$.

Note that as $p$ decreases (i.e., the hash function is more biased), the average guesswork increases at a higher rate; in fact this rate is an unbounded monotonically increasing function of $(1-p)$. However, the minimal size of the input required to achieve this average guesswork is equal to $\log(1/p) \cdot m$. Hence, as

$p$ decreases, the size of the input required to achieve the average guesswork above also increases.

The second main result in this paper deals with the more prevalent case in which a hash function *can not* be adaptively modified according to the passwords of the users. We derive lower and upper bounds for the average guesswork for this case under both offline and online attacks. We show that under offline attacks the rate at which the average guesswork increases is upper bounded by $D\left(s||p\right)$ (the rate of the average guesswork when hash functions are adaptively modified), and lower bounded by $D\left(1-s||p\right)$ when $1-p \leq s \leq 1$ as well as $0$ for $1/2 \leq s \leq 1-p$. In addition to these bounds we also find the most likely average guesswork when passwords are drawn uniformly and show that it increases at rate $H\left(p\right)-H\left(s\right)$. Furthermore, we show that under online attacks the average guesswork is upper bounded by $H\left(s\right)+D\left(s||p\right)$ when $(1-p) \leq s \leq 1$, and $2 \cdot H\left(p\right)+D\left(1-p||p\right)-H\left(s\right)$ when $1/2 \leq s \leq (1-p)$, and lower bounded by $H\left(s\right)+D\left(1-s||p\right)$, whereas the most likely average guesswork increases at rate $H\left(p\right)$. These results give quantifiable bounds for the effect of bias as well as the number of users on the average guesswork of a hash function. Interestingly enough, it turns out that the effect of bias on the average guesswork is far less significant than the effect of increasing the number of users.

Furthermore, we provide a *concentration* result that holds either when passwords can or can not be adaptively modified. In the case where hash functions can be modified adaptively we show that the probability of drawing a strategy for which the average number of guesses is smaller than $2^{(H(s)+D(s||p))\cdot m}$ for any hash function whose effective distribution is i.i.d. Bernoulli$(p)$, vanishes exponentially as a function of $m$; this result also holds in the case when considering the fraction of hash functions for which the average guesswork is smaller than $2^{(H(s)+D(s||p))\cdot m}$ for any strategy of guessing passwords. Note that $H\left(s\right)+D\left(s||p\right)$ is smaller than $2 \cdot H\left(p\right)+D\left(1-p||p\right)-H\left(s\right)$ in the range $1/2 \leq s \leq (1-p)$. A similar result holds for the case where hash functions can not be modified, as shown in the paper.

The third main result in this paper is related to strongly universal sets of hash functions [11]. In this case we consider the interplay between three elements that determine the performance of a keyed hash function: The average guesswork, the size of the key, and the number of users. When the number of users is $\lfloor 2^{m-1} \rfloor$, and the average guesswork is $2^{\alpha \cdot m}$ where $\alpha > 1$, the size of a uniform key has to be $\alpha$ times larger than the size of a key that is drawn Bernoulli$(p_0)$ such that $1+D\left(1/2||p_0\right)=\alpha$.

The result above shows that the size of a biased key required to achieve an average guesswork of $2^{\alpha \cdot m}$ is $\alpha$ times smaller than the size of a uniform key that achieves the same average guesswork for the same number of users. This surprising result stems from the fact that the users are "beamformed" to the least likely bins (i.e., the passwords of the users, are mapped to the least likely bins), and therefore the system can utilize the bias of the key in order to achieve the same average guesswork with a shorter biased key.

The paper is organized as follows. We begin in Section II with background and basic definitions. The problem setting is presented in Section III, followed by the main results in Section IV. In Section V we derive the average guesswork for keyed hash functions, and in Section VI we show that a shorter biased key can achieve the same average guesswork as a longer uniform key. In Section VII we analyze the average guesswork for any hash function when the hash functions are adaptively modified, and finally in Section VIII we bound the average guesswork for the case where the hash functions can not be modified.

## II. Background and Basic Definitions

In this section we present basic definitions and results in the literature that we use throughout the paper.

### A. Guesswork

Consider the following game: Bob draws a sample $x$ from a random variable $X$, and an attacker Alice who does not know $x$ but knows the probability mass function $P_X\left(\cdot\right)$, tries to guess it. An oracle tells Alice whether her guess is right or wrong.

The number of guesses it takes Alice to guess $x$ successfully is a random variable $G\left(X\right)$ (which is termed guesswork) that takes only positive integer values. The optimal strategy of guessing $X$ that minimizes all non negative moments of $G(X)$

$$E\left(G(X)^{\rho}\right)=\sum_{x\in X}G(x)^{\rho}\cdot P_X(x)\quad \rho\geq 0 \tag{1}$$

is guessing elements in $X$ based on their probabilities in descending order [12], [13], such that $G(x) < G(x')$ implies $p_X(x) > p_X(x')$, i.e. a dictionary attack [21]; where $G(x)$ is the number of guesses after which the attacker guesses $x$. In this paper we analyze the optimal guesswork and therefore with slight abuse of notations we define $G(X)$ to be the *optimal guesswork*.

It has been shown that $E\left(G(X)^{\rho}\right)$ is dictated by the Renyi entropy [13]

$$H_{\frac{1}{1+\rho}}\left(P_X(x)\right)=\frac{1}{\rho}\log_2\left(\left(\sum_x P(x)^{\frac{1}{1+\rho}}\right)^{1+\rho}\right). \tag{2}$$

For example, when drawing a random vector $\underline{X}$ of length $k$, which is independent and identically distributed (i.i.d.) with distribution $P=[p_1,\ldots,p_M]$, the exponential growth rate of the average guesswork scales according to the Renyi entropy $H_{\alpha}(X)$ with parameter $\alpha=1/2$ [13]:

$$\lim_{k\to\infty}\frac{1}{k}\log_2\left(E\left(G(X)\right)\right)=H_{1/2}(P)=2\cdot\log_2\left(\sum_i p_i^{1/2}\right) \tag{3}$$

where $H_{\frac{1}{2}}(P)\geq H(P)=-\sum_{x\in X}p(x)\log\left(p(x)\right)$ which is the Shannon entropy, with equality only for the uniform probability mass function. Furthermore, for any $\rho\geq 0$

$$\lim_{k\to\infty}\frac{1}{k}\log_2\left(E\left(G(X)^{\rho}\right)\right)=\rho\cdot H_{\frac{1}{1+\rho}}(P). \tag{4}$$

The definition of guesswork was also extended to the case where the attacker has a side information $Y$ available [13]. In this case the average guesswork for $Y=y$ is defined as $G(X|Y=y)$, and the $\rho$th moment of $G(X|Y)$ is

$$E\left(G(X|Y)^{\rho}\right)=\sum_y E\left(G(X|Y=y)^{\rho}\right)\cdot p_Y(y). \tag{5}$$

Arikan [13] has bounded the $\rho$th moment of the optimal guesswork, $G(X|Y)$, by

$$\begin{aligned}(1+\ln(M))^{-\rho}\sum_y\left(\sum_x P_{X,Y}(x,y)^{\frac{1}{1+\rho}}\right)^{1+\rho}&\leq\\ E\left(G(X|Y)^{\rho}\right)\leq\sum_y\left(\sum_x P_{X,Y}(x,y)^{\frac{1}{1+\rho}}\right)^{1+\rho}&\end{aligned} \tag{6}$$

where $M=|X|$ is the cardinality of $X$. Furthermore, in [13] it has been shown that when $X$ and $Y$ are identically and independently distributed (i.i.d.), the exponential growth rate of the optimal guesswork is

$$\lim_{k\to\infty}\frac{1}{k}\log_2\left(E\left(G^*(X|Y)^{\rho}\right)\right)=\rho\cdot H_{\frac{1}{1+\rho}}\left(P_{X,Y}(x,y)\right) \tag{7}$$

where $m$ is the size of $X$ and $Y$, and

$$H_{\frac{1}{1+\rho}}\left(P_{X,Y}(x,y)\right)=\frac{1}{\rho}\log_2\left(\sum_y\left(\sum_x P(x,y)^{\frac{1}{1+\rho}}\right)^{1+\rho}\right) \tag{8}$$

is Renyi's conditional entropy of order $\frac{1}{1+\rho}$ [13].

### B. Hash Functions and Strongly Universal Sets of Hash Functions

A hash function, $H(\cdot)$, is a mapping from a larger domain $A$ to a smaller range $B$. In this work we assume that the input is of size $n$ bits whereas the output is $m$ bits long, where $n > m$. We term the output of a hash function *bin* which is denoted by $b$. For the analysis of the average guesswork for any hash function, we define the fractions of mappings to any bin; we use the following definition throughout the paper.

**Definition 1** (Fractions of mappings of a hash function)**.** *Consider a hash function with an input of size $n$ and an output of size $m$, where $n > m$. The pattern of the hash function is defined as follows*

$$P_H(b) = \frac{1}{2^n}\sum_{i=1}^{2^n} \mathbb{1}_b(i) \quad b \in \{1, \ldots, 2^m\} \tag{9}$$

*where $\mathbb{1}_b(x) = \begin{cases} 1 & x = b \\ 0 & x \neq b \end{cases}$. Thus, $P_H(b)$ is the ratio between the number of mappings to $b$, and the total number of mappings.*

**Definition 2.** *A $P_H(b)$*-hash function *is a hash function whose fractions of mappings are equal to $P_H(b)$, $b \in \{1, \ldots, 2^m\}$.*

A Keyed hash function is a set of hash functions, where the actual hash function being used is determined by the value assigned to the key. An idealized version of keyed hash functions is *strongly universal set of hash functions* [11] which is essentially the set of all hash functions with an input of size $n$ bits and an output of size $m$ bits. Therefore, it can be defined as the set $\left\{H_{f(\underline{k})}(\cdot)\right\}$, where $\underline{k}$ is a key of size $m \cdot 2^n$, $f(\underline{k})$ is a bijection, and $\{H_i(\cdot)\}$, $i \in \{1, \ldots, m \cdot 2^n\}$, is the set of all possible mappings. [1]

In this work, when considering biased keys we assume the mapping $g(\underline{k})$ which is defined as follows. First, we break the key into $2^n$ segments of size $m$ such that

$$\underline{k} = \{k_1, \ldots, k_{2^n}\}. \tag{10}$$

Then, we state that

$$H_{g(\underline{k})}(i) = k_i \quad i \in \{1, \ldots, 2^n\}. \tag{11}$$

Figure 1 illustrates the definition presented above for $g(\underline{k})$.

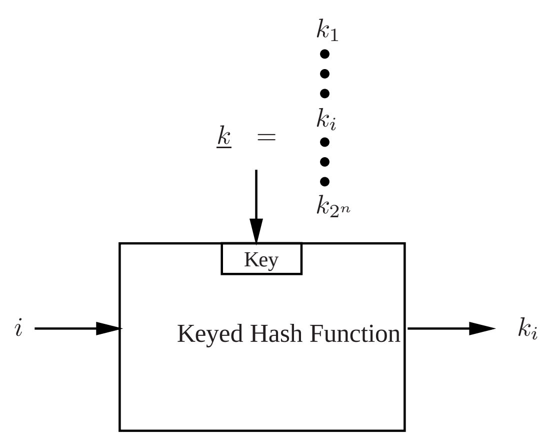


Figure 1. The definition of the keyed hash function that we analyze in this paper. When the input equals $i$, the output is equal to the $i$th segment of the key, that is, $k_i$. Essentially, this is a strongly universal set of hash functions.

[1]Note that a strongly universal set of hash functions is also used for compression and termed random binning [7].

**Remark 1.** *Note that the mapping defined in equations* (10), (11) *from the set of keys to the set of all possible hash functions, is a strongly universal set of hash functions. Any other mapping that is a bijection from the set of keys to the set of all possible hash function also constitutes a strongly universal set of hash functions. Essentially, in section VI we show that this mapping along with a biased key achieves better performance than an unbiased key with any mapping that is a bijection (see Corollary 8 for the average guesswork of an unbiased key and any mapping that is a bijection). A biased key with other mappings may also lead to better performance than the one achieved with an unbiased key; however, we have been able to quantify the average guesswork for the mapping defined in equations* (10), (11)*.*

We now define another set of keyed hash functions which we use in Subsection IV-A, that relies on Definition 1.

**Definition 3.** *The* $P_H\left(b\right)$-set of hash functions *is all hash functions for which the fractions of mappings equal* $P_H\left(b\right)$ *as given in Definition 1; that is, these hash functions are identical up to a permutation.*

## C. Method of Types

In this paper we analyze the guesswork of biased keys based on the method of types [7]. Assume a binary vector $\underline{X}$ of size $m$ which is drawn i.i.d. Bernoulli$(p)$. The realization $\underline{x}$ is of type $q$ in case $N\left(1|\underline{x}\right)/m = q$ where $N\left(1|\underline{x}\right)$ is the number of occurrences of the number 1 in the binary vector $\underline{x}$.

The first result is related to the probability of drawing a type $q$ when the underlying probability is Bernoulli$(p)$.

**Lemma 1** ([7] Theorem 11.1.2)**.** *The probability* $Q\left(q\right) = P\left(N\left(1|\underline{x}\right)/m = q\right)$ *which is the probability that* $\underline{x}$ *is of type* $q$*, is equal to*

$$Q\left(q\right) = 2^{-m\left(H(q)+D(q||p)\right)} \tag{12}$$

*where* $D\left(q||p\right) = \sum_i q_i \log\left(q_i/p_i\right)$ *is the Kullback-Leibler divergence [7].*

The next lemma bounds the number of elements for each type.

**Lemma 2** ([7] Theorem 11.1.3)**.** *The number of elements for which* $N\left(1|\underline{x}\right)/m = q$ *is bounded by the following terms*

$$\frac{1}{\left(m+1\right)^2} \cdot 2^{m \cdot H(q)} \leq \left|N\left(1|\underline{x}\right) = q\right| \leq 2^{m \cdot H(q)}. \tag{13}$$

# III. Problem Setting

In this section we define the problem and the attack model, we extend the definition of average guesswork to the case of cracking passwords, and define the process of modifying a hash function.

The following definitions address the way passwords are stored in the system, and how access to the system is granted.

**Definition 4.** *We define the method according to which the system stores passwords.*

- *There are* $M$ *users registered in the system.*
- *In order to access the system a user provides his user name and password.*
- *The system does not store the passwords, but rather stores the user names and the hashed values of the passwords.*
- *The system hashes the password with either a hash function or the strongly universal set of hash functions (a keyed hash function) defined in subsection II-B equations* (10), (11)*.*

**Definition 5.** *The following protocol grants a user access.*

- *The user sends its user name to the system.*
- *The system pulls up the bin value which is coupled with this user name.*

- *The user types a password; the system hashes the password; if the hashed value matches the bin from the previous bullet, then access is granted.*

Next, we define the attack models.

**Definition 6** (An online Attack)**.** *We begin by defining the attack model when the system uses* any hash function.

- *The attacker does not know what the hash function is; he may know the fractions of mappings given in Definition 1, but he does not know the actual mappings.*
- *The attacker does not know the passwords of the users; he also does not know the bins that are stored on the system.*
- *The attacker can guess the passwords one by one; he can choose any strategy for guessing the passwords.*
- *The attacker chooses a user name and then guesses passwords one by one. If the hash value to which the password is mapped does not match the bin, then he is not allowed to access the system; once the attacker guesses a password that is mapped to the bin coupled with the user name, he can access the system.*

*In the case when the system uses the universal set of hash functions defined in subsection II-B equations* (10), (11), *we make two additional assumptions: The key is drawn i.i.d. Bernoulli*$(p)$*; and the attacker knows the distribution according to which the key is drawn, but he does not know the actual realization of the key.*

Figure 2 presents the setting of the problem presented above.

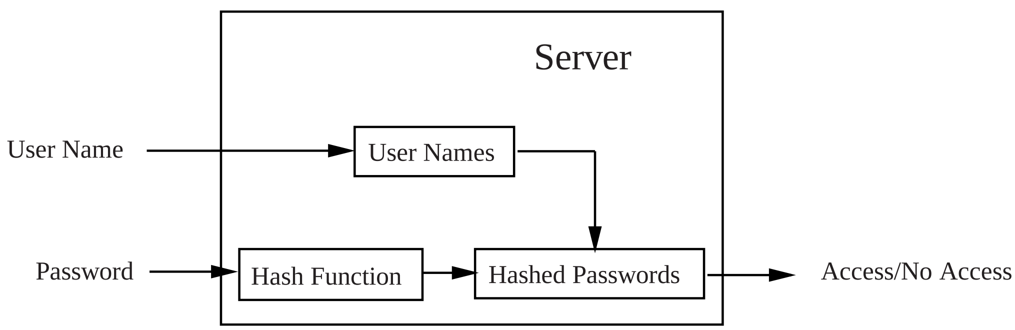


Figure 2. The user/attacker enters a user name and a password. The server hashes the password with a hash function, and compares its output to the value that is stored on the server for the user name that was entered. When the values match access is granted; otherwise access is denied.

**Definition 7** (An offline attack)**.** *Assume that the attacker* knows the bins *that are stored on the system; yet he does not know the passwords, and the hash function that the system uses (or the key when the system uses a keyed hash function). Other than that, the attack model is identical to the one in Definition 6.*

We now define the average guesswork for passwords cracking.

**Definition 8** (The guesswork for cracking a certain bin)**.** *The guesswork for cracking a bin $b$, denoted by $G(b)$, is the number of guesses required to find* a password *which is mapped to bin $b$ (i.e., not necessarily* the password *of the user).*

**Definition 9.** *Assume that the number of users $M \leq 2^m$. The attacker draws a user out of the $M$ users, according to a uniform probability mass function. The average guesswork across all users is*

$$E\left(G_T\left(B\right)\right) = \frac{1}{M}\sum_{b\in B} E\left(G\left(b\right)\right) \tag{14}$$

*where $G(\cdot)$ is the guesswork as a function of the bin as given in Definition 8, and $B$ is the set of $M$ bins to which the passwords of the $M$ users are mapped (i.e., $|B| = M$).*

**Definition 10** (Averaging Arguments)**.** *The guesswork of any $P_H(b)$-hash function is averaged over all possible strategies of guessing passwords one by one, whereas the guesswork of the $P_H(b)$-set of hash functions is averaged over all elements in the set for any strategy of guessing passwords. In both cases we assume that the average is performed over a uniform distribution. On the other hand, in this work the guesswork of a universal set of hash functions is averaged over its key which may be biased (e.g., a key that is i.i.d. Bernoulli$(p)$).*

We now define a method for allocating bins when bins allocation is in order.

**Definition 11** (bin allocation)**.** *Assume that the number of users $M \leq 2^m$, the bin allocation method is defined as follows. For each user a procedure allocates a different bin $b \in \{1, \ldots, 2^m\}$ according to the following method: The first user receives the least likely bin (i.e., when considering any hash function, this is the bin that has the smallest fraction of mappings; when considering a universal set of hash functions, this is the all ones bin), the second user receives the second least likely bin, whereas the last user receives the $M$th least likely bin.*

Three remarks regarding bins allocation are in order.

**Remark 2.** *In Definition 15 and Definition 13 we present a "backdoor" mechanism that efficiently implements the bins allocation procedure presented in Definition 11, without decreasing the average guesswork.*

**Remark 3.** *In Definition 11 we state that when two users have different passwords, they are mapped to different bins. In general, this is required in order for the backdoor mechanism discussed in remark 2 not to decrease the average guesswork.*

**Remark 4.** *mapping passwords to the least likely bins, either based on the fractions of mappings or on the distribution of the key when considering a keyed hash function, is essential for using bias to increase the average guesswork.*

## IV. Main Results

In this section we present the main results on the average guesswork both for any hash function and keyed hash functions.

### A. The First Main Result: The Average Guesswork of any $P_H(b)$-hash function as well as the $P_H(b)$-set of hash functions when hash functions can be adaptively modified

In this subsection we derive the average guesswork for any $P_H(b)$-hash function as well as for the $P_H(b)$-set of hash functions, when bins are allocated to the users. We begin by finding the average guesswork across users under an online attack when the passwords are drawn uniformly, followed by a concentration result. We then present the effect of choosing passwords non-uniformly on the average guesswork. Finally, we consider an offline attack in which the attacker tries to find a password that is mapped to any of the assigned bins.

First, we derive the rate at which the average guesswork increases, when the fractions of mappings are i.i.d. Bernoulli$(p)$.

**Theorem 1** (An online attack)**.** *Under the following assumptions.*

- *The attacker does not know the mappings of the hash function.*
- *Bins are allocated as in Definition 15 to $M = 2^{H(s)\cdot m-1}$ users, where $1/2 \leq s \leq 1$.*
- *The fractions of mappings given in Definition 1, $P_H(b)$, represent drawing $m$ bits i.i.d. Bernoulli$(p)$, $p \leq 1/2$.*

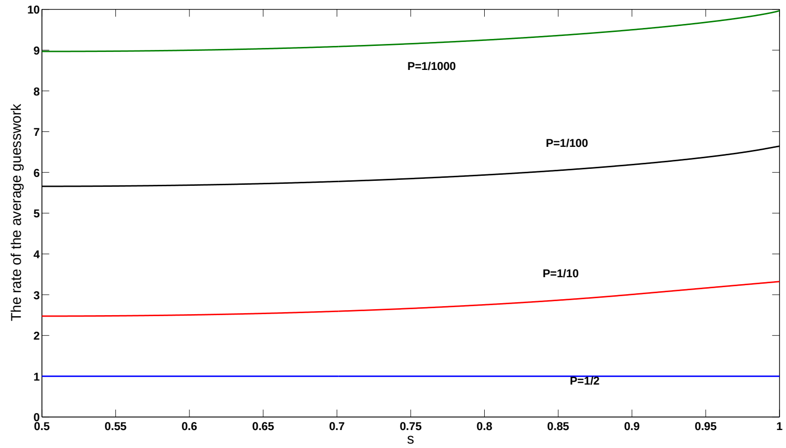


Figure 3. The rate at which the average guesswork increases (i.e., $\lim_{m\to\infty}\frac{1}{m}\log\left(E\left(G_T\left(B\right)\right)\right)$ as a function of $s\in[1/2,1]$; the number of users is $2^{H(s)\cdot m-1}$ and so the larger $s$ is, the smaller the number of users is. Hence, $s=1/2$ represents the largest number of users, whereas $s=1$ represents the smallest number. For $1/2\leq s\leq(1-p)$ the average guesswork is equal to $2\cdot H\left(p\right)+D\left(1-p||p\right)-H\left(s\right)$, and for $(1-p)\leq s\leq 1$ the average guesswork equals $H\left(s\right)+D\left(s||p\right)$. Note that for $p=1/2$ the average guesswork increases at rate that equals $1$ regardless of the number of users, whereas as $p$ decreases the rate of the average guesswork increases.

- $n\geq(1+\epsilon)\cdot m\cdot\left(\log\left(1/p\right)+H\left(s\right)\right)$ *where* $\epsilon>0$.

*The average guesswork of any* $P_H\left(b\right)$*-hash function as well as the* $P_H\left(b\right)$*-set of hash functions, when averaged in accordance with Definition 10 is equal to*

$$\lim_{m\to\infty}\frac{1}{m}\log\left(E\left(G_T\left(B\right)\right)\right)=\begin{cases}H\left(s\right)+D\left(s||p\right) & (1-p)\leq s\leq 1\\ 2\cdot H\left(p\right)+D\left(1-p||p\right)-H\left(s\right) & 1/2\leq s\leq(1-p)\end{cases}. \tag{15}$$

*Proof:* The full proof appears in Section VII. ■

Figure 3 illustrates the above result.

**Theorem 2** (An online attack)**.** *Under the following assumptions.*

- *The attacker does not know the mappings of the hash function.*
- *There are* $M=2^{H(s)\cdot m-1}$ *users,* $1/2\leq s\leq 1$*, to which bins are allocated according to Definition 15.*
- $P_H\left(b\right)$ *represents drawing* $m$ *bits i.i.d. Bernoulli*$(p)$*,* $p\leq 1/2$*.*
- $n\geq(1+\epsilon)\cdot m\cdot\left(\log\left(1/p\right)+H\left(s\right)\right)$ *where* $\epsilon>0$.

*The following concentration result holds for all users when considering either any* $P_H\left(b\right)$*-hash function or the* $P_H\left(b\right)$*-set of hash functions*

$$-\lim_{m\to\infty}\frac{1}{m}\log\left(P\left(G\left(b\right)\leq 2^{(1-\epsilon_1)\cdot(H(s)+D(s||p))\cdot m}\right)\right)\geq\epsilon_1\cdot\log\left(1/p\right)\quad\forall b\in B \tag{16}$$

*where* $0<\epsilon_1<1$.

*Proof:* The proof is in Section VII. ■

Figure 4 illustrates the intuition behind the above results.

**Remark 5.** *From Theorem 2 we can state that for any user and any* $P_H\left(b\right)$*-hash function, the fraction of strategies of guessing passwords one by one, for which the number of guesses is smaller than* $2^{(1-\epsilon_1)\cdot(H(s)+D(s||p))\cdot m}$*, decreases like* $2^{-\epsilon_1\cdot\log(1/p)\cdot m}$*. Also note that* $H\left(s\right)+D\left(s||p\right)\leq 2\cdot H\left(p\right)+D\left(1-p||p\right)-$

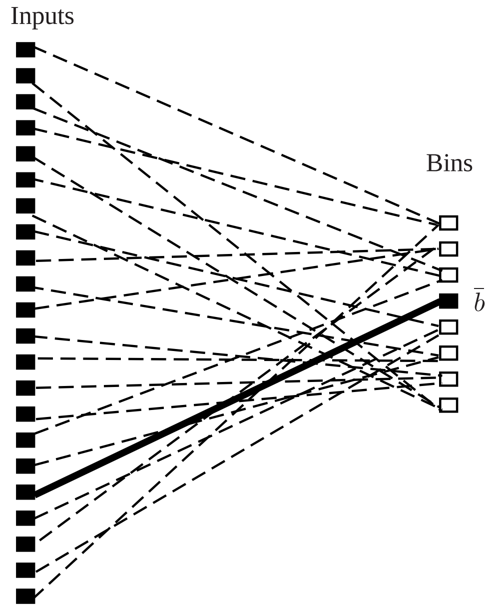


Figure 4. Due to unbalance, there is only one mapping from the set of inputs to the solid black bin $\bar{b}$ (the solid black line). Hence, the average number of guesses required to find a password that is mapped to it, is larger than the average number of guesses of the other bins, that is, the average guesswork of $\bar{b}$ is larger than the average guesswork of the others.

*$H(s)$ in the range $1/2 \leq s \leq 1-p$. Furthermore, for the $P_H(b)$-set of hash functions the fraction of $P_H(b)$-hash functions for which the number of guesses is smaller than $2^{(1-\epsilon_1)\cdot(H(s)+D(s||p))\cdot m}$ also decreases like $2^{-\epsilon_1 \cdot \log(1/p)\cdot m}$ for any user when considering any strategy of guessing passwords one by one.*

The next corollary presents the effect of a biased password on the average guesswork. We consider the case where passwords are drawn i.i.d. Bernoulli($q$). Based on the concentration result of Theorem 2, we characterize a region in which the guesswork of Theorem 1 dominates the average guesswork, as well as another region in which the average guesswork of a password (3) is the dominant one. Since in this case the optimal strategy is guessing passwords based on their probabilities in descending order [13], the optimal average guesswork should be averaged over the $P_H(b)$-set of hash functions for this strategy (i.e., not over all possible strategies of guessing passwords).

**Corollary 1** (Biased passwords)**.** *Under the following assumptions.*

- *The attacker does not know the mappings of the hash function.*
- *Bins are allocated as in Definition 15 to $M = 2^{H(s)\cdot m-1}$ users, where $1/2 \leq s \leq 1$; however, unlike Definition 15, here the passwords are drawn i.i.d. Bernoulli($\theta$), where $0 < \theta < 1/2$.*
- *$P_H(b)$ represents drawing $m$ bits i.i.d. Bernoulli($p$), $p \leq 1/2$.*
- *$n \geq (1+\epsilon)\cdot m \cdot (\log(1/p) + H(s))$ where $\epsilon > 0$.*

*When passwords are guessed based on their probabilities in descending order, the average guesswork of a user who is mapped to bin $b \in B$, averaged over the $P_H(b)$-set of hash functions in accordance with Definition 10, , is equal to*

$$\lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G\left(b\right)\right)\right) =$$

$$\lim_{m\to\infty} \begin{cases} \frac{n}{m} H_{1/2}(\theta) & 2\cdot\frac{n}{m}\cdot H\left(\frac{\sqrt{\theta}}{\sqrt{\theta}+\sqrt{1-\theta}}\right) < H\left(q\left(b\right)\right) + D\left(q\left(b\right)||p\right) \\ H\left(q\left(b\right)\right) + D\left(q\left(b\right)||p\right) & \frac{n}{m} H(\theta) > H\left(q\left(b\right)\right) + D\left(q\left(b\right)||p\right) \end{cases} \quad (17)$$

*Furthermore, the average guesswork across users is*

$$\lim_{m\to\infty}\frac{1}{m}\log\left(E\left(G_T\left(B\right)\right)\right)=\lim_{m\to\infty}\frac{n}{m}H_{1/2}\left(\theta\right) \quad \frac{2n}{m}H\left(\frac{\sqrt{\theta}}{\sqrt{\theta}+\sqrt{1-\theta}}\right)<H\left(s\right)+D\left(s||p\right) \tag{18}$$

*and*

$$\lim_{m\to\infty}\frac{1}{m}\log\left(E\left(G_T\left(B\right)\right)\right)=\begin{cases} H\left(s\right)+D\left(s||p\right) & \left(1-p\right)\leq s\leq 1\\ 2\cdot H\left(p\right)+D\left(1-p||p\right)-H\left(s\right) & 1/2\leq s\leq\left(1-p\right)\end{cases} \tag{19}$$

*when* $\lim_{m\to\infty}\frac{n}{m}H\left(\theta\right)>\log\left(1/p\right)$.

*Proof:* The full proof appears in Section VII. ■

We now give an illustrative example for the result above.

**Example 1.** *Consider the* $P_H\left(b\right)$*-set of hash functions, where* $P_H\left(b\right)$ *represents drawing* $m$ *bits i.i.d., and* $n=\left(1+\epsilon\right)\cdot\log\left(1/p\right)\cdot m$*. In addition, assume that every user out of the* $2^{H(s)\cdot m-1}$ *users draws his password i.i.d. Bernoulli*$(\theta)$*. In this case, when* $2\left(1+\epsilon\right)\cdot\log\left(1/p\right)\cdot H\left(\frac{\sqrt{\theta}}{\sqrt{\theta}+\sqrt{1-\theta}}\right)<H\left(s\right)+D\left(s||p\right)$ *the rate at which the average guesswork across users increases equals* $\left(1+\epsilon\right)\cdot\log\left(1/p\right)\cdot H_{1/2}\left(\theta\right)$*, whereas when* $\left(1+\epsilon\right)\cdot\log\left(1/p\right)\cdot H\left(\theta\right)>\log\left(1/p\right)$ *the rate is equal to* $\begin{cases} H\left(s\right)+D\left(s||p\right) & \left(1-p\right)\leq s\leq 1\\ 2\cdot H\left(p\right)+D\left(1-p||p\right)-H\left(s\right) & 1/2\leq s\leq\left(1-p\right)\end{cases}$.

**Remark 6.** *An offline attack is one in which the attacker knows which bin is mapped to which user. In this case the following question is meaningful: What is the average number of guesses required in order for the attacker to find a password that is mapped to any of the assigned bins. The next corollary answers this question.*

**Corollary 2** (An offline attack)**.** *When the following assumptions hold.*

- *The attacker does not know the mappings of the hash function.*
- *There are* $M=2^{H(s)\cdot m-1}$ *users,* $1/2\leq s\leq 1$*, to which bins are allocated according to Definition 15.*
- $P_H\left(b\right)$ *represents drawing* $m$ *bits i.i.d. Bernoulli*$(p)$*,* $p\leq 1/2$*.*
- $n\geq\left(1+\epsilon\right)\cdot m\cdot\left(\log\left(1/p\right)+H\left(s\right)\right)$ *where* $\epsilon>0$*.*
- *The attacker knows which bin is assigned to which user.*

*the average number of guesses required to find a password that is mapped to any of the bins assigned to users, is equal to*

$$\lim_{m\to\infty}\frac{1}{m}\log\left(E\left(G_{Any}\left(B\right)\right)\right)=D\left(s||p\right). \tag{20}$$

*Proof:* The full proof appears in Section VII. ■

**Remark 7.** *We assume that the number of users* $M=2^{H(s)\cdot m-1}$ *since based on Definition 11, it assures that the type of each bin* $b\in B$ *satisfies* $s\leq q\left(b\right)\leq 1$*, where* $q\left(b\right)$ *is the type of bin* $b$*. See Remark 20 for more details.*

**Remark 8.** *Note that the rate at which the average guesswork increases*

$$\begin{cases} H\left(s\right)+D\left(s||p\right) & \left(1-p\right)\leq s\leq 1\\ 2\cdot H\left(p\right)+D\left(1-p||p\right)-H\left(s\right) & 1/2\leq s\leq\left(1-p\right)\end{cases} \tag{21}$$

*is an unbounded function that increases as* $p$ *decreases. However, as* $p$ *decreases, the minimal size of the input* $n=\log\left(1/p\right)\cdot m$ *increases. Hence, the smaller* $p$ *is, the more asymptotic the result on the average guesswork becomes, in terms of the size of the input required to achieve this average.*

### *B. The Second Main Result: Bounds on the Average Guesswork of any $P_H(b)$-hash function as well as the $P_H(b)$-set of hash functions when hash functions* can not *be adaptively modified*

In this subsection as opposed to subsection IV-A, we derive the average guesswork for the case where users are mapped to bins randomly and *there is no* procedure that modifies mappings from password to the list likely bins (i.e., the hash function can not be modified).

We assume that each password consists of $n$ bits that are drawn i.i.d. Bernoulli$(1/2)$ and analyze the average guesswork of both any $P_H(b)$-hash function averaged over all possible strategies of guessing passwords one by one, and the $P_H(b)$-set of hash functions when averaged over all elements in the set and any strategy of guessing passwords one by one. We consider both online and offline attacks, and derive bounds as well as find the most likely average guesswork, that is, when $2^{H(s)\cdot m}$ users draw passwords we find the most probable average guesswork. In addition, we present a concentration result.

In order to derive the bounds we assume that the bins to which passwords are mapped are either the most likely bins or the least likely ones. This in turn enables us to come up with a lower and an upper bounds for the average guesswork, respectively. On the other hand, we find the most probable average guesswork by characterizing the most likely set of bins that have the same average guesswork.

The results of this subsection provide quantifiable bounds for the effect of bias as well as the effect of the number of users, on the average guesswork of a hash function. Furthermore, these results show that increasing the umber of users has a far worse effect on the average guesswork than bias.

We begin by considering the case when the attacker needs to guess a password that is mapped to any of the assigned bins.

**Theorem 3** (Bounds on the average guesswork under offline attacks)**.** *When the following assumptions hold.*

- *The attacker does not know the mappings of the hash function.*
- *There are $2^{H(s)\cdot m-1}$ bins to which the passwords are mapped, $1/2 \le s \le 1$.*
- *$P_H(b)$ represents drawing $m$ bits i.i.d. Bernoulli$(p)$, $p \le 1/2$.*
- *$n \ge (1+\epsilon)\cdot m \cdot \log(1/p)$ where $\epsilon > 0$.*
- *The attacker knows which bin is mapped to which user.*

*The rate of the average number of guesses required to find a password that is mapped to any of the bins to which the passwords are mapped, is bounded by*

$$\begin{cases} D(1-s||p) & 0 \le 1-s \le p \\ 0 & p \le 1-s \le 1/2 \end{cases} \le \lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G_{Any}(B)\right)\right) \le D(s||p), \tag{22}$$

*where the average for any $P_H(b)$-hash function as well as for the $P_H(b)$-set of hash functions is done in accordance with Definition 10.*

*Proof:* The full proof appears in Section VIII. ∎

We now find the most likely average guesswork.

**Corollary 3** (The most likely average guesswork under offline attacks)**.** *When the following assumptions hold.*

- *The attacker does not know the mappings of the hash function.*
- *There are $2^{H(s)\cdot m}$ users, $1/2 \le s \le 1$.*
- *$P_H(b)$ represents drawing $m$ bits i.i.d. Bernoulli$(p)$, $p \le 1/2$.*
- *$n \ge (1+\epsilon)\cdot m \cdot \log(1/p)$ where $\epsilon > 0$.*
- *The attacker knows which bin is mapped to which user.*
- *Passwords are drawn i.i.d. Bernoulli$(1/2)$.*

*The average number of guesses required to find a password that is mapped to any of the bins to which the passwords are mapped, can be characterized as follows.*

- *The probability that all passwords are mapped to different bins of type $q$ such that $0 \le 1-s < q \le p$, decreases like $e^{-2^{m\cdot(2\cdot H(1-s)-H(q))}} \times 2^{-m\cdot D(q||p)\cdot 2^{H(1-s)\cdot m}}$.*

Table I

THE RATE AT WHICH THE MOST LIKELY AVERAGE GUESSWORK INCREASES ALONG WITH THE LOWER AND UPPER BOUNDS. NOTE THAT AT $p = 1/2$ THE BOUNDS MEET. FURTHERMORE, THIS EXAMPLE ILLUSTRATES THAT INCREASING THE NUMBER OF USERS HAS A FAR WORSE EFFECT THAN BIAS AS STATED IS REMARK 9. WHEN THERE IS NO BIAS AND THE NUMBER OF USERS INCREASES LIKE $2^{m \cdot H(0.2)}$ THE AVERAGE GUESSWORK INCREASES AT THE SAME RATE AS IN THE CASE WHEN THE NUMBER OF USERS IS $2^{m \cdot H(0.1)}$ AND THE BIAS IS $p = 0.21$. FURTHERMORE, WHEN THE BIAS IS $p = 0.45$ THE AVERAGE GUESSWORK IS VERY CLOSE TO ONE.

| $p$, $1-s$ | $H(p) - H(1-s)$ | $D(1-s\|\|p)$ | $D(s\|\|p)$ |
|---|---|---|---|
| p=1/2, $1-s=0$ | 1 | 1 | 1 |
| $p = 0.45$, $1-s=0$ | 0.9948 | 0.8625 | 1.15 |
| $p = 1/2$, $1-s=0.2$ | 0.2781 | 0.2781 | 0.2781 |
| $p = 0.21$, $1-s=0.1$ | 0.2725 | 0.0622 | 1.5914 |

- *In this case the average guesswork increases like* $2^{m \cdot (H(q)+D(q||p)-H(1-s))}$.
- *Furthermore, the most likely type is* $q = p$ *in which case the average guesswork is* $2^{m \cdot (H(p)-H(s))}$.
- *Finally, when* $p \leq 1 - s \leq 1/2$ *the most likely type is again* $q = p$*, and the rate of the average guesswork is equal to* $0$.

*The average for any* $P_H(b)$*-hash function as well as for the* $P_H(b)$*-set of hash functions is done in accordance with Definition 10.*

*Proof:* The full proof appears in Section VIII. ∎

**Remark 9** (The effect of bias vs. the effect of the number of users)**.** *The effect of the number of users on the average guesswork is far worse than the effect of bias, that is, as the number of users increases the average guesswork decreases at a higher rate than the one when the number of users remains constant and bias increases. In order to illustrate this statement let us focus on the rate at which the most likely average guesswork increases,* $H(p) - H(1-s)$*, where* $0 \leq 1 - s \leq p$ *(although it holds for the bounds of Theorem 3 as well). First, let us focus on the effect of increasing the number of users on the average guesswork. The first derivative of* $H(p) - H(1-s)$ *with respect to* $p$ *is*

$$\log_2(p) - \log_2(1-p) \tag{23}$$

*which is equal to zero at* $p = 1/2$ *(i.e., when there is no bias), In addition, the first derivative around* $p = 1/2$ *is very small and therefore bias has a little effect on the average guesswork. On the other hand the rate at which the number of users increases is* $H(1-s)$*, and so the average guesswork decreases linearly with* $H(1-s)$*. This in turn shows that change in bias does not affect the average guesswork to the same extent as change in the number of users. We illustrate this observation in Table I.*

We now analyze the average guesswork under an online attack.

**Theorem 4** (Bounds on the average guesswork under online attacks)**.** *Under the following assumptions.*

- *The attacker does not know the mappings of the hash function.*
- *There are* $2^{H(s) \cdot m - 1}$ *users who are mapped to different bins, where* $1/2 \leq s \leq 1$.
- $P_H(b)$ *represents drawing* $m$ *bits i.i.d. Bernoulli*$(p)$, $p \leq 1/2$.
- $n \geq (1+\epsilon) \cdot m \cdot \log(1/p)$ *where* $\epsilon > 0$.
- *The attacker knows which bin is mapped to which user.*

*The rate of the average guesswork of any* $P_H(b)$*-hash function as well as the* $P_H(b)$*-set of hash functions, when averaged in according to Definition 10, is bounded by*

$$\begin{aligned} H(s) + D(1-s||p) &\leq \lim_{m \to \infty} \frac{1}{m} \log(E(G_T(B))) \\ &\leq \begin{cases} H(s) + D(s||p) & (1-p) \leq s \leq 1 \\ 2 \cdot H(p) + D(1-p||p) - H(s) & 1/2 \leq s \leq (1-p) \end{cases} . \end{aligned} \tag{24}$$

*Proof:* The full proof appears in Section VIII. ■

Next, we find the most likely average guesswork under online attacks.

**Corollary 4** (The most likely average guesswork under online attacks)**.** *When the following assumptions hold.*

- *The attacker does not know the mappings of the hash function.*
- *There are* $2^{H(s)\cdot m}$ *users,* $1/2 \leq s \leq 1$.
- $P_H(b)$ *represents drawing* $m$ *bits i.i.d. Bernoulli*$(p)$, $p \leq 1/2$.
- $n \geq (1+\epsilon)\cdot m \cdot \log(1/p)$ *where* $\epsilon > 0$.
- *Passwords are drawn i.i.d. Bernoulli*$(1/2)$.

*For any user, the average number of guesses required to find a password that is mapped to its bin can be characterized as follows.*

- *The probability that all passwords are mapped to different bins of type* $q$ *such that* $0 \leq 1-s < q \leq p$, *decreases like* $e^{-2^{m\cdot(2\cdot H(1-s)-H(q))}} \times 2^{-m\cdot D(q||p)\cdot 2^{H(1-s)\cdot m}}$.
- *In this case the average guesswork of any of the users increases like* $2^{m\cdot(H(q)+D(q||p))}$.
- *Furthermore, the most likely type is* $q = p$ *in which case the average guesswork is* $2^{m\cdot H(p)}$.

*The average for any* $P_H(b)$*-hash function as well as for the* $P_H(b)$*-set of hash functions is done in accordance with Definition 10.*

*Proof:* The full proof appears in Section VIII. ■

**Corollary 5** (The Average Guesswork when averaged over all passwords)**.** *When each user chooses its password uniformly, the average guesswork when averaging over the passwords increases at rate that is equal to*

$$\lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G_D\left(B\right)\right)\right) = 1$$

*where* $G_D(B)$ *is the guesswork of any user, and* $n \geq (1+\epsilon)\cdot m \cdot \log(1/p)$.

*Proof:* When $n \geq (1+\epsilon)\cdot m \cdot \log(1/p)$ the proof follows along the same line as Corollary 7. ■

Finally, we provide a concentration result for each bin that is stored on the sever, as well as under offline attacks.

**Theorem 5** (Concentration result under both online and offline attacks)**.** *Under the following assumptions.*

- *The attacker does not know the mappings of the hash function.*
- *There are* $2^{H(s)\cdot m}$ *bins to which the passwords are mapped,* $1/2 \leq s \leq 1$.
- $P_H(b)$ *represents drawing* $m$ *bits i.i.d. Bernoulli*$(p)$, $p \leq 1/2$.
- $n \geq (1+\epsilon)\cdot m \cdot \log(1/p)$ *where* $\epsilon > 0$.
- *The attacker knows which bin is mapped to which user.*

*For any of the bins to which the passwords are mapped, the following concentration result holds for any* $P_H(b)$*-hash function, and for the* $P_H(b)$*-set of hash functions*

$$-\lim_{m\to\infty} \frac{1}{m} \log\left(P\left(G\left(b\right) \leq 2^{(1-\epsilon_1)\cdot(H(q(b))+D(q(b)||p))\cdot m}\right)\right) \geq \epsilon_1 \cdot \log(1/p), \tag{25}$$

*where* $0 < \epsilon_1 < 1$ *and bin* $b$ *is of type* $q(b)$.

*This concentration result applies to the average guesswork of Theorem 3 as well.*

*Proof:* The proof is in Section VIII. ■

A few remarks are in order.

**Remark 10.** *From Theorem 5 we can state that for any* $P_H(b)$*-hash function and any bin, the fraction of strategies of guessing passwords one by one, for which the number of guesses is smaller than* $2^{(1-\epsilon_1)\cdot(H(q(b))+D(q(b)||p))\cdot m}$, *decreases like* $2^{-\epsilon_1\cdot\log(1/p)\cdot m}$*; in addition, for the* $P_H(b)$*-set of hash functions,* *any bin and any strategy of guessing passwords one by one, the fraction of $P_H(b)$-hash functions for which the number of guesses is smaller than $2^{(1-\epsilon_1)\cdot(H(q(b))+D(q(b)||p))\cdot m}$, also decreases like $2^{-\epsilon_1\cdot\log(1/p)\cdot m}$.*

**Remark 11.** *The fact that the average guesswork of every bin scales with its probability, as presented in Corollary 12, leads to average guesswork over all passwords that scales like $2^m$ which is larger than $2^{m\cdot H_{1/2}(p)}$.*

**Remark 12.** *The concentration result of Theorem 5 shows that the probability mass function of the average guesswork when cracking password is concentrated around its mean value. This is in contrast with the average guesswork for guessing a password [13] in which case the probability mass function is concentrated around the typical set in the i.i.d. case, whereas the average guesswork can be derived based on a large deviations argument [17].*

**Remark 13.** *In this subsection we assume that passwords are drawn uniformly. When passwords are biased there is a certain optimal strategy [13] and therefore, there is no use of averaging over all possible strategies. In this case averaging is done over the $P_H(b)$-set of hash functions as presented in Definition 10. In this case, similarly to Corollary 1 the average guesswork is equal to the dominant term between the average number of guesses required to guess* the password*, and the average number of guesses required to guess* a password *that is mapped to the same bin.*

### *C. The Third Main Result: The Average Guesswork of Strongly Universal Sets of Hash Functions (Keyed Hash Functions) with bins allocation*

In this subsection we show that when the average guesswork is larger than the number of users, the minimal size of a biased key that achieves the average guesswork above with a universal set of hash functions defined in subsection II-B equations (10), (11), is smaller than the size of a uniform key (i.e., a key which is i.i.d. Bernoulli$(1/2)$) that achieves the same guesswork for the same number of users, with any mapping from the set of keys to the set of hash functions.

**Theorem 6.** *When the number of users $M = 2^{m-1}$, a uniform key that achieves average guesswork $2^{m\cdot\alpha}$, where $\alpha > 1$, with any mapping from the set of keys to the set of all hash functions, is $\alpha$ times larger than a biased key which is i.i.d. Bernoulli$(p_0)$ that achieves the same average guesswork with the universal set of hash functions defined in subsection II-B equations* (10), (11), *where $p_0$ satisfies*

$$1 + D(1/2||p_0) = \alpha. \tag{26}$$

*Proof:* The proof follows directly from Corollary 9 and Corollary 10 of section VI. ■

Figure 5 illustrates Theorem 6.

## V. The Average Guesswork of a Keyed Hash Function

In this section we derive the average guesswork for strongly universal set of hash functions when the key is drawn i.i.d. Bernoulli$(p)$, $0 < p \leq 1/2$. In Subsection V-A we derive the average guesswork for any bin as a function of the bias $p$. Then, in Subsection V-B we calculate the average guesswork as a function of the number of users and the bias. Furthermore, we present the backdoor mechanism to allocate bins. Finally, in Subsection V-C we analyze the guesswork when the hash function is not modified.

### *A. The Guesswork of Each Bin*

In this subsection we derive the guesswork for each bin, when the key is i.i.d. Bernoulli$(p)$. Assume a strongly universal set of hash functions as defined in subsection II-B equations (10), (11). Note that although in Definition 12 we describe procedure according to which bins are allocated, this procedure is different from the backdoor mechanism which is given in Definition 13.

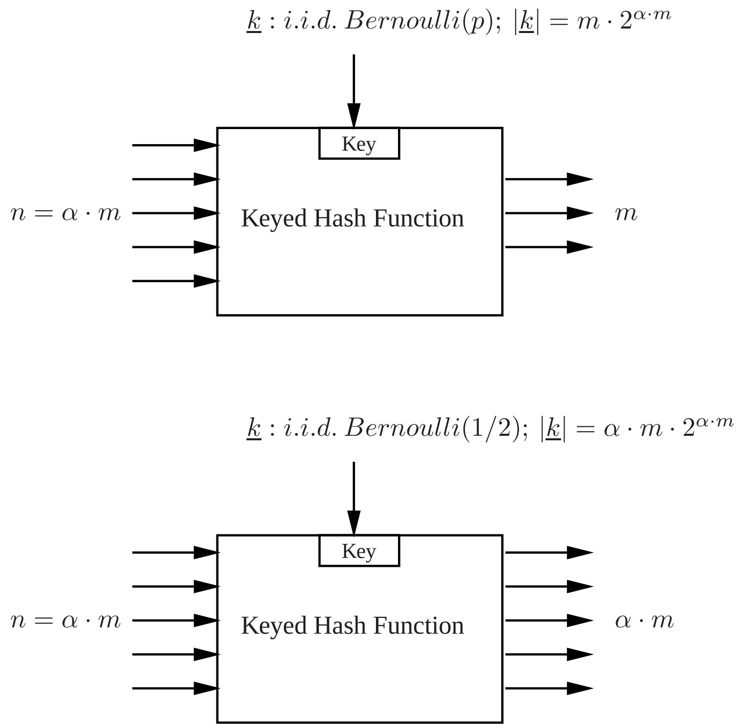


Figure 5. When the number of users is $2^{m-1}$, and the average guesswork is equal to $2^{\alpha \cdot m}$, the size of a biased key is $\alpha > 1$ times smaller than a uniform key, where $\alpha = 1 + D\,(1/2||p)$. Note that in order to achieve an average guesswork $2^{\alpha \cdot m}$, the output of a strongly universal set of hash functions has to be of size $\alpha \cdot m$ when the key is unbiased, whereas when the key is biased the output can reduce to $m$ bits, as long as the number of users is no larger than $2^{m-1}$.

**Definition 12** (Bins allocation)**.** *Assume there are $\lfloor 2^{H(s) \cdot m-1} \rfloor$ users, where $s \in [1/2, 1]$. For each user a procedure allocates a different integer $b \in \{1, \ldots, 2^m\}$ according to the following method: The first user receives the least likely bin (i.e., the all ones bin), the second user receives the second least likely bin, whereas the last user receives the $\lfloor 2^{H(s) \cdot m-1} \rfloor$th least likely bin. Then, assuming $L_b$ passwords are mapped to bin $b$, the procedure uniformly draws a password from $\{1, \ldots, |L_b|\}$. In the case when $|L_b| = 0$, the bin is not mapped to any password and we assume that the average guesswork is equal to zero.*

A remark is in order regarding the case where a user has no password.

**Remark 14.** *Note that the backdoor mechanism presented in Definition 13 does not allow a scenario where a user has no password. As shown in Theorem 10, this does not affect the average guesswork.*

The next lemma proves that the order according to which the attacker guesses passwords, does not affect the average guesswork.

**Lemma 3.** *For any bin $b$, the average guesswork is not affected by the order according to which the attacker guesses passwords one by one, when averaging over all possible keys.*

*Proof:* Consider a certain bin $b \in \{1, \ldots, 2^m\}$. The attacker knows the probability mass function according to which the key is drawn. He also knows the set of bins allocated to the $\lfloor 2^{H(s) \cdot m-1} \rfloor$ users, as presented in Definition 11. However, he does not know the actual realizations, and therefore the mapping from passwords to bins. Hence, from symmetry arguments, any strategy according to which the attacker guesses passwords one by one results in the same average guesswork, when averaging over all possible keys. ■

Based on Lemma 3 we assume without loss of generality that the attacker guesses passwords one by one in ascending order. Now, we can derive the average guesswork for any bin $b$.

**Theorem 7** (Average number of guesses for each bin)**.** *Assume that the attacker tries to guess a password that is mapped to bin $b$. In this case when $n \geq (1+\epsilon_1) \cdot \log(1/p) \cdot m$, $\epsilon_1 > 0$, the average guesswork of $G(b)$ is equal to*

$$E(G(b)) = 2^{m \cdot (H(q(b)) + D(q(b)||p))} - \left(1 - 2^{-m \cdot (H(q(b)) + D(q(b)||p))}\right)^{2^n} \cdot \left(2^{m \cdot (H(q(b)) + D(q(b)||p))} + 2^n\right) \tag{27}$$

*where $q(b) = \frac{N(1|b)}{m}$ is the type of $b$, the key is drawn i.i.d. Bernoulli$(p)$, and the universal set of hash functions is defined in subsection II-B equations* (10)*,* (11)*.*

*Proof:* Based on Lemma 3 and without loss of generality we assume that the attacker guesses passwords one by one in ascending order. Essentially, in order to crack bin $b$ the attacker has to find *a password* that is mapped to this bin. The proof relies on the observation that the chance of success in the $l$th guess (i.e., the first time a password which is mapped to bin $b$ is guessed) is drawn according to the following *geometric distribution*

$$P_K(b) \cdot (1 - P_K(b))^{l-1} \quad 1 \leq l \leq 2^n \tag{28}$$

where $P_K(\cdot)$ is the probability mass function of a vector of length $m$ that is drawn i.i.d. Bernoulli$(p)$. The probability that bin $b$ is not mapped to any password is

$$(1 - P_K(b))^{2^{n+1}}. \tag{29}$$

Since $P_K(b) \geq p^m$, we can state that as long as $n \geq (1+\epsilon_1) \cdot \log(1/p) \cdot m$ where $\epsilon_1 > 0$, the probability for this event vanishes exponentially fast and therefore does not affect the average guesswork; furthermore, we assume that when this event occurs, it takes zero guesses for the attacker to crack the bin, such that this event does not even add up to the average guesswork. We wish to stress that the "backdoor" procedure presented in Definition 13 does not allow for a situation where a user does not have a password.

The mean value of (28) is equal to

$$1/P_K(b) - (1 - P_K(b))^{2^n} \cdot (1/P_K(b) + 2^n). \tag{30}$$

Assuming bin $b$ is of type $q(b) = \frac{N(1|b)}{m}$, and that the key is drawn i.i.d. Bernoulli$(p)$, we get from Lemma 1 that

$$1/P_K(b) = 2^{m \cdot (H(q(b)) + D(q(b)||p))}. \tag{31}$$

By assigning the equation above in (30) we get (27). ■

Now, we show that the average guesswork converges uniformly to $2^{m \cdot (H(q(b)) + D(q(b)||p))}$, $\forall b$.

**Corollary 6.** *The average guesswork uniformly converges to $1/P_K(b)$ across bins. The difference is upper bounded by the following term.*

$$\begin{aligned} |2^{m \cdot (H(q(b)) + D(q(b)||p))} - E(G(b))| &\leq \left(2^{m \cdot (H(q(b)) + D(q(b)||p))} + 2^n\right) \cdot e^{-2^{n - m \cdot (H(q(b)) + D(q(b)||p))}} \\ &\leq \left(2^{m \log(1/p)} + 2^n\right) \cdot e^{-2^{n - m \log(1/p)}} \quad \forall b \end{aligned} \tag{32}$$

*Proof:* we wish to upper bound the term $\left(1 - 2^{-m \cdot (H(q(b)) + D(q(b)||p))}\right)^{2^n} \cdot \left(2^{m \cdot (H(q(b)) + D(q(b)||p))} + 2^n\right)$ for any $b \in \{1, \ldots, 2^m\}$ in order to show that it converges uniformly when $n \geq (1+\epsilon_1) \cdot \log(1/p)$, $\epsilon_1 > 0$. We begin by using the inequality

$$\left(1 - \frac{1}{x}\right)^x \leq e^{-1} \quad \forall x > 1 \tag{33}$$

in order to get

$$\left(1 - 2^{-m \cdot (H(q(b)) + D(q(b)||p))}\right)^{2^n} \leq e^{-2^{n - m \cdot (H(q(b)) + D(q(b)||p))}}. \tag{34}$$

Furthermore, since the least likely type $q(b_0) = 1$ occurs with probability $p^m = 2^{-m\log(1/p)}$, we get

$$H(q) + D(q||p) \leq \log(1/p) \quad 0 \leq q \leq 1 \tag{35}$$

with equality at q=1. Therefore, we get that $2^{-m\cdot(H(q(b))+D(q(b)||p))} \geq 2^{-m\log(1/p)}$ and $2^{m\cdot(H(q(b))+D(q(b)||p))} \leq 2^{m\log(1/p)}$. By assigning these terms along with the inequality in equation (34) we get the desired result. ∎

The next theorem shows that the guesswork concentrates around its mean value.

**Theorem 8** (Concentration)**.** *The probability of finding a password that is mapped to bin* $b$ *in less than* $2^{m\cdot l}$ *guesses, where* $l \leq n$*, is upper bounded by*

$$P\left(G(b) \leq 2^{m\cdot l}\right) \leq 1 - e^{-2\cdot 2^{-(H(q(b))+D(q(b)||p)-l)\cdot m}}. \tag{36}$$

*Therefore, whenever* $l < H(q(b)) + D(q(b)||p)$ *the probability of success in less than* $2^{m\cdot l}$ *attempts decays to zero.*

*Proof:* Based on the fact that the guesswork has a geometric distribution, we get that the chance of success within $2^{m\cdot l}$ guesses is equal to

$$P\left(G(b) \leq 2^{m\cdot l}\right) = P_K(b) \cdot \sum_{i=1}^{2^{m\cdot l}} (1 - P_K(b))^{i-1} = 1 - (1 - P_K(b))^{2^{m\cdot l}}. \tag{37}$$

The upper bound for $P\left(G(b) \leq 2^{m\cdot l}\right)$ is obtained by assigning $P_K(b) = 2^{-m(H(q(b))+D(q(b)||p))}$ to the following inequality

$$e^{-x} \leq 1 - \frac{x}{2} \quad x \in [0, 1.58]. \tag{38}$$

Therefore, when $l < H(q(b)) + D(q(b)||p)$ the probability of finding a password that is mapped to bin $b$ goes to zero. ∎

**Remark 15.** *Note that although the number of passwords scales exponentially with* $n$ *(i.e., there are* $2^n$ *passwords), the average number of guesses scales with the size of the bins* $m$*. This is due to the fact that cracking a password requires finding* a password *which is mapped to the same bin as the true password.*

### *B. The Average Guesswork for Cracking a Password of a User under Bins Allocation*

We later use the solution to the following optimization problem, in order to derive the optimal average guesswork for cracking a password of a user.

**Lemma 4.**

$$\max_{0\leq q\leq 1} 2\cdot H(q) + D(q||p) = 2\cdot H(p) + D(1-p||p) \tag{39}$$

*where the optimal solution occurs at* $q = 1 - p$*. Furthermore,* $2\cdot H(p) + D(1-p||p)$ *is a positive and unbounded function that monotonically increases as* $p$ *decreases.*

*Proof:* First let us break down the expression in (39).

$$2\cdot H(q) + D(q||p) = H(q) + q\cdot\log(1/p) + (1-q)\cdot\log(1/(1-p)). \tag{40}$$

The above expression is a concave function as a function of $q$ as it is the summation of two concave functions; hence, this function has a maximal value. By calculating the first derivative and making it equal to zero we get

$$\log\left(\frac{1-q}{q}\right) = \log\left(\frac{p}{1-q}\right). \tag{41}$$

Equality holds only when $q = 1 - p$. By assigning it to the expression on the left hand side in (39) we get the desired result. ∎

We now quantify the number of mappings from passwords to each bin.

**Lemma 5.** *Following the definition in subsection II-B equations* (10), (11), *assume that* $k_i$ *is the mapping from the* $i$*th password to the range of the hash function, where* $k_i \in \{1,\ldots,2^m\}$ *and* $i \in \{1,\ldots,2^n\}$*. In this case*

$$P\left(|2^{-n}\cdot\sum_{i=1}^{2^n}\mathbb{1}_b\left(k_i\right)-P_K\left(b\right)|>\epsilon\right)\leq\frac{1}{2^n\cdot\epsilon^2} \tag{42}$$

*where* $\mathbb{1}_b\left(x\right)=\left\{\begin{array}{ll}1 & x=b\\ 0 & x\neq b\end{array}\right.$.

*Proof:* Essentially $1_b\left(k_i\right)$ is a random variable whose mean value is equal to $P_K\left(b\right)$ and variance value is smaller than 1. Furthermore, $1_b\left(k_i\right)$, $i\in\{1,\ldots,2^n\}$ are all i.i.d. random variables. Therefore, we get (42) from Chebyshev's inequality. ■

Now we can derive the maximal average guesswork.

**Theorem 9** (The average number of guesses)**.** *Let us denote the number of users by* $\lfloor 2^{H(s)\cdot m-1}\rfloor$*, where* $s\in[1/2,1]$ *is a parameter. The exponential rate at which the optimal average guesswork increases as a function of* $m$ *is equal to*

$$\lim_{m\to\infty}\frac{1}{m}\log\left(E\left(G_T\left(B\right)\right)\right)=\left\{\begin{array}{cc}H\left(s\right)+D\left(s||p\right) & \left(1-p\right)\leq s\leq 1\\ 2\cdot H\left(p\right)+D\left(1-p||p\right)-H\left(s\right) & 1/2\leq s\leq\left(1-p\right)\end{array}\right. \tag{43}$$

*where* $n\geq\left(1+\epsilon_1\right)\cdot m\cdot\log\left(1/p\right)$, $\epsilon_1>0$.

*Proof:* The general idea behind the proof is to write the average in terms of summation over types, and then use the fact that a vector of length $m$ has up to $m$ types in order to bound the average by the most dominant term among the $m$ types.

First let us define the set of all possible bins $\{b_1,\ldots,b_{2^m}\}$ such that $P_K\left(b_i\right)\geq P_K\left(b_j\right)$ if and only if $j<i$; therefore, according to Definition 11 $b_1$ is assigned to the first user, whereas $b_{2^{H(s)\cdot m}}$ goes to the $2^{H(s)\cdot m-1}$th user. The average guesswork when the attacker chooses a user name to attack, uniformly from $\left\{1,\ldots,2^{H(s)\cdot m-1}\right\}$, is equal to

$$E\left(G_T\left(B\right)\right)=2^{-H(s)\cdot m+1}\sum_{i=1}^{2^{H(s)\cdot m-1}}\frac{1}{p_K\left(b_i\right)}-\epsilon\left(m,n,q\left(b_i\right)\right) \tag{44}$$

where $\epsilon\left(m,n,q\left(b_i\right)\right)=\left(2^{m\cdot(H(q(b))+D(q(b)||p))}+2^n\right)\cdot e^{-2^{n-m\cdot(H(q(b))+D(q(b)||p))}}$ in accordance with Corollary 6. Note that the term above takes into consideration the event when a mapping to $b$ does not occur within the $2^n$ inputs; this probability is multiplied by zero guesses, which leads to the expression in equation (44).

Based on the method of types [7] every bin in $\left\{b_1,\ldots,2^{H(s)\cdot m-1}\right\}$, which is the set of bins that are allocated to the $2^{H(s)\cdot m-1}$ users, is of type $s\leq q\left(b_i\right)\leq 1$, where $1\leq i\leq 2^{H(s)\cdot m-1}$, and $q\left(b_i\right)\geq q\left(b_j\right)$ if and only if $i<j$. Therefore, from Lemma 1 and Lemma 2 we can rewrite the summation such that is goes across types, such that we can bound the average guesswork by the following terms

$$\frac{1}{\left(m+1\right)^2}2^{-H(s)\cdot m+1}\max_{s\leq q\leq 1}\left(2^{m\cdot(2\cdot H(q)+D(q||p))}-\epsilon\left(m,n,q\right),0\right)\leq E\left(G_T\left(B\right)\right)\leq 2^{-H(s)\cdot m+1}\sum_{s\leq q\leq 1}2^{m\cdot(2\cdot H(q)+D(q||p))}. \tag{45}$$

The number of types is equal to the size of each bin, and therefore there are only $m$ types; furthermore, the elements in the summations above all are non-negative. Hence, when $n\geq\left(1+\epsilon_1\right)\cdot m\cdot\log\left(1/p\right)$ and

$m \gg 1$ the following terms bound the average guesswork.

$$\frac{2^{-H(s)\cdot m+1}}{(m+1)^2}\cdot 2^{m\cdot \max_{s\leq q\leq 1}(2\cdot H(q)+D(q||p))} \leq E\left(G_T\left(B\right)\right) \leq m\cdot 2^{-H(s)\cdot m+1}\cdot 2^{m\cdot \max_{s\leq q\leq 1}(2\cdot H(q)+D(q||p))}. \quad (46)$$

We are interested in the rate at which the guesswork grows, which based on the above bound is equal to

$$\lim_{n\to\infty}\frac{1}{m}\log\left(E\left(G_t\left(B\right)\right)\right) = -H\left(s\right) + \max_{s\leq q\leq 1}\left(2\cdot H\left(q\right)+D\left(q||p\right)\right). \quad (47)$$

The function $2\cdot H(q)+D(q||p)$ is concave as a function of $q$, with a maximal value at $q=1-p$ as shown in Lemma 4. Therefore, when $(1-p)\leq s\leq 1$ the maximal value is obtained at $q=s$; whereas when $1/2\leq s\leq(1-p)$ the maximal value $2\cdot H(1-p)+D(q||p)=2\cdot H(p)+D(q||p)$ is achieved for $q=1-p$. Note that $n\geq\log(1/p)\geq H(q)+D(q||p)$ for $0\leq q\leq 1$ which leads to the elimination of $\epsilon(n,m,q)$. These arguments conclude the proof. ∎

We now present a backdoor mechanism that enables to modify a hash function efficiently without decreasing the average guesswork.

**Remark 16** (A Back door Mechanism for Allocating bins)**.** *Essentially, in order for a procedure to allocate bins to the users, it first allocates a bin to a user, and then maps the password of this user to this bin; furthermore, there is a certain chance that because of the key realization, there is no mapping to a specific bin. This operation is as exhaustive as cracking the bin itself. In order for the system to allocate bins efficiently it has to use a* backdoor *that enables it to allocate bins without the need to crack the hash function, as well as without compromising the security of the system (i.e., maintaining the same average guesswork). In Definition 13 and Theorem 10 we present a back door mechanism that satisfies both of these requirements.*

Now we define a back door mechanism that enables to plant mappings in an efficient way without compromising the security level of the hash function.

**Definition 13.** *Given a key $K=\{k_1,\ldots,k_{2^n}\}$ where $k_i\in\{1,\ldots,2^m\}$ and $1\leq i\leq 2^n$, and a strongly universal set of hash functions as defined in subsection II-B equations* (10), (11)*, the back door mechanism for allocating bins is defined as follows.*

- *For each user choose a different bin according to the procedure described in Definition 11 (i.e., the first user gets the least likely bin $b_1$, the second user receives the second least likely bin $b_2$, and the $2^{H(s)\cdot m-1}$th user receives $b_{2^{H(s)\cdot m-1}}$).*
- *Each of the users from the first to the $2^{H(s)\cdot m-1}$th draws a password uniformly by drawing $n$ bits i.i.d. Bernoulli$(1/2)$.*
- *For each user, if the number that was drawn $g\in\{1,\ldots,2^n\}$ has not been drawn by any of the users that have bins that are less likely than the current bin, then replace $k_g$ with the bin number allocated to the user.*
- *In the case when another user who is coupled with a less likely bin, has drawn $g$:* Do not change *the value of the key again (i.e., $k_g$ is changed only once by the user who is coupled with the least likely bin among the bins of the users whose password is $g$). Instead, change the bin allocated to the user to the least likely bin among the bins allocated to users whose password is $g$. Therefore, $k_g$ is changed only once, to the least likely bin that is mapped to $g$.*

Figure 6 illustrates the back door mechanism.

**Theorem 10.** *The optimal average guesswork when allocating bins using the back door mechanism given in Definition 13 is the same as the guesswork from Theorem 9 and is equal to*

$$\lim_{m\to\infty}\frac{1}{m}\log\left(E\left(G_T\left(B\right)\right)\right) = \begin{cases} H\left(s\right)+D\left(s||p\right) & (1-p)\leq s\leq 1 \\ 2\cdot H\left(p\right)+D\left(1-p||p\right)-H\left(s\right) & 1/2\leq s\leq(1-p) \end{cases} \quad (48)$$

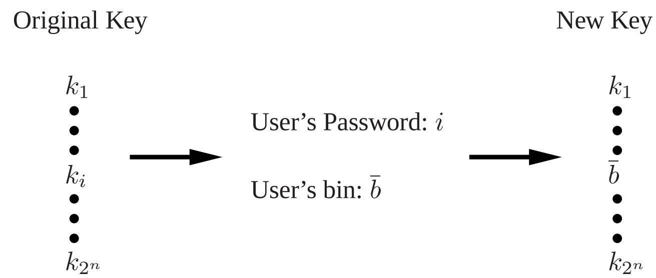


Figure 6. The backdoor mechanism. When the password of the user is equal to $i$, and no other user who is coupled with a bin that is less likely than the bin of the user, has come up with this password, the $i$th segment of the key is replaced with the bin that is coupled with the user. In the case when another user who is coupled with a less likely bin, has already come up with the exact same password, the more likely bin of the user (and not the key) changes to the bin of the other user.

when $(1+\epsilon_1)\cdot m\cdot\log(1/p)$, $\epsilon_1>0$.

*Proof:* The general idea behind the proof is that the back door procedure described in Definition 13 can not add more than one mapping to each bin. Therefore, given any strategy of guessing passwords one by one, we show that the chance of drawing a password (in the backdoor procedure) that decreases the number of guesses bellow the average guesswork vanishes exponentially fast.

First, a few words are in order regarding the effect of the backdoor procedure presented in Definition 13. The procedure can not add more than one mapping to each bin. This is due to the fact that when a password that is mapped to a specific bin is drawn, then the mapping is changed to the bin of the user for which the password was drawn. Furthermore, if the same password is drawn more than once by several users, then the mapping does not change again; at worst more than one user is mapped to the same bin (the probability that two users have the same password is the same as the probability of a collision, and is equal to $2^{-n}$). When a collision occurs, the mapping to the least likely bin among the bins that are coupled with these users, is the one assigned to all these users. Therefore, collision does not decrease the average guesswork.

Following what we have discussed above, for any strategy we are interested in the probability of drawing a password which is guessed within a smaller number of attempts than the average guesswork (i.e., given any strategy of guessing passwords one by one, what is the probability that the backdoor mechanism decreases the number of guesses bellow the average guesswork). When this event occurs we assume that the number of guesses required to break the strongly universal set of hash functions is zero. When there are $2^{H(s)\cdot m-1}$ users, the set of type $q(b)=\max(s,1-p)$ bounds the average guesswork as shown in equation (46). Therefore, it is sufficient to focus on the bins of this type; as even when the guesswork of bins of other type is equal to zero, the average guesswork remains the same.

Without loss of generality let us focus on the case when passwords are guessed one by one in ascending order; the following argument holds for any other strategy of guessing passwords. For every user, the backdoor mechanism draws a password uniformly. When $n\geq(1+\epsilon_1)\cdot m\cdot\log(1/p)$ as defined in Theorem 9, the probability of drawing a password that is guesses within a smaller number of attempts than $(1+\epsilon_2)\cdot m\cdot\log(1/p)$ where $0<\epsilon_2<\epsilon_1$ is

$$\frac{2^{(1+\epsilon_2)\cdot m\cdot\log(1/p)}}{2^n}\leq 2^{-(\epsilon_1-\epsilon_2)\cdot\log(1/p)\cdot m}. \tag{49}$$

Thus, for each bin of type $q(b)=\max(s,1-p)$, the average guesswork is multiplied by a factor $\left(1-2^{-(\epsilon_1-\epsilon_2)\cdot\log(1/p)\cdot m}\right)$ that approaches $1$ exponentially fast as $m$ increases. Therefore, the average guesswork is not affected by the back door mechanism. The only difference compared to Theorem 9 is that now the decay of $\epsilon(n,m,1-p)$ in is dictated by $\epsilon_2$ instead of $\epsilon_1$. ■

**Remark 17.** *Note that the procedure for allocating bins which is described in Definition 13, does not require knowledge of the key in order to achieve the average guesswork of Theorem 10. The back door* *mechanism requires only knowledge of which bins are the least likely to occur based on the probability mass function according to which the key is drawn.*

### *C. The Average Guesswork when Bins are not Allocated to Users*

In this subsection we show that when the users choose their own passwords, without changing the hash function accordingly, the average guesswork of the strongly universal set of hash functions (averaged over all passwords) is equal to $2^m$ for any $p$.

**Definition 14** (No bins allocation)**.** *When bins are not allocated to users each user chooses his own password, without modifying the hash function. The server stores the bin to which a password is mapped.*

**Corollary 7** (The Average Guesswork of the strongly universal set of hash functions with no bins Allocation)**.** *When each user chooses the password uniformly over* $\{1,...,2^n\}$*, the average guesswork* when averaging over the passwords, *increases at rate that is equal to*

$$\lim_{m\to\infty}\frac{1}{m}\log\left(E\left(G_D\left(B\right)\right)\right)=1$$

*where* $G_D\left(B\right)$ *is the guesswork of any user,* $n\geq(1+\epsilon_1)\cdot m\cdot\log\left(1/p\right)$*, and the hash function is defined in subsection II-B equations* (10), (11)*.*

*Proof:* From Theorem 7 and equation (30) we can state that the average guesswork for each bin is equal to

$$E\left(G_D\left(b\right)\right)=1/P_K\left(b\right)-\left(1-P_K\left(b\right)\right)^{2^n}\cdot\left(1/P_K\left(b\right)+2^n\right)\quad b\in\{1,...,2^m\}\,. \tag{50}$$

From Corollary 6 we know that the right hand side of the equation above goes to zero when $n\geq(1+\epsilon_1)\cdot m\cdot\log\left(1/p\right)$.

When a user chooses a password uniformly over $\{1,...,2^n\}$, and the key is drawn i.i.d. Bernoulli$(p)$, the probability of average guesswork that is equal to $E\left(G\left(b\right)\right)$ (i.e., the probability of drawing a password that is mapped to $b$) is equal to $P_K\left(b\right)$. Hence, we get that the average guesswork is equal to

$$E\left(G_D\left(B\right)\right)=\sum_{b=1}^{2^m}P_K\left(b\right)E\left(G_D\left(b\right)\right) \tag{51}$$

which leads to

$$\lim_{m\to\infty}\log\left(E\left(G_D\left(B\right)\right)\right)=1. \tag{52}$$

■

**Remark 18.** *Note that when the least likely bins are allocated to the users (e.g., bins are allocated as in Definition 13), the fact that there is a very small number of mappings to these bins, enables to achieve a better average guesswork with a shorter biased key (as we show in the next section). However, when the bins are not allocated (i.e., the hash function is not modified), the chance of drawing a password that is mapped to a certain bin is equal to the probability of mapping a password to the very same bin. Therefore, in this case there is a small probability of drawing a bin that has a very small number of passwords mapped to it. This leads to an average guesswork that grows like* $2^m$*.*

## VI. Uniform Keys Versus Biased Keys: Size and Storage Requirements

In this subsection we show that when the number of users is smaller than the number of bins, a shorter biased key can achieve the same level of security as a longer unbiased key. Furthermore, we show that the storage space required to store the bins of all users is also significantly smaller.

First, we show that the average guesswork of biased keys is larger than the average guesswork of any strongly universal set of hash functions whose key is drawn i.i.d. according to Bernoulli$(1/2)$, that is, following the definitions of Section II-B, any mapping from the set of keys to the set of hash functions would lead to the same guesswork when the key is uniform.

**Corollary 8.** *When $n \geq m$, the guesswork of any strongly universal set of hash functions whose key is i.i.d. Bernoulli$(1/2)$ satisfies*

$$\lim_{m\to\infty} \frac{1}{m} \log \left( G_T^{(u)} \left(B, f\left(\cdot\right)\right)\right) = 1 \leq \lim_{m\to\infty} \frac{1}{m} \log \left(G_T\left(B\right)\right) \quad 0 \leq s \leq 1 \tag{53}$$

*with equality only when $p = 1/2$; where $f(\cdot)$ is a bijection from the set of keys to the set of all possible hash functions. Therefore, whenever $0 < p < 1/2$, the average guesswork of the strongly universal set of hash functions defined in subsection II-B equations* (10), (11)*, is larger than the one achieved by any mapping $f(\cdot)$ over a balanced key.*

*Proof:* When the key is drawn i.i.d. Bernoulli$(1/2)$ the problem of finding the average guesswork can be regarded as a counting problem. Since $f(\cdot)$ is a bijection, the number of functions for which the $i$th password is mapped to the $j$th bin is the same for any $i$ and $j$. Furthermore, when $l$ mappings are revealed, the set of keys decreases by a factor of $2^{l\cdot m}$ and the remaining mappings are still balanced. Therefore, for any bin the chance of cracking a password in the $l$th guess is a geometric distribution that equals to $2^{-m} \cdot \left(1-2^{-m}\right)^{l-1}$, $l \geq 0$; this holds whenever $f(\cdot)$ is a bijection. Hence, the rate at which the average guesswork increases when $n \geq (1+\epsilon)\cdot m$ and $\epsilon > 0$, is equal to

$$\lim_{m\to\infty} \log \left( G_T^{(u)} \left(B, f\left(\cdot\right)\right)\right) = 1 \quad s \in [1/2, 1] \tag{54}$$

whereas from Theorem 9 we get that

$$\lim_{m\to\infty} \log \left(G_T\left(B\right)\right) \geq 2 \cdot H\left(p\right) + D\left(1-p||p\right) - 1 \geq 1 \tag{55}$$

with equality only when $p = 1/2$, where the second inequality results from the fact that $2 \cdot H(p) + D(1-p||p)$ decreases as $p$ approaches $1/2$, and is equal to $2$ at $p = 1/2$. ∎

We now decrease the minimal size of the input, $n$, such that when $1/2 \leq s \leq (1-p)$, the average guesswork is smaller than the one in (43); by decreasing $n$, we later show that in some cases the size of a biased key required to achieve a certain average guesswork, is significantly smaller than the size of an unbiased key that achieves the same guesswork.

**Lemma 6.** *When the number of users is $2^{H(s)\cdot m-1}$, $n = (1+\epsilon_1)\cdot m\cdot(H(s) + D(s||p))$ where $1/2 \leq s \leq 1$, and the key is drawn i.i.d. Bernoulli$(p)$, the average guesswork increases at rate that is equal to*

$$\lim_{m\to\infty} \frac{1}{m} \log \left(E\left(G_T\left(B\right)\right)\right) = H\left(s\right) + D\left(s||p\right). \tag{56}$$

*Proof:* The general idea behind the proof is related to the fact that when the number of users is $2^{m\cdot H(s)-1}$, and the users are coupled with the $2^{m\cdot H(s)-1}$ least likely bins, the average guesswork of any user is larger than or equal to $2^{m\cdot(\cdot H(s)+D(s||p))}$ as long as $n \geq (1+\epsilon_1)\cdot m \cdot \log(1/p)$; this follows from Theorem 7, the fact that $s \leq q(b_i) \leq 1$ where $i \in \left\{1, \ldots, 2^{m\cdot H(s)-1}\right\}$, and also because the backdoor mechanism does not decrease the guesswork of any bin. However, here we consider the case when the input is equal to $n = (1+\epsilon_1)\cdot m \cdot (H(s) + D(s||p))$, for which some of the assumptions above may not hold. Therefore, we examine two cases: The case where $q(b_i) = s$, as well as $s < q(b_i) \leq 1$. Once we show that for both cases the rate at which the average guesswork increases is larger than or equal to $H(s) + D(s||p)$, we can also show that the average guesswork across all users increases at this rate.

For $q(b_i) = s$ the size of the input supports the average guesswork, and so following along the same lines as Theorem 9 and Theorem 10, we get that the average guesswork of these bins increases at rate $H(s) + D(s||p)$.

For $s < q(b_i) \leq 1$ the probability of a mapping to these bins is $P_K(b_i) =^{-m\cdot(H(q(b_i))+D(q(b_i)||p))}$, where $H(q(b_i)) + D(q(b_i)||p) > H(s) + D(s||p)$. Therefore, the average guesswork of these elements has to be larger than or equal to the case when $q(b_i) = s$. In fact, when $\epsilon_1$ is small enough and $n = (1+\epsilon_1)\cdot m\cdot(H(s)+D(s||p))$, the probability that there is a mapping to bin $b_i$ within the $2^n$ inputs goes to zero, and so the only occurrences of $b_i$ is due to the backdoor mechanism; the backdoor mechanism achieves average guesswork that increases at rate $(1+\epsilon_1)\cdot(H(s)+D(s||p))$ when the password is drawn uniformly in accordance with Definition 13 (again, following the same lines as Theorem 10).

■

**Remark 19.** *In order to show that a shorter biased key achieves the same guesswork that a longer balanced key achieves we need to find the actual guesswork and not only the rate at which it increases. The next lemma resolves this issue.*

Now, we derive bounds for the actual average guesswork and not only for the rate at which it increases.

**Lemma 7.** *When the number of users is* $2^{H(s)\cdot m-1}$, $1/2 \le s \le 1$, *and* $n = (1+\epsilon_1)\cdot m\cdot(H(s)+D(s||p))$ *the average guesswork of the universal hash function defined in subsection II-B equations* (10), (11)*, a key which is i.i.d. Bernoulli*$(p)$*, and an output of size* $m$*, is lower bounded by*

$$E(G_T(B)) \ge (1-\gamma(m,p,s))\cdot 2^{m\cdot(H(s)+D(s||p))} - e^{-2^{\epsilon_1\cdot m}}\cdot\left(2^{m\cdot(H(s)+D(s||p))}+2^n\right) \tag{57}$$

[2] *where* $\gamma(s,p,m)$ *decays exponentially to zero (i.e.,* $\lim_{m\to\infty}\gamma(s,p,m) = 0$*). On the other hand, for any universal set of hash functions, with a key that is i.i.d. Bernoulli*$(1/2)$*, and an output of size* $(H(s)+D(s||p))\cdot m$*, the average guesswork is equal to*

$$E\left(G_T^{(u)}(B), f(\cdot)\right) = 2^{m\cdot(H(s)+D(s||p))} - \left(1-2^{-m\cdot(H(s)+D(s||p))}\right)^{2^n}\cdot\left(2^{m\cdot(H(s)+D(s||p))}+2^n\right). \tag{58}$$

*Therefore, we get that*

$$\lim_{m\to\infty}\frac{E(G_T(B))}{E\left(G_T^{(u)}(B), f(\cdot)\right)} \ge 1.$$

*Proof:* Let us start by proving the inequality in (57). When $n = (1+\epsilon_1)\cdot m\cdot(H(s)+D(s||p))$, the probability of a bin of type $q(b) > s$ to be mapped by the key to any password decreases according to the expression

$$1-\left(1-2^{-m\cdot(H(q(b))+D(q(b)||p))}\right)^{2^{(1+\epsilon_1)\cdot m\cdot(H(s)+D(s||p))}}; \tag{59}$$

this is because of the fact that when $q(b) > s$ the term $(H(q(b))+D(q(b)||p)) > (H(s)+D(s||p))$. The back door mechanism in Definition 13 plants a mapping by drawing a password uniformly. Therefore, we get with probability $(1-\gamma_1(s,p,m))$ where $\gamma_1(s,p,m)$ decays to zero exponentially fast (i.e., with probability that approaches $1$ very quickly), that when $q(b) > s$ the problem of cracking a bin is as hard as guessing the mapping created by the back door mechanism. Since in this case the password is uniformly distributed over all possible passwords (i.e., $2^n$ possibilities), it can be easily shown that the average guesswork is equal to $\frac{2^n}{2} \ge 2^{(H(s)+D(s||p))\cdot m}$ when $m \ge m_0$. Finally, by assigning the right parameters to equations (27), (32) when $q(b) = s$, the average guesswork is lower bounded by

$$G(b|q(b)=s) \ge (1-\gamma_2(s,p,m))\cdot 2^{(H(s)+D(s||p))\cdot m} - e^{-2^{\epsilon_1\cdot m}}\cdot\left(2^{m\cdot(H(s)+D(s||p))}+2^n\right). \tag{60}$$

Hence, the average guesswork over all users (or alternatively all bins for which $s \le q(b) \le 1$) is lower bounded by the term in equation (57).

In the case when the key is uniform, (58) is proven by assigning the right parameters in equation (27) and keeping in mind that when the key is uniform and $f(\cdot)$ is a bijection, the average guesswork is the same for any bin. These arguments prove the equality in equation (58). ■

[2]Note that the result from Theorem 10 can also be extended to a lower bound when $(1-p) \le s \le 1$ by arguments that follow the same lines as Lemma 7.

**Remark 20.** *Note that when $s = 1/2$, the number of users is $\sum_{i=1}^{m/2} \binom{m}{i}$. Thus, in this case as $m$ increases we get that $\frac{\sum_{i=1}^{m/2} \binom{m}{i}}{2^m} = 1/2$. By doing so we do not allocate bins that have an average guesswork which is smaller than the total average guesswork. Furthermore, in our analysis we restrict the size of the input, $n$, to be proportional to $m$. However, in practice $n$ can be much larger. This can be achieved by simply duplicating the mappings for the first $(H(s) + D(s||p)) \cdot m$ inputs over and over again. By doing so the number of inputs can increase as much as needed, whereas the size of the key and the total average guesswork remain the same. Finally, by allocating a bin to each user and performing the back door procedure in Definition 13, the chance of collision between any two users is the same as the chance of collision for a strongly universal hash function with a uniform key, that has the same average guesswork, that is, $2^n$.*

We now show that when the average guesswork is larger than the number of users, a biased key which "beamforms" to the subset of users, requires a key of smaller size than an unbiased key that achieves the same level of security.

**Theorem 11.** *When the number of users is $2^{H(s)\cdot m-1}$, $1/2 \le s \le 1$, any universal set of hash functions for which $f(\cdot)$ is a bijection, requires an unbiased key $k_u$ (i.e., which is drawn i.i.d. Bernoulli$(1/2)$) of size*

$$|k_u| = (H(s) + D(s||p)) \cdot m \cdot 2^{(H(s)+D(s||p))\cdot m} \tag{61}$$

*and output of size $|m_u| = (H(s) + D(s||p)) \cdot m$, in order to achieve an average guesswork for which*

$$\lim_{m\to\infty} \frac{E\left(G_T^{(u)}(B, f(\cdot))\right)}{2^{(H(s)+D(s||p))\cdot m}} = 1.$$

*On the other hand when the key is drawn Bernoulli$(p)$ the universal set of hash functions defined in subsection II-B equations* (10), (11), *achieves the same average guesswork with a key $k_b$ of size*

$$|k_b| = m \cdot 2^{(H(s)+D(s||p))\cdot m} \tag{62}$$

*and an output of size $m$. In both cases the minimal input is of size $n = (1 + \epsilon_1) \cdot (H(s) + D(s||p)) \cdot m$. The average size of the biased key can be decreased further to $H(p) \cdot |k_b|$ through entropy encoding.*

*Finally, distributing passwords according to the procedure in Definition 13 requires an unbiased key of size*

$$(1 + \epsilon_1) \cdot (H(s) + D(s||p)) \cdot m \cdot 2^{H(s)\cdot m}. \tag{63}$$

*Proof:* From Corollary 8 and Lemma 7 we get that when the key is drawn i.i.d. Bernoulli$(1/2)$, the average guesswork is determined by the size of the output. Therefore, in order to achieve an average guesswork that equals $2^{m\cdot(H(s)+D(s||p))}$ the size of the output must be $(H(s) + D(s||p)) \cdot m$. In addition $n = (1 + \epsilon_1 \cdot)(H(s) + D(s||p))$ and so the size of the key is equal to $|k_u| = (H(s) + D(s||p)) \cdot m \cdot 2^{(1+\epsilon_1)\cdot m\cdot(H(s)+D(s||p))}$.

For the case when the key is drawn i.i.d. Bernoulli$(p)$, we get from Lemma 7 that an output of size $m$ achieves the desired guesswork. Hence, in this case the size of the key is $|k_b| = m \cdot 2^{(1+\epsilon_1)\cdot m\cdot(H(s)+D(s||p))}$

Finally, in order to allocate bins according to the method in Definition 13 each user has a key of size $n$, and since there are $2^{H(s)\cdot m}$ users the size of the key required to draw passwords is $(1 + \epsilon_1) \cdot (H(s) + D(s||p)) \cdot m \cdot 2^{H(s)\cdot m}$. ■

**Corollary 9.** *When the number of users is equal to $2^{H(s)\cdot m-1}$ and the average guesswork increases at rate $(H(s) + D(s||p))$, the ratio between the size of an unbiased key and a biased key which is drawn Bernoulli$(p)$, when both keys achieve the average guesswork above, is*

$$\lim_{m\to\infty} |k_u|/|k_b| = H(s) + D(s||p).$$

*Proof:* The proof results from the fact that

$$\lim_{m\to\infty}\frac{\left(H\left(s\right)+D\left(s||p\right)\right)\cdot m\cdot 2^{(1+\epsilon_1)\cdot m\cdot(H(s)+D(s||p))}}{(1+\epsilon_1)\cdot\left(H\left(s\right)+D\left(s||p\right)\right)\cdot m\cdot 2^{H(s)\cdot m}+m\cdot 2^{(1+\epsilon_1)\cdot m\cdot(H(s)+D(s||p))}}=H\left(s\right)+D\left(s||p\right).$$

■

In the next corollary we find the minimal size of a key that enables to achieve through bias, a desired average guesswork that is larger than the number of users.

**Corollary 10.** *When there are $2^{m-1}$ users and the desired average guesswork increases at rate $\alpha>1$, the key of minimal size is drawn Bernoulli$(p_0)$ such that*

$$1+D\left(1/2||p_0\right)=\alpha. \tag{64}$$

*A universal set of hash functions defined in subsection II-B equations* (10), (11)*, whose output is of size $m$ achieves the average guesswork above.*

*Proof:* From Corollary 8 and Lemma 7 we know that when $\alpha>1$ and the key is i.i.d. Bernoulli$(1/2)$, the size of the output has to be $\alpha\cdot m$ in order to achieve an average guesswork $2^{\alpha\cdot m}$. Since in this case the size of the input has to be larger than $\alpha\cdot m$, the size of an unbiased key that achieves the desired average guesswork is $\alpha\cdot m\cdot 2^{\alpha\cdot m}$.

From Lemma 7 we also know that for any biased key that achieves average guesswork $2^{\alpha\cdot m}$ the size of the input has to be at least $\alpha\cdot m$. However, the size of the output can in fact be smaller than $\alpha\cdot m$. Essentially, we can define the output as $m'$ such that number of users $2^{m-1}=2^{H(s_0)\cdot m'-1}$ and $m'<\alpha\cdot m$, where $1/2\leq s_0\leq 1$. The size of the key in this case is equal to $m'\cdot 2^{\alpha\cdot m}$. The minimal value of $m'$ that still enables to allocate a different bin to each user is $m'=m$ (i.e., $s=1/2$). In this case the size of the key is $m\cdot 2^{\alpha\cdot m}$; this key is drawn i.i.d. Bernoulli$(p_0)$ such that

$$1+D\left(1/2||p_0\right)=\alpha.$$

■

Now, we show that biased keys also require smaller space to store the bins on the server.

**Corollary 11.** *When the number of users is $2^{H(s)\cdot m-1}$ and the average guesswork is equal to $2^{m\cdot(H(s)+D(s||p))}$, a biased key which is drawn Bernoulli$(p)$ requires to store $2^{H(s)\cdot m-1}$ bins of size $m$ each, whereas an unbiased key requires to store $2^{H(s)\cdot m-1}$ bins of size $\left(H\left(s\right)+D\left(s||p\right)\right)\cdot m$. Therefore, under the constraints above a biased key decreases the storage space by a factor of $\left(H\left(s\right)+D\left(s||p\right)\right)$. The average stored space can be reduced further to $H\left(p\right)\cdot m\cdot 2^{H(s)\cdot m-1}$ through entropy encoding.*

*Proof:* The proof is straight forward and results from the fact that when considering a biased key the output is $H\left(s\right)+D\left(s||p\right)$ times smaller than the output when the key is unbiased. ■

A remark is in order regarding the results above.

**Remark 21.** *Note that the results above apply to strongly universal sets of hash functions, which are balanced sets. On the other hand in sections VII and VIII we consider sets of hash functions which are not balanced, that is, the size of a conditional set depends on the actual values of the bins on which it is conditioned; for example, if a fraction of $\left(1-P_K\left(b_0\right)\right)$ mappings is revealed such that none of these mappings map to $b_0$, then the remaining fraction of $P_k\left(b_0\right)$ mappings are all mapped to $b_0$. Hence, in this case the conditional set is of size $1$.*

## VII. Guesswork For Any Hash Function

In this section we analyze the guesswork for any hash function, given the pattern of mappings from inputs to outputs. We begin by analyzing the average guesswork when bins are not allocated to the users, as presented in Definition 14, assuming that the attacker *has broken the hash function*, and for any bin knows of a password that is mapped to it. Then, we show that when the attacker does not know the hash function (i.e., any $P_H(b)$-hash function), and the least likely bins are allocated to the users according to the back door procedure presented in Definition 15, the average guesswork *over all strategies of cracking the hash function* is the same as the average guesswork of Theorem 10, followed by a concentration result regarding the probability that the number of guesses is bellow the average guesswork. In addition, we analyze the effect of *biased passwords* on the average guesswork. Finally, we find the average number of guesses required in order to find a password that is mapped to one of the allocated bins. Throughout this section we show that these results also apply to the $P_H(b)$-set of hash functions and any strategy of guessing passwords one by one, when averaging the guesswork over this set.

We now calculate the guesswork when bins are not allocated to the users, and for any bin the attacker knows of a password that is mapped to it.

**Theorem 12.** *When each user draws his password uniformly over $(1,\ldots,2^n)$ as in Definition 14, and for every bin the attacker knows of a password that is mapped to it, the $\rho$th moment of the guesswork of every user is*

$$E\left((G_H(B))^\rho\right) \geq \frac{1}{(1+m\cdot\ln(2))^\rho}\left(\sum_{b=1}^{2^m}(P_H(b))^{1/(1+\rho)}\right)^{1+\rho}. \tag{65}$$

*When $P_H(b)$ represents drawing $m$ bits i.i.d. Bernoulli$(p)$, the $\rho$th moment of the guesswork increases at rate that is equal to*

$$\lim_{m\to\infty}\frac{1}{m}\log\left(E\left((G_H(B))^\rho\right)\right) = H_{1/(1+\rho)}(p) \tag{66}$$

*where $H_{1/(1+\rho)}(p)$ is the Renyi entropy which is defined in equation* (2)*.*

*Proof:* We assume that for every bin the attacker knows of a password that is mapped to it. Therefore, the problem of guessing a password reduces to the problem of guessing the correct bin for each user. Since each user chooses the password uniformly over $\{1,\ldots,2^n\}$, the chance of him being mapped to bin $b\in\{1,\ldots,2^m\}$ is $P_H(b)$. Hence, the problem is reduced to the guesswork problem that is analyzed in [13]. We get the average guesswork that is stated above by following the same lines as [13] . ■

**Remark 22.** *Essentially, the result of Theorem 12 captures the average time required to crack a hash function when a* key stretching mechanism *[22] is used in order to protect a system against passwords cracking. For example, when $\rho = 2$, it means that the time that elapses between attempts increases quadratically.*

**Remark 23.** *When passwords are not drawn uniformly, the optimal strategy of guessing passwords one by one is to first calculate the probability of each bin by summing the probabilities over all passwords that are mapped to this bin, and then guessing bins according to their probabilities in descending order.*

We begin by defining a back door procedure for any hash function.

**Definition 15.** *The back door mechanism for modifying a hash function is defined as follows.*

- *For each user choose a different bin according to the procedure described in Definition 11 (i.e., the first user gets the least likely bin $b_1$, the second user receives the second least likely bin $b_2$, and the $2^{H(s)\cdot m-1}$th user receives $b_{2^{H(s)\cdot m-1}}$).*
- *Each of the users from the first to the $2^{H(s)\cdot m-1}$th draws a password uniformly by drawing $n$ bits i.i.d. Bernoulli$(1/2)$.*
- *For each user, if the number that was drawn $g\in\{1,\ldots,2^n\}$ has not been drawn by any of the users that have bins that are less likely than the current bin, then replace the mapping of $g$ with a mapping from $g$ to the bin assigned to the user.*
- *In the case when another user who is assigned to a less likely bin, has drawn $g$:* Do not change *the value of the mapping of $g$ again (i.e., the mapping from $g$ is changed only once by the user who is coupled with the least likely bin among the bins of the users whose password is $g$). Instead, change*

*the bin allocated to the user to the least likely bin among the bins allocated to users whose password is $g$. Therefore, the mapping from $g$ is changed only once, to the least likely bin that is mapped to $g$.*

We are now ready to prove Theorem 1.

*The proof of Theorem 1:* The underlying idea behind the proof is to lower bound the average guesswork by upper bounding the probability of finding a password that cracks the bin at each round; we upper bound the probability by incorporating the maximal number of mappings that the backdoor mechanism can add to each bin, as well as the number of unsuccessful guesses that have been made so far. We start by proving this theorem for any $P_H(b)$-hash function when averaged over all possible strategies of guessing passwords one by one, and we then show this also for the set of $P_H(b)$-hash functions and for any strategy. Then, in order to prove the equality rather than just an upper bound, we show that the probability that the number of mappings eliminated by the backdoor mechanism affects the fraction of mappings, is small to the extent that does not allow the average to grow beyond the lower bound discussed above.

Since the attacker does not know the mappings of the hash function, there is no reason for him to favor any guessing strategy over the other. Hence, we average over all strategies of guessing passwords one by one; essentially, this is equivalent to averaging over all possible permutations of the inputs, assuming that the permutations are uniformly distributed. This can be achieved by first drawing a random variable uniformly over $\{1,\ldots,2^n\}$, then drawing a random variable over the remaining $2^n-1$ possibilities, etc.

We now wish to lower bound the average guesswork by upper bounding $P_H(b)$. In order to lower bound the average guesswork we consider only the first $(1+\epsilon_1)\cdot m\cdot\log(1/p)$ guesses when averaging, where $0<\epsilon_1<\epsilon$. Clearly the number of guesses can be greater than the term above and may go up to $2^n$. Furthermore, we observe that after $k$ unsuccessful guesses, the chance of guessing bin $b$ is $P_H(b)\cdot\frac{2^n}{2^n-k}$ where $0\le k\le(1-P_H(b))\,2^n$. In addition, the back door procedure of Definition 15 can add one mapping at most to each bin. Thus, we can upper bound the probability by

$$P_H(b)\le\frac{P_H(b)\cdot 2^n+1}{2^n-k}=P_H^{(k)}(b)\qquad k\in\left\{0,\ldots,2^{(1+\epsilon_1)\cdot m\cdot\log(1/p)}\right\}. \tag{67}$$

Note that $\lim_{m\to\infty}\frac{2^{(1+\epsilon_1)\cdot m\cdot\log(1/p)}}{(1-P_H(b))2^n}=0$ since $\epsilon_1<\epsilon$.

The average guesswork of each bin can be lower bounded by

$$E(G(b))\ge P_H(b)\cdot\left(1+\sum_{i=1}^{2^{m_0}-1}(i+1)\cdot\prod_{j=1}^{i}\left(1-P_H^{(j)}(b)\right)\right)\ge P_H(b)\cdot\sum_{i=0}^{2^{m_0}-1}(i+1)\cdot\left(1-P_H^{(2^{m_0})}(b)\right)^i \tag{68}$$

where $m_0=(1+\epsilon_1)\cdot m\cdot\log(1/p)$; the first inequality is due to equation (67), whereas the second one is because of the fact that $P_H^{(2^{m_0})}(b)\ge P_H^{(i)}(b)$ for any $i\in\{1,\ldots,2^{m_0}-1\}$. Hence, we get

$$E(G(b))\ge\frac{P_H(b)}{P_H^{(2^{m_0})}(b)}\cdot\left(1/P_H^{(2^{m_0})}(b)-\left(1-P_H^{(2^{m_0})}(b)\right)^{2^{m_0}}\cdot\left(1/P_H^{(2^{m_0})}(b)+2^{m_0}\right)\right). \tag{69}$$

Now, let us break equation (69) down. First, we know that

$$P_H^{(2^{m_0})}(b)=\frac{P_H(b)\cdot 2^{(1+\epsilon)\cdot m\cdot\log(1/p)}+1}{2^{(1+\epsilon)\cdot m\cdot\log(1/p)}-2^{(1+\epsilon_1)\cdot m\cdot\log(1/p)}}=\frac{P_H(b)+2^{-(1+\epsilon)\cdot m\cdot\log(1/p)}}{1-2^{-(\epsilon-\epsilon_1)\cdot\log(1/p)\cdot m}} \tag{70}$$

In addition, since $P_H(b)\ge 2^{-m\cdot\log(1/p)}$ for any $b\in\{1,\ldots,2^m\}$ we get

$$P_H(b)\le P_H^{(2^{m_0})}(b)\le P_H(b)\cdot\frac{1+2^{-\epsilon\cdot\log(1/p)\cdot m}}{1-2^{-(\epsilon-\epsilon_1)\cdot\log(1/p)\cdot m}}\quad\forall b\in\{1,\ldots,2^m\}. \tag{71}$$

From Corollary 6 we get that

$$\lim_{m\to\infty}\left(1-P_H^{(2^{m_0})}(b)\right)^{2^{m_0}}\cdot\left(1/P_H^{(2^{m_0})}(b)+2^{m_0}\right)=0. \tag{72}$$

Further,

$$1/P_H^{(2^{m_0})}(b)\geq 1/P_H(b)\cdot\frac{1-2^{-(\epsilon-\epsilon_1)\cdot\log(1/p)\cdot m}}{1+2^{-\epsilon\cdot\log(1/p)\cdot m}}\quad\forall b\in\{1,\ldots,2^m\} \tag{73}$$

Hence, by averaging over all the types of the assigned bins similarly to what is done in Theorem 9 we get the inequality

$$\lim_{m\to\infty}\frac{1}{m}\log\left(E\left(G_T\left(B\right)\right)\right)\geq\left\{\begin{array}{cc} H(s)+D(s||p) & (1-p)\leq s\leq 1\\ 2\cdot H(p)+D(1-p||p)-H(s) & 1/2\leq s\leq(1-p)\end{array}\right.. \tag{74}$$

The equality is achieved based on the following arguments. Assume that the backdoor mechanism eliminates $l$ mappings to bin $b$. In order for the inequality in (74) to turn to equality, $l$ has to increase at a rate that is smaller than $2^n\cdot P_H(b)$. Therefore, since there are $2^{H(s)\cdot m-1}$ users, whenever $n\geq(1+\epsilon)\cdot(\log(1/p)+H(s))$, $l$ can not affect the rate at which $2^n\cdot P_H(b)$ increases.

Furthermore, this theorem also applies to the case when averaging over the $P_H(b)$-set of hash functions for any strategy of guessing passwords one by one (i.e., for any fixed strategy, the average is performed over the set of hash functions). This is due to the fact that any $P_H(b)$-hash function can be represented by a bipartite graph (e.g., Figure 4) for which averaging over the order according to which nodes that represent the domain, are chosen while keeping the edges fixed, leads to the same result as averaging over the order of the edges connected to these nodes while choosing the nodes in any order. The effect of the backdoor procedure given in Definition 15 is similar. When considering a user who is mapped to bin $b\in B$, the other users can only decrease the number of mappings to this bin. Based on the arguments presented in this proof, the other users can not increase the average guesswork. Furthermore, the user can add up to one mapping, which does not affect the average guesswork. ∎

Next, we prove Theorem 2.

*The proof of Theorem 2:* We are interested in upper bounding the probability

$$Pr\left(G(b)\leq 2^{(1-\epsilon_1)\cdot(H(s)+D(s||p))\cdot m}\right)=P_H^{(0)}(b)+\sum_{i=1}^{2^{(1-\epsilon_1)\cdot(H(s)+D(s||p))\cdot m}}P_H^{(i)}(b)\cdot\prod_{j=0}^{i-1}\left(1-P_H^{(j)}(b)\right) \tag{75}$$

where

$$P_H^{(i)}(b)=\frac{P_H(b)\cdot 2^n+1}{2^n-i}. \tag{76}$$

As $P_H(b)<P_H^{(0)}(b)<\cdots<P_H^{(2^{m_1})}(b)$ where $m_1=(1-\epsilon_1)\cdot(H(s)+D(s||p))\cdot m$, we can upper bound the expression in (75) by

$$Pr\left(G(b)\leq 2^{(1-\epsilon_1)\cdot(H(s)+D(s||p))\cdot m}\right)\leq\sum_{i=0}^{2^{m_1}-1}P_H^{(2^{m_1})}(b)\cdot(1-P_H(b))^i=\frac{P_H^{(2^{m_1})}}{P_H(b)}\cdot\left(1-(1-P_H(b))^{2^{m_1}}\right). \tag{77}$$

Following along the same lines as the proof of Theorem 1 we know that the ratio $\lim_{m\to\infty}\frac{P_H^{(2^{m_1})}}{P_H(b)}=1$, and because of the fact that $P_H(b)\leq 2^{-(H(s)+D(s||p))\cdot m}$ for any $b\in B$ we get

$$\lim_{m\to\infty}1-(1-P_H(b))^{2^{(1-\epsilon_1)\cdot(H(s)+D(s||p))\cdot m}}\leq\lim_{m\to\infty}1-\left(1-2^{-(H(s)+D(s||p))\cdot m}\right)^{2^{(1-\epsilon)\cdot(H(s)+D(s||p))\cdot m}}=0 \tag{78}$$

for any $b\in B$.

Note that the result above also applies to the case when

$$Pr\left(G\left(b\right)\leq 2^{(1-\epsilon_1)\cdot(H(q(b))+D(q(b)||p))\cdot m}\right)\leq \lim_{m\to\infty} 1-\left(1-P_H\left(b\right)\right)^{2^{(1-\epsilon_1)\cdot(H(q(b))+D(q(b)||p))\cdot m}}=0 \tag{79}$$

This result also applies to the case when averaging over the $P_H\left(b\right)$-set of hash functions and considering any strategy of guessing passwords one by one. It results from the same arguments that are given in Theorem 1; the backdoor mechanism can add no more than one mapping which is drawn uniformly over the set of passwords. Therefore, the same bounds and arguments hold for this case as well. This concludes the proof. ■

**Corollary 12.** *When the attacker does not know the mappings of the hash function, $P_H\left(b\right)$ represents drawing $m$ bits i.i.d. Bernoulli$(p)$, and $n\geq(1+\epsilon)\cdot m\cdot(\log(1/p)+H(s))$ where $\epsilon>0$; the average guesswork of bin $b\in\{1,\ldots,2^m\}$ when averaging according to Definition 10*

$$\lim_{m\to\infty}\frac{1}{m}\log\left(E\left(G_T\left(b\right)\right)\right)=\log\left(1/P_H\left(b\right)\right)=H\left(q\left(b\right)\right)+D\left(q\left(b\right)||p\right) \tag{80}$$

*where $q\left(b\right)$ is the type of bin $b$.*

*Proof:* The proof is straightforward based on the proof of Theorem 1. ■

Next, we prove Corollary 1.

*The proof of Corollary 1:* The proof of this corollary relies on the concentration property of the average guesswork of every bin presented in the proof of Theorem 2, along with the fact that the rate at which the average guesswork of a password increases is dominated by elements of type $\frac{\sqrt{\theta}}{\sqrt{\theta}+\sqrt{1-\theta}}$ [17].

We consider the case where $n\geq(1+\epsilon)\cdot m\cdot(\log(1/p)+H(s))$ as for this case the average guesswork of bin $b$ can not increase due to mappings that are removed by other users (for more details see the proof of Theorem 1).

We denote the average guesswork of bin $b\in B$ when averaged over the $P_H\left(b\right)$-set of hash functions by $G^{(H)}\left(b\right)$. From equation (73), along with the fact that $n\geq(1+\epsilon)\cdot m\cdot(\log(1/p)+H(s))$, we know that

$$\lim_{m\to\infty}\frac{1}{m}\log\left(E\left(G^{(H)}\left(b\right)\right)\right)=H\left(q\left(b\right)\right)+D\left(q\left(b\right)||p\right). \tag{81}$$

Furthermore, let us denote the average guesswork of a password that is drawn i.i.d. Bernoulli$(\theta)$ by $G^{(ps)}\left(\theta\right)$ whose rate is [13]

$$\lim_{n\to\infty}\frac{1}{n}E\left(G^{(ps)}\left(\theta\right)\right)=H_{1/2}\left(\theta\right). \tag{82}$$

The average guesswork of bin $b\in B$ is upper bounded by

$$\left(E\left(G\left(b\right)\right)\right)\leq\min\left(E\left(G^{(H)}\left(b\right)\right),E\left(G^{(ps)}\left(\theta\right)\right)\right) \tag{83}$$

because of the fact that the guesswork $G\left(b\right)=\min\left(G^{(H)}\left(b\right),G^{(ps)}\left(\theta\right)\right)$.

We begin be considering the case when $\lim_{m\to\infty}2\cdot\frac{n}{m}H\left(\frac{\sqrt{\theta}}{\sqrt{\theta}+\sqrt{1-\theta}}\right)\leq(1-\epsilon_2)\cdot(H\left(q\left(b\right)\right)+D\left(q\left(b\right)||p\right))$ for any $0<\epsilon_2<1$. We wish to show that in this case the rate at which the average guesswork increases is equal to $\lim_{m\to\infty}\frac{n}{m}H_{1/2}\left(\theta\right)$. First, note that

$$Pr\left(G\left(b\right)=i|G^{(H)}\left(b\right)\geq 2^{(1-\epsilon_2)\cdot(H(q(b))+D(q(b)||p))\cdot m}\right)=Pr\left(G^{(ps)}\left(\theta\right)=i\right) \tag{84}$$

when $i\in\left\{0,\ldots,2^{(1-\epsilon_2)\cdot(H(q(b))+D(q(b)||p))\cdot m}\right\}$ because $G^{(H)}\left(b\right)$ and $G^{(ps)}\left(\theta\right)$ are independent as well as $G\left(b\right)=\min\left(G^{(H)}\left(b\right),G^{(ps)}\left(\theta\right)\right)$. In [17] it was shown based on large deviation arguments that the rate at which the average guesswork of a password that is drawn i.i.d. Bernoulli$(\theta)$ increases is dominated by elements of type $\frac{\sqrt{\theta}}{\sqrt{\theta}+\sqrt{1-\theta}}$; and when guessing passwords according to their probabilities in descending order the number of guesses that include all elements of this type increases like $2^{2\cdot n\cdot H\left(\frac{\sqrt{\theta}}{\sqrt{\theta}+\sqrt{1-\theta}}\right)}$. Hence, based on this fact and (84) we get that the rate at which the conditional average guesswork increases is lower bounded by

$$\lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G\left(b|G^{(H)}(b) \geq 2^{(1-\epsilon_2)\cdot(H(q(b))+D(q(b)||p))\cdot m}\right)\right)\right) \geq \frac{n}{m} H_{1/2}(\theta) \tag{85}$$

when $\lim_{m\to\infty} 2 \cdot \frac{n}{m} H\left(\frac{\sqrt{\theta}}{\sqrt{\theta}+\sqrt{1-\theta}}\right) \leq (1-\epsilon_2)\cdot(H(q(b)) + D(q(b)||p))$. In addition, we know from equation (79) that

$$\lim_{m\to\infty} \frac{1}{m} \log\left(Pr\left(G^{(H)}(b) \geq 2^{(1-\epsilon_2)\cdot(H(q(b))+D(q(b)||p))\cdot m}\right)\right) = 0 \tag{86}$$

Therefore, we can lower bound the rate at which the average guesswork of bin $b$ increases by

$$\begin{aligned}\lim_{m\to\infty} \frac{1}{m} \log(E(G(b))) \geq \frac{1}{m} \log\left(E\left(G\left(b|G^{(H)}(b) \geq 2^{(1-\epsilon_2)\cdot(H(q(b))+D(q(b)||p))}\right)\right)\right) \times \\ Pr\left(G^{(H)}(b) \geq 2^{(1-\epsilon_2)\cdot(H(q(b))+D(q(b)||p))}\right) \geq \frac{n}{m} H_{1/2}(\theta).\end{aligned} \tag{87}$$

Hence, based on (83) and (87) we get

$$\lim_{m\to\infty} \frac{1}{m} \log(E(G(b))) = \frac{n}{m} H_{1/2}(\theta). \tag{88}$$

Next we wish to find the average guesswork when $H(q(b)) + D(q(b)||p) \leq (1-\epsilon_2)\cdot \frac{n}{m} H(\theta)$ for any $0 < \epsilon_2 < 1$. We wish to show that in this case the rate at which the average guesswork increases is equal to $H(q(b)) + D(q(b)||p)$. First, note that

$$Pr\left(G(b) = i|G^{(ps)}(\theta) > 2^{(1-\epsilon_2)\cdot(H(\theta))\cdot n}\right) = Pr\left(G^{(H)}(b) = i\right) \tag{89}$$

when $i \in \left\{0, \ldots, 2^{(1-\epsilon_2)\cdot(H(\theta))\cdot n}\right\}$ because $G^{(H)}(b)$ and $G^{(ps)}(\theta)$ are independent as well as $G(b) = \min\left(G^{(H)}(b), G^{(ps)}(\theta)\right)$. Based on equation (79) it can be easily shown that the average guesswork of the $P_H(b)$-set of hash functions is concentrated around its mean value. Hence, based on this fact and (89) we get that the rate at which the conditional average guesswork increases is lower bounded by

$$\lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G\left(b|G^{(ps)}(\theta) \geq 2^{(1-\epsilon_2)\cdot H(\theta)\cdot n}\right)\right)\right) \geq H(q(b)) + D(q(b)||p) \tag{90}$$

when $H(q(b)) + D(q(b)||p) \leq (1-\epsilon_2)\cdot \frac{n}{m} H(\theta)$. In addition

$$\lim_{n\to\infty} \frac{1}{n} \log\left(Pr\left(G^{(ps)}(\theta) \geq 2^{(1-\epsilon_2)\cdot H(\theta)\cdot n}\right)\right) = 0. \tag{91}$$

Therefore, we can lower bound the rate at which the average guesswork of bin $b$ increases by

$$\begin{aligned}\lim_{m\to\infty} \frac{1}{m} \log(E(G(b))) \geq \frac{1}{m} \log\left(E\left(G\left(b|G^{(ps)}(\theta) \geq 2^{(1-\epsilon_2)\cdot H(\theta)\cdot n}\right)\right)\right) \times \\ Pr\left(G^{(ps)}(\theta) \geq 2^{(1-\epsilon_2)\cdot H(\theta)\cdot n}\right) \geq H(q(b)) + D(q(b)||p).\end{aligned} \tag{92}$$

Hence, based on (83) and (92) we get

$$\lim_{m\to\infty} \frac{1}{m} \log(E(G(b))) = H(q(b)) + D(q(b)||p). \tag{93}$$

Equations (18) and (19) follow in a straightforward fashion when either $2^{n\cdot H\left(\frac{\sqrt{\theta}}{\sqrt{\theta}+\sqrt{1-\theta}}\right)}$ is smaller than the minimal average guesswork, that is, smaller than $2^{m\cdot(H(s)+D(s||p))}$; or when $\frac{n}{m}\cdot H(\theta)$ is larger than the maximal rate at which the average guesswork increases (i.e., larger than $\log(1/p)$). ■

We now prove Corollary 2.

*The proof of Corollary 2:* In order to calculate the average guesswork for guessing a password that is mapped to any bin that is allocated to any user, we first need to bound the probability $P_B$ which is the probability of guessing such a password in the first guess (i.e., the fraction of mappings to any of those bins). Since $P_B$ is the probability of guessing a password that is mapped to any of the assigned bins, we get

$$P_B = \sum_{b \in B} P_H\left(b\right) \tag{94}$$

where $B$ is the set of assigned bins as in Definition 11. Therefore, based on Lemma 1, Lemma 2, the fact that there are $m$ types, and the fact that the number of users $M = 2^{H(s)\cdot m-1} < 2^{H(s)\cdot m}$, as well as the fact that $\max_{1/2\leq s\leq q\leq 1} D\left(q||p\right) = D\left(s||p\right)$ when $p \leq 1/2$; $P_B$ can be bounded by

$$\frac{1}{\left(m+1\right)^2}\cdot 2^{-m\cdot D(s||p)} \leq P_B \leq m\cdot 2^{-m\cdot D(s||p)}. \tag{95}$$

Next, similarly to the proof of Theorem 1 equation (67) we can further bound the probability of success after $k$ unsuccessful guesses by

$$P_B \leq P_B^{(k)} \leq \frac{P_B\cdot 2^n + m\cdot 2^{m\cdot H(s)}}{2^n - k} \tag{96}$$

where $P_B^{(k)}$ is the probability of guessing a password that is mapped to any $b \in B$, after $k$ unsuccessful guesses; and $m\cdot 2^{m\cdot H(s)}$ takes into consideration the fact that the backdoor mechanism from Definition 15 may add a mapping to each bin. Similarly to (68) we can lower bound the guesswork by

$$E\left(G_{Any}\left(b\in B\right)\right) \geq P_B\cdot \sum_{i=0}^{2^{m_0}-1}\left(i+1\right)\cdot\left(1-P_B^{(2^{m_0})}\right)^i. \tag{97}$$

where $m_0 = \left(1+\epsilon_1\right)\cdot m\cdot\log\left(1/p\right)$, $0<\epsilon_1<\epsilon$. Since

$$P_B^{(2^{m_0})} \leq P_B\frac{1+\left(m+1\right)^2\cdot m\cdot 2^{-\epsilon\cdot\log(1/p)\cdot m}}{1-2^{-m\cdot\log(1/p)\cdot(\epsilon-\epsilon_1)}}. \tag{98}$$

Hence, for the exact same arguments as the ones in Theorem 1 we can state that

$$\lim_{m\to\infty}\frac{1}{m}\log\left(E\left(G_{Any}\left(b\in B\right)\right)\right) \geq \lim_{m\to\infty}\frac{1}{m}\log\left(1/P_B\right) = D\left(s||p\right). \tag{99}$$

The equality is achieved based on similar arguments to the ones provided in the proof of Theorem 1; the number of mappings that other users remove in the backdoor mechanism is negligible. Furthermore, these results also apply to the case when averaging over the $P_H\left(b\right)$-set of hash functions and considering any strategy of guessing passwords one by one. It results from the same arguments that are given in Theorem 1; each user in $B$ can add no more than one mapping, and the total number of mappings does not affect the average guesswork, as shown in equation (99).

■

**Remark 24.** *Note that the rate at which the average guesswork increases*

$$\begin{cases} H\left(s\right)+D\left(s||p\right) & \left(1-p\right)\leq s\leq 1 \\ 2\cdot H\left(p\right)+D\left(1-p||p\right)-H\left(s\right) & 1/2\leq s\leq\left(1-p\right) \end{cases} \tag{100}$$

*is an unbounded function that increases as $p$ decreases. However, as $p$ decreases, the minimal size of the input $n = \left(\log\left(1/p\right)+H\left(s\right)\right)\cdot m$ increases. Hence, the smaller $p$ is, the more asymptotic the result on the average guesswork becomes, in terms of the size of the input required to achieve this average.*

## VIII. Bounds on the Average Guesswork with no Bins Allocation

In order to upper bound the rate at which the average guesswork increases we consider the case where all users are mapped to the $2^{H(s)\cdot m-1}$ most likely bins. This in turn enables us to give an upper bound to the average guesswork by considering the results for the case of bins allocation. The lower bound is derived by assuming that the users are mapped to the $2^{H(s)\cdot m-1}$ most likely bins.

We begin by proving Theorem 5.

*proof of Theorem 5:* This result follows directly from equation (73) which lower bounds the probability when bins are allocated, by using the probability when there is no bins allocation. ■

Next, we prove Theorem 3

*proof of Theorem 3:* The upper bound follows directly from Corollary 2 as the average guesswork with bins allocation is equal to the result when considering the most likely bins.

The lower bound is based on the assumption that the users are mapped to the $2^{H(s)\cdot m-1}$ most likely bins. This leads to the following optimization problem

$$\min_{0\leq q\leq 1-s} D\left(q||p\right) \tag{101}$$

where $1/2 \leq s \leq 1$. The minimal value is achieved at $q = p \leq 1/2$, and is equal to $0$. Therefore, the lower bound is equal to

$$\begin{cases} D\left(1-s||p\right) & 0 \leq 1-s \leq p \\ 0 & p \leq 1-s \leq 1/2 \end{cases}. \tag{102}$$

Finally, because *there is no* backdoor procedure that modify the hash function, the results hold for $n \geq (1+\epsilon)\cdot m \cdot \log 1/p$. This concludes the proof. ■

We now prove Corollary 3.

*proof of Corollary 3:* We begin by proving that when $0 \leq 1-s < q \leq p$ the probability that all passwords are mapped to bins of type $q$ is $e^{-2^{m\cdot(2\cdot H(1-s)-H(q))}} \times 2^{-m\cdot D(q||p)\cdot 2^{H(1-s)\cdot m}}$. Since each password is drawn Bernoulli$(1/2)$ and $P_H\left(b\right)$ is i.i.d. Bernoulli$(p)$, the probability that a password is mapped to a certain bin of type $q$ is $2^{-m\cdot(H(q)+D(q||p))}$. Since there are $2^{m\cdot H(s)}$ users, the probability that all users are mapped to a certain set of bins in which all bins are of type $q$ is $2^{-m\cdot 2^{m\cdot H(1-s)}\cdot(H(q)+D(q||p))}$. The number of combinations in which the users are mapped to different bins of type $q$ is $\frac{2^{m\cdot H(q)}!}{\left(2^{m\cdot H(q)}-2^{m\cdot H(1-s)}\right)!}$. According to Sterling's approximation, when $m \gg 1$

$$2^{m\cdot H(q)}! \sim \frac{2^{m\cdot H(q)\cdot 2^{m\cdot H(q)}}}{e^{2^{m\cdot H(q)}}} \tag{103}$$

whereas for $m \gg 1$

$$\left(2^{m\cdot H(q)} - 2^{m\cdot H(1-s)}\right)! \sim 2^{m\cdot H(q)\cdot\left(2^{m\cdot H(q)}-2^{m\cdot H(1-s)}\right)} \times e^{2^{m\cdot(2\cdot H(1-s)-H(q))}-2^{m\cdot H(q)}}. \tag{104}$$

The ratio of the two equations above increases like

$$2^{m\cdot H(q)\cdot 2^{m\cdot H(1-s)}} \times e^{-2^{m\cdot(2\cdot H(1-s)-H(q))}}. \tag{105}$$

Therefore, the probability that all passwords are mapped to different bins of type $q$ is equal to

$$2^{-m\cdot 2^{m\cdot H(1-s)}\cdot(H(q)+D(q||p))} \times 2^{m\cdot H(q)\cdot 2^{m\cdot H(1-s)}} \times e^{-2^{m\cdot(2\cdot H(1-s)-H(q))}} = e^{-2^{m\cdot(2\cdot H(1-s)-H(q))}} \times 2^{-m\cdot D(q||p)\cdot 2^{H(1-s)\cdot m}}. \tag{106}$$

Furthermore the smallest rate at which the probability decreases is achieved when $q = p$ and is equal to

$$e^{-2^{m\cdot(2\cdot H(1-s)-H(p))}}. \tag{107}$$

Based on arguments similar to the ones in Theorem 3 we get that when all users are mapped to bins of type $q$ the probability of finding a password that is mapped to any of the bins is $2^{-m(H(q)+D(q||p)-H(1-s))}$

and therefore the average guesswork increases like $2^{m(H(q)+D(q||p)-H(1-s))}$. In the case when $q=p$ we get that the average guesswork increase like $2^{m(H(p)-H(1-s))}$

Finally, when $q<1-s\leq 1/2$ the most probable event is when all passwords are mapped to bins of type $p$ in which case the average guesswork does not increase exponentially, that is, the rate at which it increases is equal to zero. ∎

Next, we now prove Corollary 4.

*proof of Corollary 4:* The probability that the users are mapped to a particular type $q$ follows along the same lines as the proof of Corollary 3. When all users are mapped to bins of type $q$, the probability of guessing a password that is mapped to a bin of a user is the same across users and is equal to $2^{-m\cdot(H(q)+D(q||p))}$, and therefore the average guesswork is equal to $2^{m\cdot(H(q)+D(q||p))}$. ∎

Finally, we prove Theorem 4.

*proof of Theorem 4:* The upper bound follows directly from Theorem 1 as the average guesswork with bins allocation is equal to the result of averaging across the most likely bins. Because *there is no* backdoor procedure that modify the hash function, the results hold for $n\geq(1+\epsilon)\cdot m\cdot\log 1/p$.

The lower bound is based on the assumption that the users are mapped to the $2^{H(s)\cdot m-1}$ most likely bins. By arguments similar to Theorem 9 we arrive at the following optimization problem

$$\max_{0\leq q\leq 1-s} 2\cdot H\left(q\right)+D\left(q||p\right) \tag{108}$$

where $1/2\leq s\leq 1$. In lemma 4 it is shown that $2\cdot H\left(q\right)+D\left(q||p\right)$ is a concave function whose maximal value is at $q=1-p\geq 1/2$. Hence, when $0\leq q\leq 1-s\leq 1/2$ the maximal value is achieved for $q=1-s$ and is equal to $2\cdot H\left(1-s\right)+D\left(1-s||p\right)=2\cdot H\left(s\right)+D\left(1-s||p\right)$. This concludes the proof. ∎

## IX. Summary and Future Research

In this work we derive the average guesswork for cracking passwords for both keyed hash functions as well as regular ones. For the case where bins are allocated to the users, we find the rate at which the average guesswork increases, whereas in the case when the hash function can not be modified we bound this rate as well as find the most likely one.

When the hash function (or the key) can be modified (i.e., bins allocation), bias can in fact increase the average guesswork significantly. Furthermore, for strongly universal sets of hash functions it also leads to the fact that a smaller biased key achieves the same average guesswork as a larger unbiased key. In addition, there exists a backdoor mechanism that enables modifying any hash function efficiently without compromising the average guesswork. Finally, it turns out that when the hash function can not be modified, increasing the number of users has a far worse effect than bias on the average guesswork of hash functions.

## References


[1] S. Marechal, "Advances in password cracking," *Journal in Computer Virology*, vol. 4, no. 1, pp. 73–81, 2008.

[2] W. E. Burr, D. F. Dodson, and W. T. Polk, "Electronic authentication guideline," *NIST Special Publication 800-64*, 2006.

[3] U. Manber, "A simple scheme to make passwords based on one-way functions harder to crack," *Computers & Security*, vol. 15, no. 2, pp. 171–176, 1996.

[4] 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.

[5] 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.

[6] B. Schneier, "Cryptanalysis of md5 and sha: Time for a new standard," 2004.

[7] T. Cover and J. Thomas, *Elements of Information Theory*. John Wiley& Sons, 2006.

[8] T. Cormen, C. R. R. Leiserson, and C. STEIN, *Introduction to Algorithms*. Second Edition. MIT Press, Cambridge, MA., 2001.

[9] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk, "Cryptographic hash functions: A survey," *Department of Computer Science, Univ. Wollongong, Tech. Rep.*, pp. 1–26, 1995.

[10] M. Berlare, R. Canetti, and H. Krawczyk, *Keying Hash Functions for Message Authentication*. Springer Berlin Heidelberg, 1996, pp. 1–15.
[11] M. Wegman and J. Carter, “New hash functions and their use in authentication and set equality,” *Journal of Computer and System Sciences*, vol. 22, no. 3, pp. 265 – 279, June 1981.
[12] J. Massey, “Guessing and entropy,” in *ISIT 1994*, 1994.
[13] E. Arikan, “An inequality on guessing and its application to sequential decoding,” *IEEE Tran. on Inf. Th.*, vol. 42, 1996.
[14] E. Arikan and N. Merhav, “Guessing subject to distortion,” *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 1041–1056, May 1998.
[15] R. Sundaresan, “Guessing under source uncertainty,” *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 269–287, Jan 2007.
[16] ——, “Guessing under source uncertainty with side information,” in *Information Theory, 2006 IEEE International Symposium on*, July 2006, pp. 2438–2440.
[17] D. Malone and W. G. Sullivan, “Guesswork and entropy,” *IEEE Transactions on Information Theory*, vol. 50, no. 3, pp. 525–526, March 2004.
[18] M. M. Christiansen, K. R. Duffy, F. du Pin Calmon, and M. Medard, “Multi-user guesswork and brute force security,” *IEEE Transactions on Information Theory*, vol. 61, no. 12, pp. 6876–6886, Dec 2015.
[19] N. Merhav and E. Arikan, “The shannon cipher system with a guessing wiretapper,” *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 1860–1866, Sep 1999.
[20] C. Schieler and P. Cuff, “The henchman problem: Measuring secrecy by the minimum distortion in a list,” *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3436–3450, June 2016.
[21] K. Scarfone and M. Souppaya, “Guide to enterprise password management,” *Recommendations of the National Institute of Standards and Technology*.
[22] J. Kelsey, B. Schneier, C. Hall, and D. Wagner, *Secure applications of low entropy keys*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 121–134.