The Password Allocation Problem

Strategies for Reusing Passwords Effectively

Rishab Nithyanand
Department of Computer Science
Stony Brook University
New York, USA
rnithyanand@cs.stonybrook.edu

Rob Johnson
Department of Computer Science
Stony Brook University
New York, USA
rob@cs.stonybrook.edu

ABSTRACT

Each Internet user has, on average, 25 password-protected accounts, but only 6.5 distinct passwords [4]. Despite the advice of security experts, users are obviously re-using passwords across multiple sites. So this paper asks the question: given that users are going to re-use passwords across multiple sites, how should they best allocate those passwords to sites so as to minimize their losses from accidental password disclosures?

We provide both theoretical and practical results. First, we provide a mathematical formulation of the Password Allocation (*PA*) problem and show that it is NP-complete with a reduction via the 3-Partition problem. We then study several special cases and show that the optimal solution is often a contiguous allocation – i.e., similar accounts share passwords. Next, we evaluate several human- and machine-computable heuristics that have very good performance and produce solutions that are reasonably close to optimal. We find that the human-computable heuristics do not perform nearly as well as the machine-computable heuristics, however, they provide a useful and easy to follow set of guidelines for re-using passwords.

Categories and Subject Descriptors

F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; H.1.2 [**Models and Principles**]: User/Machine Systems—*Human Factors*

General Terms

Algorithms, Security, Human Factors

Keywords

Password re-use; Heuristics; Authentication; Usability

WPES'13, November 4, 2013, Berlin, Germany.

ACM 978-1-4503-2485-4/13/11 ...$15.00.
http://dx.doi.org/10.1145/2517840.2517870 .

1. INTRODUCTION

Current guidelines for password re-use rarely go beyond: *don't do it* [7]. However, the problem of how to allocate and reuse passwords deserves more practical solutions due to the rapid proliferation of password protected web services in recent years and the facts that (1) human memory rarely allows for the commitment of more than six to eight *base* passwords (i.e., not including minor variations which are easily guessed) [4] and (2) evidence strongly suggests that user chosen pseudonyms are easily linkable for a majority of users – therefore the compromise of a single account due to password disclosure leads to the effective compromise of all accounts that share that password [9], [8].

Our work focuses on developing strategies for password re-use for *no-fault* users. These are users that do not accidentally disclose their passwords to attackers by themselves – i.e., they are phish-proof, key-logger free, etc. Instead, our focus is on password leaks caused by theft of password files stored by service providers. Recently, such thefts have impacted millions of users of services provided by organizations such as Canonical, FBI, Hotmail, IEEE, LinkedIn, Sony, Ubisoft, Yahoo!, and many others. Often, due to an organizations poor security practices such as using plaintext password files (examples from the previous list are the FBI, IEEE, Sony, and Yahoo! [2]), even having high entropy passwords and following password guidelines does not protect its users in the event of a server breakin.

In this paper we approach the problem of password allocation and reuse from both, a theoretical and a practical setting. Towards the understanding of the theoretical aspects of the problem, we provide a mathematical formulation of the Password Allocation (*PA*) problem and the problem is shown to be NP-complete in section 2. In section 3, we study several special cases and show that the optimal solution often has the property of contiguous allocations – i.e., similar accounts share passwords. From the practical standpoint, in sections 4 and 5, we study several human which perform reasonably well and machine computable heuristics that produce near optimal solutions. The performance of these heuristics yields a preliminary set of guidelines for re-using passwords in section 7.

1.1 Problem Definition

Informally, the *Password Allocation (PA)* problem may be described as follows: If a user has n web accounts and k passwords and the users i^{th} password is used for the j^{th} account, then the probability, q_{ij}, that an attacker obtains the i^{th} password by breaking into the j^{th} web service is a func-

tion of (1) the security of the j^{th} service and (2) the strength of the i^{th} password (if service j hashes passwords). Thus, given estimates of the value (v_j) of the j^{th} account (based on the perceived value or sensitivity of contained data), the security of each service provider (based on historical data and published policies), and strength of each password (based on entropy), how should a user allocate passwords to accounts so as to minimize their expected loss (or, equivalently – maximize their expected surviving value) from server break-ins?

The PA problem is mathematically formulated as:

PROBLEM 1. *PA Problem: Given $v_1, \ldots, v_n$ and p_{ij} for $i = 1, \ldots k$ and $j = 1 \ldots, n$, find partition $S_1 \cup \cdots \cup S_k$ of $\{1, \ldots, n\}$ that maximizes $ES = \sum_{i=1}^{k} (\sum_{j \in S_i} v_j \prod_{j \in S_i} p_{ij})$.*

Here, $p_{ij} = 1 - q_{ij}$, S_i denotes the set of accounts allocated to the i^{th} password, and ES denotes the expected surviving value of all accounts after password allocation is complete. We will also use the notation $V_i = \sum_{j \in S_i} v_j$ and $P_i = \prod_{j \in S_i} p_{ij}$, so $ES = \sum_{i=1}^{k} ES_i = \sum_{i=1}^{k} P_i V_i$. Accounts that provide the means to obtain access to other *linked accounts* that do not share the same password (eg., email accounts that may be used to recover other account passwords) are referred to as *gateway accounts*. Our formulation allows the modeling of gateway accounts by having the values associated with gateway accounts set to be the sum of the values of the corresponding linked accounts. While the compromise probability of the gateway account does not need to be changed, the compromise probability of each account should incorporate the compromise probabilities of each of their gateway accounts.

The mathematical formulation of the password allocation problem requires users to assign values to the data contained in each service/account. We emphasize that there should be no standard guidelines for assigning values to the data (other than requiring users to maintain consistency between accounts). As long as users are aware of (1) the data made available in each service and (2) how the data may be used by an adversary, the absence of a standard procedure in assigning values to services is desirable since it permits users to better protect accounts that they *perceive* to be more important – e.g., a company spokesperson might find that their social network accounts are generally more valuable (and important to protect) than their shopping accounts, while this might not be the case for other users. Note that compromise probabilities are easily assigned using publicly available data about password file type (i.e., hashed vs. unhashed), recent password file compromises, and estimated number of current users. Such information can be found using public breach/vulnerability databases (e.g., [2]).

2. COMPLEXITY

PROBLEM 2. *Decisional PA Problem (D-PA): Given the instance of a PA problem and a threshold r, does there exist an allocation such that $ES \geq r$?*

THEOREM 1. *D-PA problems are* NP-Complete.

PROOF. Membership in NP is easily established. Given a guess for $S_1, \ldots, S_k$, we just compute ES and verify that it is at least r. This can be done in time proportional to the length of the words representing the values and probabilities.

We will reduce the well studied 3-Partition problem (3P) [5], to the D-PA problem. Given an instance $I_{3P} : X = \{x_1, \ldots, x_{3n}\}$ (where $\sum_{j=1}^{3n} x_j = nB$) of the $3P$ problem, we create an instance I_{PA} of the D-PA problem with $3n$ accounts and n passwords as follows: (1) Pick a rational number $b \in (1, e^{4/nB})$, (2) Set $v_j = x_j$, $\forall j \in \{1, \ldots, 3n\}$, (3) Set $p_{ij} = b^{-x_j}$, $\forall j \in \{1, \ldots, 3n\}$ and $\forall i \in \{1, \ldots, n\}$, and (4) Set $r = n \times B \times b^{-B}$.

By construction, $\sum_{i=1}^{n} V_i = nB$ and, for any allocation $S_1, \ldots, S_n$, we have $ES = \sum_{i=1}^{n} V_i b^{-V_i}$. We now argue that, because of the choice of the base, b, $ES \leq \sum_{i=1}^{n} B b^{-B} = r$. Consider any unequal allocation, i.e. in which there exist i and i' such that $V_i \neq V_{i'}$. We will show that the allocation would be improved by redistributing the value equally between passwords i and i'. Consider the function $f(x) = x b^{-x} + (c - x) b^{-(c-x)}$, where $c = V_i + V_{i'} \leq nB$. First, observe that $f'(c/2) = 0$. Second, since $b < e^{4/c}$, $f'(x) > 0$ for all $x \in [0, c/2)$ and $f'(x) < 0$ for $x \in (c/2, c]$. Hence $x = c/2$ is a global maximum. Thus, if an allocation has any i, i' such that $V_i \neq V_{i'}$, we could improve it by replacing V_i and $V_{i'}$ by $(V_i + V_{i'})/2$. Hence the optimal allocation must have $ES \leq \sum_{i=1}^{n} B b^{-B} = r$, and this can only be obtained if $V_1 = \cdots = V_n = B$.

Therefore, if there is a solution to the 3P problem, then there exists an allocation for the D-PA problem in which $V_1 = \cdots = V_n = B$ and $ES = r$. If, on the other hand, there is an allocation of the D-PA problem such that $ES \geq r$, then we must have $ES = r$ and $V_1 = \cdots = V_n = B$, and there is a solution to the 3P problem. □

Theorem 1 shows that even the restricted D-PA problem – where probabilities are account dependent but password independent (for eg., when all passwords are equally secure, or all websites use unhashed password files) – is weakly NP-Complete. It remains an open question whether the D-PA problem is strongly NP-Complete.

3. SPECIAL CASES: WHEN OPTIMAL ALLOCATIONS ARE CONTIGUOUS

3.1 Identical Risk Accounts

Here, each account possesses a distinct value and compromise probabilities that are password dependent (but, account independent) – i.e., we have $p_{ij} = p_i$ ($\forall i$, $\forall j$). This special case is applicable to the PA problem when dealing with organizations that are known to use hashed password files and are (more or less) equally vulnerable to attacker breakins. The compromise probabilities are dependent only on the strength of the passwords allocated to each account. The problem can be stated as: Given $p_{ij} = p_i$ and v_j for $i = 1, \ldots k$ and $j = 1 \ldots, n$, find partition $S_1 \cup \cdots \cup S_k$ of $\{1, \ldots, n\}$ that maximizes $ES = \sum_{i=1}^{k} (\sum_{j \in S_i} v_j p_i^{|S_i|})$.

THEOREM 2. *If accounts and passwords are ordered by decreasing values (v_j) and survival probabilities (p_i), respectively, the allocation of accounts to passwords is contiguous.*

PROOF. Let $v_1 \geq v_2 \geq \cdots \geq v_n$. Consider the allocation given by $S = \langle S_1, \ldots, S_k \rangle$ where without loss of generality, $p_1^{|S_1|} \geq p_2^{|S_2|} \geq \cdots \geq p_k^{|S_k|}$. Let S_i be the first password with a non-contiguous allocation – i.e., account $l \in S_i$ but account $m \in S_{i+1}$ (where $v_l \leq v_m$). Now, consider the allocation given by $S^* = \langle S_1, \ldots, S_i^*, S_{i+1}^*, \ldots, S_k \rangle$ where account $l \in S_{i+1}^*$ and account $m \in S_i^*$ – i.e., the allocation where the passwords of accounts l and m are swapped.

Now, observe that: $[ES_i + ES_{i+1}] - [ES_i^* + ES_{i+1}^*] = \left[p_i^{|S_i|}(v_m - v_l) - p_{i+1}^{|S_{i+1}|}(v_l - v_m)\right] \leq 0$.

Therefore, the expected survival value of a contiguous allocation is always at-least as good as any non-contiguous allocation. □

Since the optimal allocation of accounts to passwords is contiguous (given accounts sorted by their values), the following recursive relation may be used to find the optimal allocation:

$OPT(i,j) = \max_{1 \leq l \leq i} \left[p_j^{i-l} \sum_{m=l}^{i} v_m + OPT(l, j-1)\right]$.

Here, $OPT(i,0) = \infty$ and $OPT(n,k)$ returns the maximum expected survival value for n accounts allocated to k passwords in $O(n^2k)$ time.

3.2 Identical Passwords and Identical Valued Accounts

In this scenario we have that all accounts are (nearly) equally valuable and have compromise probabilities that are account dependent (but, password independent) – i.e., we have (*w.l.o.g*) $p_{ij} = p_j$ and $v_j = 1$ $(\forall i, \forall j)$. For example, when allocating passwords to equally valuable accounts where all passwords have the same entropy, or when password files are unhashed by all service providers. The problem can be stated as: Given $p_{ij} = p_j$ and $v_j = 1$ for $i = 1, \ldots, k$ and $j = 1, \ldots, n$, find partition $S_1 \cup \cdots \cup S_k$ of $\{1, \ldots, n\}$ that maximizes $ES = \sum_{i=1}^{k}(|S_i| \prod_{j \in S_i} p_j)$.

THEOREM 3. *If accounts are ordered by decreasing survival probabilities (p_j), each password is allocated a contiguous subset of accounts.*

PROOF. Without loss of generality, let $p_1 \geq p_2 \geq \cdots \geq p_n$. Consider the allocation given by $S = \langle S_1, \ldots, S_k \rangle$ where $\forall i \in \{1, \ldots, k-1\}$ we have $ES_i \geq ES_{i+1}$. Let S_i be the first password with a non-contiguous allocation – i.e., account $l \in S_i$ but account $m \in S_{i+1}$ (where $p_l \leq p_m$). Now, consider the allocation given by $S^* = \langle S_1, \ldots, S_i^*, S_{i+1}^*, \ldots, S_k \rangle$ where account $l \in S_{i+1}^*$ and account $m \in S_i^*$ – i.e., the allocation where the passwords for accounts l and m are swapped.

Now, observe that: $[ES_i + ES_{i+1}] - [ES_i^* + ES_{i+1}^*] = [ES_i + ES_{i+1}] - \left[\frac{p_m}{p_l} ES_i + \frac{p_l}{p_m} ES_{i+1}\right] \leq [ES_i + ES_{i+1}] - \left[ES_i + ES_{i+1}(\frac{p_l}{p_m} + \frac{p_m}{p_l} - 1)\right] \leq 0$.

Therefore, the expected survival value of a contiguous allocation is always at-least as good as any non-contiguous allocation. □

Since the optimal allocation of accounts to passwords is contiguous (given accounts sorted by their survival probabilities), the following recursive relation may be used to find the optimal allocation.

$OPT(i,j) = \max_{1 \leq l \leq i} \left[(i-l) \prod_{m=l}^{i} p_m + OPT(l, j-1)\right]$

Here, $OPT(i,0) = \infty$ and $OPT(n,k)$ returns the maximum expected survival value for n accounts allocated to k passwords in $O(n^2k)$ time.

3.3 Identical Passwords and Atleast Exponentially Varying Accounts

Here, each account possesses a distinct value and compromise probabilities that are account dependent (but, password independent) – i.e., we have $p_{ij} = p_j$ $(\forall i, \forall j)$. For example, when allocating passwords to accounts with a large range of values where all passwords have the same entropy, or when password files are unhashed by all service providers. Observe that when we make no assumptions about the distributions of the values and probabilities, the problem is NP-Complete as shown in Theorem 1. However, the problem becomes instantly solvable in the case where we have $p_j \leq (\prod_{l=1}^{j-1} p_l)$ and $v_j \geq (\sum_{l=j+1}^{n} v_l)$, $\forall j \in \{1, \ldots, n\}$, as illustrated by theorem 4.

THEOREM 4. *If accounts are ordered by decreasing values (v_j), then allocation of accounts to passwords is contiguous.*

PROOF. Let $v_1 \geq v_2 \geq \cdots \geq v_n$. Consider the allocation given by $S = \langle S_1, \ldots, S_k \rangle$ where without loss of generality, $ES_1 \geq ES_2 \geq \cdots \geq ES_k$. Let S_i be the first password with a non-contiguous allocation – i.e., account $l \in S_i$ but account $m \in S_{i+1}$ (where $v_l < v_m$). Now, consider the allocation given by $S^* = \langle S_1, \ldots, S_i^*, S_{i+1}^*, \ldots, S_k \rangle$ where account $l \in S_{i+1}^*$ and account $m \in S_i^*$ – i.e., the allocation where the passwords of accounts l and m are swapped. Let $P_i = \frac{1}{p_l} \prod_{a \in S_i} p_a$, $P_{i+1} = \frac{1}{p_m} \prod_{a \in S_{i+1}} p_a$ and $V_i = v_l + \sum_{a \in S_i} v_a$, $V_{i+1} = v_m + \sum_{a \in S_{i+1}} v_a$.

Now, observe that: $[ES_i + ES_{i+1}] - [ES_i^* + ES_{i+1}^*] = -[P_iV_i(p_m - p_l) + P_i(v_m p_m - v_l p_l) + P_{i+1}V_{i+1}(p_l - p_m) + P_{i+1}(p_l v_l - p_m v_m)] \leq 0$.

Therefore, the expected survival value of a contiguous allocation is always at-least as good as any non-contiguous allocation. □

Since the optimal allocation of accounts to passwords is contiguous (given accounts sorted by their values), the following recursive relation may be used to find the optimal allocation: $OPT(i,j) = \max_{1 \leq l \leq i} [\prod_{m=l}^{i} p_m \sum_{m=l}^{i} v_m + OPT(l, j-1)]$. Here, $OPT(i,0) = \infty$ and $OPT(n,k)$ returns the maximum expected survival value for n accounts allocated to k passwords in $O(n^2k)$ time.

4. HUMAN-COMPUTABLE HEURISTICS

From section 3 and other analysis, it is observed that in many cases, some contiguous allocation of accounts (sorted by value) to passwords results in optimal (or, near-optimal) solutions to the PA problem. Based on this observation, we provide three simple human computable heuristics for allocating passwords to accounts. A performance comparison of the heuristics is provided in Table 1. PA problem instances for the results shown in Table 1 were randomly generated.

4.1 The k-Drops Heuristic

This heuristic is based on the idea that similarly valued accounts should be allocated to the same password. In the k-Drops heuristic, accounts are ordered by decreasing values and the decrease in value between each successive account is computed. Let the accounts with the $k-1$ largest drops in value be denoted by $\delta_1, \ldots, \delta_{k-1}$ where $v_{\delta_1} \geq v_{\delta_2} \geq \cdots \geq v_{\delta_{k-1}}$. Now the accounts are partitioned into k subsets $(A_1, \ldots, A_k)$ as follows: accounts with values in the range $[v_1, v_{\delta_1}]$ are placed in subset A_1, ..., accounts with values in the range $(v_{\delta_{k-1}}, v_n]$ are placed in the subset A_k. Now, accounts in subset A_l are allocated to the password for which their cumulative compromise probability is minimum – i.e., to password i where $i \leftarrow \arg\min \{\prod_{j \in A_l} p_{1j}, \ldots, \prod_{j \in A_l} p_{kj}\}$ and password i has not been allocated previously.

n	k	k-Drops		Bounded Range		NES	
		μ	σ	μ	σ	μ	σ
25	5	.13	.070	.053	.011	.213	.071
	10	.397	.101	.083	.028	.567	.068
50	10	.140	.052	.020	.008	.250	.044
	25	.562	.056	.104	.021	.693	.025
100	25	.238	.039	.039	.006	.419	.032
	50	.635	.044	.142	.022	.764	.015
250	50	.208	.031	.019	.003	.408	.016
	100	.561	.038	.084	.009	.738	.008

Table 1: Mean (μ) and Std Devn (σ) of the ratio between human computable heuristic ES and upper-bound ES for varying n and k (50 trials each).

4.2 The Bounded Range Heuristic

The general idea behind this heuristic is to bound the range of values allocated to each password. As before, we order accounts by decreasing values. We then compute δ such that $\delta^k \geq v_0 - v_n$. Now the accounts are partitioned into k subsets $(A_1, \ldots, A_k)$ as follows: accounts with values in the range $[v_n, v_n + \delta^1]$ are placed in subset A_1, ..., accounts with values in the range $(v_n + \delta^{k-1}, v_1]$ are placed in the k^{th} subset A_k. Now, accounts in the subset A_l are allocated to the password for which their cumulative compromise probability is minimum – i.e., to password i where $i \leftarrow \arg\min \{\prod_{j \in A_l} p_{1j}, \ldots, \prod_{j \in A_l} p_{kj}\}$ and password i has not been allocated previously.

4.3 The Near Even Split (NES) Heuristic

The general idea behind the NES heuristic is to allocate a near equal value to each password. In the NES heuristic, accounts are ordered by decreasing values and the running sum of values is computed. Let $T \leftarrow \lceil \sum_{j=1}^{n} v_j/k \rceil$ and $T_i \leftarrow \sum_{j=i}^{n} v_j$. Now the accounts are partitioned into k subsets $(A_1, \ldots, A_k)$ as follows: accounts which have T_i in the range $(T * (j-1), T * j]$ are placed in the j^{th} subset A_i. Similar to the k-Drops heuristic, accounts in subset A_l are allocated to the password for which their cumulative compromise probability is minimum – i.e., to password i where $i \leftarrow \arg\min \{\prod_{j \in A_l} p_{1j}, \ldots, \prod_{j \in A_l} p_{kj}\}$ and password i has not been allocated previously. Based on the impressive performance of the NES heuristic, one may be tempted to claim that optimal solutions must have nearly equal values allocated to each password (regardless of contiguity), however, simple counter-examples may be made by manipulating the compromise probabilities of certain account-password pairs.

5. MACHINE COMPUTABLE HEURISTICS

The heuristics presented here are based on greedy and dynamic programming approaches which are unlikely to be usable as human computable heuristics. A performance comparison of the heuristics is provided in Table 2. PA problem instances for the results shown in Table 2 were randomly generated.

5.1 The Ordered Maximum Marginal Return

The Maximum Marginal Return algorithm is a $O(nk)$ greedy approach which makes allocations of accounts to the passwords that result in the largest increase (or, smallest decrease) in the value of the objective function (i.e., the ES). The accounts are allocated in order of decreasing value using the algorithm illustrated in algorithm 1. Our experimental analysis revealed that allocating in decreasing order of value performed better (on average) than when accounts were unordered, or ordered by increasing values.

Algorithm 1 The MMR and CMMR Algorithms

```
function MMR(n, k, v_1, ..., v_n, p_11, ..., p_kn, cmmr)        ▷
cmmr ← 1 if running Clairvoyent MMR
    for i = 1 → k do
        S_i ← ∅, ES_i ← 0, S_dumpster ← ∅
    end for
    for i = 1 → n do
        for j = 1 → k do
            T_j ← S_j ⋃ i, δ_j ← COMPUTE-ES(T_j) − ES_j
        end for
        m ← arg max{δ_1, ..., δ_k}
        if cmmr = 1 ∧ δ_m < 0 then
            S_dumpster ← S_dumpster ⋃ i
        else
            S_m ← S_m ⋃ i, ES_m ← COMPUTE-ES(S_m)
        end if
    end for
    if cmmr = 1 then
        for j = 1 → k do
            T_j ← S_j ⋃ S_dumpster, δ_j ← COMPUTE-ES(T_j) − ES_j
        end for
        m ← arg max{δ_1, ..., δ_k}
    end if
    return ⟨S_1, ..., S_k⟩
end function
```

5.2 Clairvoyant MMR

The following variation is made to the Ordered-MMR algorithm: If there is no password to which the current account can be allocated without causing a drop in the cumulative expected survival value, then that account is placed in a *dumpster*. After initial allocation of all accounts is complete, all accounts in the dumpster are allocated together (as a single account) to the one password that experiences the smallest drop in ES.

5.3 Dynamic Programming Based Heuristic

The DPH algorithm runs in $O(n^2 k)$ time and is loosely based on a dynamic programming approach with state space trimming [10]. Consider allocating the accounts to passwords one at a time, i.e., we allocate the first account, then the second, etc. Let S_{ti} be the set of accounts allocated to the i^{th} password at time step t, $V_{ti} = \sum_{j \in S_{ti}} v_j$, and $P_{ti} = \prod_{j \in S_{ti}} p_{ij}$, $ES_{ti} = P_{ti} V_{ti}$, and $ES_t = \sum_{i=1}^{k} ES_{ti}$. If we allocate the t+1st account to the ℓth password, then we will have $P_{t+1,i} = P_{t,i} p_{\ell,t+1}$ if $i = \ell$ and $P_{t+1,i} = P_{t,i}$ otherwise. Similarly, $V_{t+1,i} = V_{t,i} + v_{\ell,t+1}$ if $i = \ell$ and $V_{t+1,i} = V_{t,i}$ otherwise. Thus $(V_{t1}, \ldots, V_{tk}, P_{t1}, \ldots, P_{tk})$ is the only state information we need to compute the state after allocating the t'th account. This gives a dominance relation among allocations: if $V_{ti} \leq V'_{ti}$ and $P_{ti} \leq P'_{ti}$ for all i, then every extension of allocation $(V_{t1}, \ldots, V_{tk}, P_{t1}, \ldots, P_{tk})$ will have lower ES than the corresponding extension of $(V'_{t1}, \ldots, V'_{tk}, P'_{t1}, \ldots, P'_{tk})$. Thus we only need to consider $(V'_{t1}, \ldots, V'_{tk}, P'_{t1}, \ldots, P'_{tk})$ in our search for the optimal allocation. Now, we reduce the size of the state space from exponentially to polynomially large by collapsing similar states into one. To do this, we

n	k	O-MMR		C-MMR		DP-H	
		μ	σ	μ	σ	μ	σ
25	5	.491	.054	.518	.076	.498	.092
	10	.799	.024	.807	.041	.799	.043
50	10	.594	.027	.561	.02	.594	.041
	25	.924	.019	.930	.011	.927	.024
100	25	.862	.029	.851	.031	.862	.021
	50	.956	.027	.956	.006	.951	.009
250	50	.898	.013	.871	.019	.901	.028
	100	.973	.003	.973	.004	.973	.002

Table 2: Mean (μ) and Std Devn (σ) of the ratio between machine computable heuristic ES and upper-bound ES for varying n and k (50 trials each).

divide the state space into n uniformly sized blocks ($\sqrt{n}$ regions for the Ps and Vs, respectively) and maintain exactly n states for each iteration of the dynamic program – i.e., the state with the highest ES for each block. Therefore, in each iteration, no more than n *promising* states are maintained while the remaining are culled. Finally, after all n accounts are allocated, the state with the maximum ES is returned. The algorithm is illustrated in algorithm 2.

Algorithm 2 Dynamic Program Based Heuristic

function DOH-SOLVE($v_1, \ldots, v_n, p_{11}, \ldots, p_{kn}$)
  $\Phi_0 \leftarrow \{(0, \ldots, 0, 1, \ldots, 1)\}$
  for $j = 1 \rightarrow n$ **do**
    $\Phi_j \leftarrow \emptyset$
    for all $(V_1, \ldots, V_k, P_1, \ldots, P_k) \in \Phi_{j-1}$ **do**
      for all $i = 1 \rightarrow k$ **do**

$$P'_\ell = \begin{cases} P_i p_{ij} & \text{if } i = \ell \\ P_\ell & \text{if } i \neq \ell \end{cases}$$

$$V'_\ell = \begin{cases} V_i + v_{ij} & \text{if } i = \ell \\ V_\ell & \text{if } i \neq \ell \end{cases}$$

        $\Phi_j \leftarrow \Phi_j \cup \{(V'_1, \ldots, V'_k, P'_1, \ldots, P'_k)\}$
      end for
    end for
    $\Phi_j \leftarrow$ TRIM(Φ_j) ▷ Divide state-space into n blocks and store the state with highest ES in each block.
  end for
  return $\max_{D \in \Phi_n} ES(D)$
end function

6. DISCUSSION: FUTURE RESEARCH

It is clear that the question of how to re-use passwords deserves significantly more attention and practical solutions than it has so far received from the academic community. Besides providing a mathematical framework for the password allocation problem, the focus of this paper has been to identify strategies to minimize the expected *user perceived* value of data lost by sharing passwords between accounts and the methods suggested in sections 4 and 5 appear to be reasonably successful at achieving this. However, there still remains a number of avenues for future research that *need* to be pursued for the password allocation problem to be properly understood. Below, we identify several directions of future research and some important questions that need to be answered.

- **User Password Re-use Behavior.** Current user studies and empirical analysis [4], [6], [3], [1] have shown that users are clearly sharing passwords between sites. However, they shed little light on the following questions: (1) Are users aware of the risks of sharing passwords? (2) Do users share passwords arbitrarily? Or, do they follow certain self-formulated guidelines (if yes, how are these formulated)?

  Finding answers to these questions could reveal insights into whether the problem for end-user security is a lack of password re-use risk awareness (that may be solved by simple methods – eg., browsers that notify users of the risks of re-using passwords at every sign-up page), or simply the lack of standard guidelines, suggesting that users would be receptive to *usable* guidelines for password re-use.

- **Usability of Re-use Guidelines and Heuristics.** Since any proposed guidelines and heuristics (including those presented in this paper) require active user participation, its usability will be one of the key factors influencing its potential acceptance.

  Therefore, usability studies need to be conducted for any proposed guidelines and heuristics in order to answer the following key questions: (1) Are users willing to undertake periodic (or, one-time) overheads in order to minimize their expected losses? (2) Are users able to reasonably follow the guidelines without significant errors – i.e., errors that cause a significant increase in their expected loss? (3) How does the usability of re-use guidelines compare with the usability of password managers – in both, the single device and multi-device scenario?

7. CONCLUSIONS

In this paper we studied the problem of how to effectively re-use passwords from a theoretical and practical standpoint. Although the general problem is shown to be NP-complete, several relevant and efficiently solvable special cases were identified. In addition, human and machine computable heuristics were evaluated and found to perform reasonably well. Based on the special cases identified and the performance of the human computable heuristics, we provide the first set of practical and effective guidelines for effective on-the-fly password allocations and re-use when the algorithms described in section 5 are inapplicable.

1. When dealing with accounts that have similar security infrastructure/policies and are known to use hashed password files, maintain a contiguous allocation. In this case, the k-Drops heuristic is found to be the best performing human computable heuristic if the strength of passwords has high deviation from the mean. Otherwise, the NES heuristic performs best.

2. When dealing with equally valuable accounts (eg., social networks) and similar strength passwords or accounts with unhashed password files, the optimal allocation is contiguous in terms of probabilities. In this case, a variant of the k-Drops heuristic (where allocations are made based on the largest drops in running product of probabilities) is found to work best.

3. When dealing with accounts that are highly varying (eg., exponentially increasing values/probabilities), the same rules as case (1) apply.

4. For allocating passwords to new accounts on the fly (when generating a new password is not an option), inserting the account into the allocation generated by the NES is recommended as long as periodic re-partition of accounts-to-passwords is performed.

5. For any other case not mentioned above, the NES appears yield best average performance.

6. The most effective method to improve heuristic performance is to reduce the n/k ratio.

8. REFERENCES

[1] Reused login credentials: Security advisory. Trusteer, Inc., February 2010.

[2] Office of inadeqaute security. `http://www.databreaches.net/category/breach-types/exposure/`, 2013.

[3] J. Bonneau. Measuring password re-use empirically. Light Blue Touchpaper, February 2011.

[4] D. Florencio and C. Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, 2007.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[6] S. Gaw and E. W. Felten. Password management strategies for online accounts. In *Proceedings of the second symposium on Usable privacy and security*, SOUPS '06, pages 44–55, New York, NY, USA, 2006. ACM.

[7] A. Huth, M. Orlando, and L. Pesante. Password security, protection, and management. *United States Computer Emergency Readiness Team*, October 2012.

[8] B. Ives, K. R. Walsh, and H. Schneider. The domino effect of password reuse. *Communications of the ACM*, 47(4):75–78, Apr. 2004.

[9] D. Perito, C. Castelluccia, M. Kaafar, and P. Manils. How unique and traceable are usernames? In *Proceedings of Privacy Enhancing Technologies*. 2011.

[10] G. J. Woeginger. When does a dynamic programming formulation guarantee the existence of an fptas? In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, 1999.