The Effect of Bias on the Guesswork of Hash Functions

Yair Yona
University of California, Los Angeles
Email: yairyo99@ucla.edu

Suhas Diggavi
University of California, Los Angeles
Email: suhas@ee.ucla.edu

***Abstract*—In this work we analyze the average guesswork for the problem of hashed password cracking (i.e., finding a password that has the same hash value as the actual password), when averaging over all hash functions whose effective distribution is i.i.d. Bernoulli(p) for any strategy of guessing passwords one by one (i.e., the fractions of passwords that are hashed to any bin, correspond to a probability mass function, which is i.i.d. Bernoulli(p)).**

We analyze the average guesswork under both online and offline attacks by deriving upper and lower bounds on the average guesswork as a function of the bins to which passwords are hashed, along with the most likely average guesswork, that is, the average guesswork of the most likely set of bins. Furthermore, we provide a *concentration* result that shows for this problem, that the probability mass function of guesswork is concentrated around its mean value.

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. However, when there exists a backdoor mechanism that enables "beamforming" certain passwords to the least likely bins, bias can in fact increase the average guesswork.

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 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]. Essentially, cryptographic hash functions are the "workhorses of modern cryptography" [4] and enable, among other uses, to securely protect passwords against offline attacks. Hashing has many applications in various fields such as compression [5], search problems [6], as well as cryptography [7]. When it comes to message authentication codes (MAC) [8] 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 a key, which is a secret [9]. A practical keyed hash function that enables one to securely use off-the-shelf computationally secured hash functions, is a keyed-hash message authentication code (HMAC) [8].

This work was supported in part by NSF grants 1321120 and 1423271.

The two main scenarios of password cracking are online guessing attack and offline attack [2]. In an 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 values to the same value as the original password. Offline attack occurs when an attacker gets a hold of the hashed values that are stored in the system; the attacker searches for a password that is hashed to any of the these values.

In this work we derive the average guesswork for password 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 [10] 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 [11], 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 [12], guessing under source uncertainty with and without side information [13], [14], using guesswork to lower bound the complexity of sequential decoding [11], guesswork for Markov chains [15], guesswork for multi-user systems [16] as well as guesswork for the Shannon cipher system [17].

In this work we consider a slightly different version of guesswork, as a hash functions is a mapping from a larger domain to a smaller range, and so an attacker needs to guess *a password* that has the same hash value as the original password and not necessarily *the password*.

We derive the average guesswork when either averaging over all hash functions whose effective distribution (i.e., the fraction of passwords that are hashed to any bin, which is the number of inputs that are hashed to a bin divided by the total

number of inputs) is i.i.d. Bernoulli(p) and for any strategy of guessing passwords one, or by averaging over all strategies of guessing passwords one by one and for any hash function that has effective distribution which is i.i.d. Bernoulli(p), when passwords are uniformly distributed.

We show that under an online attack, when the number of bins is 2^m and the number of users is $2^{H(s)\cdot m}$, where $1/2 \leq s \leq 1$, the rate at which the average guesswork increases (as a function of m) 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 is lower bounded by $H(s) + D(1-s||p)$, where $D(\cdot||\cdot)$ is the Kullback-Leibler divergence, and $H(\cdot)$ is the binary Shannon entropy [5]. Furthermore, we show that the most likely rate at which the average guesswork increases when passwords are drawn uniformly is $H(p)$. Finally, we prove that when averaging over all passwords (as well as over all hash functions or all possible strategies) the average guesswork scales like 2^m.

Under an offline attack, we derive lower and upper bounds for the average guesswork and show that the rate at which the average guesswork increases 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 $1/2 \leq s \leq 1-p$. We also find the most likely average guesswork when passwords are drawn uniformly and show that it increases at rate $H(p) - H(s)$.

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, 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.

Note that the upper bounds in both online and offline attacks are unbounded functions that increase as p decreases. These upper bounds are achieved when users choose passwords that are hashed to the least likely bins, in which case finding a password that is hashed to one of them is much harder than finding a password that is hashed to balanced bins (i.e., bins of a hash function that is not biased). In [18] it is shown that when there exists a back door mechanism that "beamformes" passwords to the least likely bins, these upper bounds can be achieved.

Finally, we provide a *concentration* result showing that the probability of drawing a strategy for which the 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 $H(s)+D(s||p)$ is smaller than $2 \cdot H(p) + D(1-p||p) - H(s)$ in the range $1/2 \leq s \leq (1-p)$.

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. Section V concludes the paper. In this paper we present sketch of proofs, the full proofs can be found in [18].

II. Background and Basic Definitions

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

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 $P_X(\cdot)$, 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 $G(X)$ (which is termed guesswork) that takes only positive integer values. The optimal strategy of guessing X that minimizes all non negative moments of $G(X)$ is guessing elements in X based on their probabilities in descending order [10], [11], such that $G(x) < G(x')$ implies $p_X(x) > p_X(x')$, i.e. a dictionary attack [19]; 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 the moments of guesswork are dictated by the Renyi entropy [11]. 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 $\alpha = 1/2$ [11]:

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

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

In this paper we derive guesswork for *hash functions*, where 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 inputs that are hashed to any bin; we use the following definition throughout the paper.

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

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

where $\mathbb{1}_b(x) = \begin{cases} 1 & x = b \\ 0 & x \neq b \end{cases}$*, and b represents any bin of $h(\cdot)$. Thus, $F_h(b)$ is the ratio between the number of inputs that are hashed to bin b, and the total number of inputs.*

Definition 2. *A* $P_h(b)$-hash function *is a hash function whose effective probability $F_h(b)$ as presented in Definition 1 equals*

the pobability mass function $P_h(b)$, where $b \in \{1,\ldots,2^m\}$ represents the bins of $h(\cdot)$.

Definition 3. *A $P_h(b)$-set of hash functions is all hash functions whose effective probability equals $P_h(b)$ as given in Definition 1, that is, these hash functions are identical up to a permutation of their inputs.*

III. Problem Setting

In this section we define the problem as well as attack models, and extend the definition of average guesswork to the case of cracking passwords.

The method according to which passwords are stored in the system is as follows: Users are 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. Furthermore, 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 that the system in turn hashes so that when the hashed value matches the bin stored in the system access is granted.

Definition 4 (An online Attack)**.** *The attack model:*

- *The attacker does not know does not which hash function is used. He may know its effective probability.*
- *The attacker neither knows the passwords of the users nor their hash values, which are stored in the system.*
- *The attacker guesses the passwords one by one according to whatever order he chooses.*
- *The attacker chooses a user name and then guesses passwords one by one. If the hash value of his guess does not match the hash value of the password, then he moves on to the next guess; once the hashed value of his guess is equal to the one of the password, he can access the system.*

Definition 5 (An offline attack)**.** *Assume that the attacker* knows the hashed values of the passwords of the users*, which are stored in the system; yet he does not know the passwords of the users and what hash function the system uses. Other than that, the attack model is identical to the one in Definition 4.*

Definition 6 (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 hashed to bin b (i.e., not necessarily* the password *of the user).*

Definition 7 (Averaging Arguments)**.** *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, whereas the guesswork of any $P_h(b)$-hash function is averaged over all possible strategies of guessing passwords one by one. The average is performed over a uniform distribution.*

Remark 1. *In this paper we assume that passwords are drawn uniformly. When passwords are biased there is a certain optimal strategy [11] 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 7. In this case 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 hashed to the same bin. More details appear in [18].*

Definition 8 (The average guesswork under an online attack)**.** *Assume that the number of users $M \leq 2^m$ and that the attacker draws uniformly a user out of the M users. The average guesswork across all users is*

$$E(G_T(B)) = \frac{1}{M}\sum_{b\in B} E(G(b)) \tag{3}$$

where $G(\cdot)$ is the guesswork as a function of the bin as given in Definition 6, the average is done according to Definition 7 assuming that the attacker only knows that passwords are drawn uniformly, and given that the passwords are hashed to the set of M bins B.

IV. Main Results

In this section we derive the average guesswork when the attacker knows that passwords are drawn uniformly, that is, each password consists of n bits that are drawn i.i.d. Bernoulli$(1/2)$. We average according to Definition 7 for both online and offline attacks, and derive upper and lower bound assuming that the passwords are hashed to a certain set of bins 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 hashed 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 assuming that none of the pairs of passwords is hashed to the same bin. Furthermore, 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 section 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.

A. The average guesswork under an online attack

We analyze the guesswork based on the method of types [5]. A bin b of m bits is of type $q(b)$ in case $N(1|\underline{b})/m = q(b)$ where $N(1|\underline{b})$ is the number of occurrences of the number 1 in the binary vector b.

Lemma 1 (The average guesswork of every bin)**.** *When the attacker does not know which hash function hashes the passwords, $P_h(b)$ represents drawing m bits i.i.d. Bernoulli(p),*

and $n \geq (1+\epsilon) \cdot m \cdot (\log(1/p))$ *where* $\epsilon > 0$*, the average guesswork of bin* $b \in \{1, \ldots, 2^m\}$ *increases at rate*

$$\lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G_T\left(b\right)\right)\right) = \lim_{m\to\infty} \frac{1}{m} \log\left(1/P_h\left(b\right)\right) = H\left(q\left(b\right)\right) + D\left(q\left(b\right)||p\right). \quad (4)$$

Sketch of proof. The full proof appears in [18] Corollary 12. The general idea is bounding the probability of guesswing a password that is hashed to bin b by geometric distribution. □

Theorem 1 (Bounds on the average guesswork under an online attack)**.** *When the attacker does not know which hash function hashes the passwords, there are* $2^{H(s)\cdot m-1}$ *users whose passwords are hashed to different bins, where,* $1/2 \leq s \leq 1$*,* $P_h\left(b\right)$ *represents drawing* m *bits i.i.d. Bernoulli*(p)*, and* $n \geq (1+\epsilon) \cdot m \cdot \log(1/p)$ *where* $\epsilon > 0$*, the rate of the average guesswork is bounded by*

$$H\left(s\right) + D\left(1-s||p\right) \leq \lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G_T\left(B\right)\right)\right) \leq \begin{cases} H\left(s\right) + D\left(s||p\right) & (1-p) \leq s \leq 1 \\ 2 \cdot H\left(p\right) + D\left(1-p||p\right) - H\left(s\right) & 1/2 \leq s \leq (1-p) \end{cases} \quad (5)$$

Proof. The full proof appears in [18] Theorem 4. It is achieved by analyzing the concave function $2 \cdot H\left(q\right) + D\left(q||p\right)$. □

Corollary 1 (The most likely average guesswork under online attacks)**.** *When the attacker does not which hash function hashes the passwords, there are* $2^{H(s)\cdot m}$ *users,* $1/2 \leq s \leq 1$*,* $P_h\left(b\right)$ *represents drawing* m *bits i.i.d. Bernoulli*(p)*,* $n \geq (1+\epsilon) \cdot m \cdot \log(1/p)$*, and passwords are drawn i.i.d. Bernoulli*$(1/2)$*, we get the following results:*

- *The probability that all passwords are hashed to different bins of type* q *such that* $0 \leq 1-s < q \leq p \leq 1/2$*, 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* $2^{m\cdot H(p)}$.

Proof. The full proof appears in [18] Corollary 4. □

Corollary 2 (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 $G_D\left(B\right)$ *is the guesswork of any user, and* $n \geq (1+\epsilon) \cdot m \cdot \log(1/p)$.

Proof. When passwords are drawn uniformly, the probability of drawing a password that is hashed to bin b, whose guesswork increases like $1/P_h\left(b\right)$, is $P_h\left(b\right)$. Summing over all bins, we obtain the result above. □

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

B. The average guesswork under an offline attack

Theorem 2 (Bounds on the average guesswork under an offline attack)**.** *When the attacker does not know which hash function hashes the passwords, there are* $2^{H(s)\cdot m-1}$ *bins to which the passwords are hashed,* $1/2 \leq s \leq 1$*,* $P_h\left(b\right)$ *represents drawing* m *bits i.i.d. Bernoulli*(p)*, and* $n \geq (1+\epsilon) \cdot m \cdot \log(1/p)$*, where* $\epsilon > 0$*, the rate of the average number of guesses required to find a password that is hashed to any of the bins to which the passwords are hashed, is bounded by*

$$\begin{cases} D\left(1-s||p\right) & 0 \leq 1-s \leq p \\ 0 & p \leq 1-s \leq 1/2 \end{cases} \leq \lim_{m\to\infty} \frac{1}{m} \log\left(E\left(G_{Any}\left(B\right)\right)\right) \leq D\left(s||p\right). \quad (6)$$

Proof. The full proof appears in [18] Theorem 3. □

Remark 3. *Note that the rate at which the upper bounds on the average guesswork of Theorem 1 and Theorem 2 (i.e., under online and offline attacks) are unbounded functions that increase as* p *decreases. However, as* p *decreases, the minimal size of the input* $n = \log(1/p) \cdot m$ *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. Moreover, these upper bounds are achieved when users choose passwords that are hashed to the least likely bins, in which case finding a password that is hashed to one of these bins is far harder than in the case when the hash function is not biased.*

Corollary 3 (The most likely average guesswork under offline attacks)**.** *When the attacker does not know which hash function hashes the passwords, there are* $2^{H(s)\cdot m}$ *users,* $1/2 \leq s \leq 1$*,* $P_h\left(b\right)$ *represents drawing* m *bits i.i.d. Bernoulli*(p)*, and* $n \geq (1+\epsilon) \cdot m \cdot \log(1/p)$*, where* $\epsilon > 0$*, the average number of guesses required to find a password that is hashed to any of the bins to which the passwords are hashed, can be characterized as follows.*

- *The probability that all passwords are hashed 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 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.

Proof. The full proof appears in [18] Corollary 3. □

Remark 4 (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*

Table I
THE RATE AT WHICH THE MOST LIKELY AVERAGE GUESSWORK INCREASES ALONG WITH THE LOWER AND UPPER BOUNDS UNDER AN OFFLINE ATTACK. 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 4. NOTE THAT 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

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 2 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

$$\log_2(p) - \log_2(1-p) \tag{7}$$

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.

C. A Concentration Result

Theorem 3 (Concentration result under both online and offline attacks)**.** *When the attacker does not know which hash function hashes the passwords, there are $2^{H(s)\cdot m}$ bins to which the passwords are hashed, $1/2 \leq s \leq 1$, $P_h(b)$ represents drawing m bits i.i.d. Bernoulli(p), and $n \geq (1+\epsilon)\cdot m \cdot \log(1/p)$ the following concentration result holds for any of the bins to which the passwords are hashed*

$$-\lim_{m\to\infty}\frac{1}{m}\log\left(P\left(G(b) \leq 2^{(1-\epsilon_1)\cdot(H(q(b))+D(q(b)||p))\cdot m}\right)\right) \geq \epsilon_1 \cdot \log(1/p), \tag{8}$$

where $0 < \epsilon_1 < 1$ and bin b is of type $q(b)$. This concentration result applies to the average guesswork of Theorem 2 as well.

Proof. The proof is in [18] Theorem 5. □

Remark 5. *From Theorem 3 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 6. *The concentration result of Theorem 3 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 [11] 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 [15].*

Remark 7. *In this work we extend the case of balanced hash functions to the case when the hash functions are i.i.d. Bernoulli(p) (e.g., from $p = 1/2$ to $p < 1/2$). This in turn allows us to provide closed form expressions for the average guesswork across users. However, the results of Lemma 1 do not require the i.i.d. assumption, and so the results of this work can be extended to more elaborate scenarios.*

V. SUMMARY

In this work we quantify the effect of bias on the guesswork of hash functions and show that increasing the number of users has a far worse effect than bias on guesswork.

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

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

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

[7] 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.

[8] M. Berlare, R. Canetti, and H. Krawczyk, *Keying Hash Functions for Message Authentication*. Springer Berlin Heidelberg, 1996, pp. 1–15.

[9] 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.

[10] J. Massey, "Guessing and entropy," in *ISIT 1994*, 1994.

[11] E. Arikan, "An inequality on guessing and its application to sequential decoding," *IEEE Tran. on Inf. Th.*, vol. 42, 1996.

[12] E. Arikan and N. Merhav, "Guessing subject to distortion," *IEEE Tran. on Inf. Th.*, vol. 44, no. 3, pp. 1041–1056, May 1998.

[13] R. Sundaresan, "Guessing under source uncertainty," *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 269–287, Jan 2007.

[14] ——, "Guessing under source uncertainty with side information," in *Inf. Th., 2006 IEEE Int. Symp. on*, July 2006, pp. 2438–2440.

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

[16] 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.

[17] 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.

[18] Y. Yona and S. Diggavi, "Password cracking: The effect of bias on the average guesswork of hash functions," *Available at https://arxiv.org/abs/1608.02132*.

[19] K. Scarfone and M. Souppaya, "Guide to enterprise password management," *Rec. of the Nat.l Inst. of St. and Tech.*