"Give Me Letters 2, 3 and 6!": Partial Password Implementations & Attacks*

David Aspinall[1] and Mike Just[2]

[1] University of Edinburgh, david.aspinall@ed.ac.uk
[2] Glasgow Caledonian University, mike.just@gcu.ac.uk

Abstract. A *partial password* is a query of a subset of characters from a full password, posed as a challenge such as *"Give me letters 2, 3 and 6 from your password"*. Partial passwords are commonly used in the consumer financial sector, both online and in telephone banking. They provide a cheap way of providing a varying challenge that prevents eavesdroppers or intermediate systems learning a shared secret in a single step. Yet, despite widespread adoption among millions of consumers, this mechanism has had little attention in the academic literature. Answers to obvious questions are not clear, for example, how many observations are needed for an attacker to learn the complete password, or to successfully answer the next challenge? In this paper we survey a number of online banking implementations of partial passwords, and investigate the security of the mechanism. In particular, we look at *guessing attacks* with a *projection dictionary* ranked by likelihood, and *recording attacks* which use previous information collected by an attacker. The combination of these techniques yields the best attack on partial passwords.

1 Introduction

A *partial password* is a query of a subset of characters from a full password. The mechanism is widely adopted in the financial sector; it is particularly popular in consumer online banking in the UK [1]. As far as we can tell, the idea spread from telephone banking, where it was invented to prevent an operator seeing a customer's complete password. It is now widely used in online banking web applications and by some instances of the 3DSecure system employed by Visa and Mastercard. Online, the main benefit of the mechanism is that it is a cheap way to impede attackers who can observe password entry by shoulder surfing, key logging or browser malware. This is because it allows a time-varying challenge that does not reveal complete information in a single step.

Despite widespread adoption among millions of consumers, this mechanism seems to have received only cursory attention in the academic literature so far. There are some obvious questions that remain unanswered, such as: what level

* Accepted as a full paper to appear in *Proceedings of the 17th International Conference on Financial Cryptography & Data Security*, Springer LNCS, Apr. 2013 (http://fc13.ifca.ai/). The final publication will be available at springerlink.com

of security is provided by this scheme? How many observations does an attacker need to learn the complete password or the correct response to the next challenge? Is it safe to use weak passwords? What are good choices for parameters such as the challenge size or the schedule of challenges issued? Is there already an industry consensus on preferred range of parameters?

To address these questions, we studied the security of the protocol design and also examined (externally, as users) 15 different implementations across 4 countries. Most of the implementations we examined were UK-based; we believe the protocol is more commonly used in the UK than other countries at present.

Security is assessed by measuring the difficulty of attacks. Password attacks may occur *online* or *offline*. The difficulty of offline attacks, where a password database is stolen, will depend on exactly what is stored in the database. To support the partial protocol the implementation will need to either store plaintext for the password, or devise a mechanism for performing one-way checks on all combinations that might be queried (which can be a large number for long passwords). We don't investigate this attack mode here.

Our focus is on online attacks. These may be either *targeted* against a particular user, or *trawling*, working through as many accounts as possible to break a percentage of them. Trawling takes statistical advantage of large numbers of users to circumvent the rate-limiting that applies on individual accounts. There are two basic techniques the attacker can use: he can *guess* responses using background information (e.g., a dictionary) or he can *record* previous observations of the protocol and try to replay them appropriately. These techniques can be combined: recording can be used to reduce the guesses needed at each stage. In either case, the goal is to break into accounts by either responding correctly to the *next challenge* issued in the protocol, or learning the *complete password* to answer all future challenges.

Contributions and highlights. To the best of our knowledge, this is the first time a detailed study of the partial password protocol has been given. Our main contributions are: (1) a survey of current implementations of partial passwords in online banking; (2) a framework for analysing and comparing these in terms of parameter choices; (3) investigation of a family of attacks based on guessing and recording, and an analysis of their success rates.

A short partial challenge is obviously less secure than a challenge to produce the full password, even guessing randomly without background information. To see what happens with background data, we measure guessing against subsets of the widely analysed RockYou leaked password set, coalescing entries for each challenge to form what we call a *projection dictionary*. We find that with 6 guesses, an attacker can respond correctly to 2-place challenges on 6-digit PINs with a success rate of 30%. Recording up to 4 runs, an attacker can succeed over 60% of the time, or by combining guessing and recording, over 90%. Alphanumeric passwords do somewhat better: responding to 3-place challenges on 8-character alphanumeric passwords, with up to 10 guesses, the attacker can achieve a success rate of 5.5%. Combining guessing and recording increases that to 25% with one recorded run and at least 80% with four runs.

Related work. Human identification methods attempt to mitigate against an observer by using time-varying challenges to prevent replay, and by concealing the shared secret by requiring the user to do some manipulations [2,3]. These schemes are reminiscent of partial passwords but require the user to do much more work, going well beyond what would be comfortable for most mainstream users, to achieve probabilistic security guarantees [4]. The actual mechanisms in use by industry have not been widely examined in the open literature. One paper that has examined it is the study of Goring et al [5], which considered a particular attack enabled by hardware key-loggers that build up information about responses but not challenges; our recording attacks consider more powerful attackers who can see both challenge and response. This incremental learning about a secret has similarity to PIN guessing through faulty APIs in which an attacker is able to query the API to gradually learn different PIN positions [6,7] although in the case of partial passwords, active attacks are not required. Instead, an observer can use partial information, either to answer the next challenge directly, or in combination with information about likely password distributions to make very good guesses. To show this, we adapt recent work on guessing attacks [8,9] and analysing large leaked password sets [10,11,12] to partial passwords.

Overview. The remainder of the paper is organised as follows. In Section 2 we survey 15 implementations of partial passwords, to find the parameters they use. Broadly, these split into PINs and alphanumeric passwords; we choose two typical cases for further investigation. In Section 3 we consider guessing attacks, building up the capability of the attacker: we go from random guessing, through attacks that use letter frequencies and finally to dictionaries, in particular, a *projection dictionary* attack that guesses responses based upon projected characters from likely password dictionaries. In Section 4 we analyse recording attacks, where an adversary has the capability to observe previous runs. We show how the information learned can be used to respond exactly to a fraction of next challenges and how quickly the whole password is eventually (almost certainly) discovered. We then combine recording with guessing to achieve our optimal attacks. Section 5 concludes and discusses the different attacks as well as some recommendations.

2 System model and survey

Fig. 1 shows some partial password challenges from real implementations. These show two different choices in the format for presenting the challenge, and for filling in the response. The overall protocol works as follows:

Registration In the initialisation phase, the user chooses a password. Often, this is a short (e.g., 6–10 characters) password taken from a restricted alphabet (e.g., lower case letters and numbers only), and without rules enforcing special password formats, so weak (but memorable) passwords are allowed. In several implementations, short PINs are used.

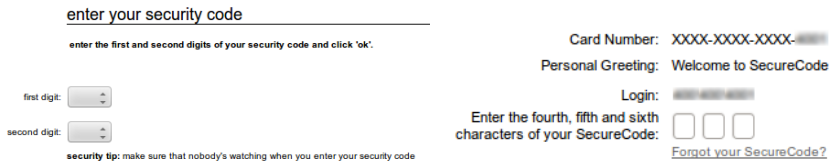

Fig. 1. Partial password implementations

Login The user is presented with a challenge such as: *Please provide letters 2, 3 and 6 from your password.* He or she responds by projecting out the requested positions.
Here is an explicit example:

Positions:	1	2	3	4	5	6	7
User password:	a	s	h	u	f	1	0
Correct response:		s	h			1	

From our anecdotal discussions, some users perform the projection mentally, some count off letters using their fingers; others write the password down as above. If the correct response is given, the user has completed partial password authentication.

Retry If the response is not correct, the user is challenged again, perhaps using the same challenge or perhaps changing the challenge. We observed both cases, and rate limits being enforced to restrict the number of attempts before re-registration is required.

Next time On the next login, the user is presented with a new challenge.

This process achieves the basic requirement of not revealing the password to an observer in a single step. In almost all cases we've seen, the partial password is used in addition to another credential (e.g., a full password) in a *multi-stage authentication*, where both credentials must be entered correctly to proceed.

2.1 Survey

We conducted a survey of online web banking implementations in four countries: Canada, Germany, Ireland, and the UK. We investigated the credentials they use for authentication (password, PINs, hardware token, etc.), and the way they check those credentials (partial queries, sequentially or together, etc.). We used several collection methods, including publicly available demos and web pages (see Appendix A.1) and direct access with personal accounts. Most of the implementations were in the UK, as we were able to generate accounts there to try them out; moreover we believe that the UK is the biggest user of this form of authentication at present. The data we collected is necessarily a snapshot; banking implementations are updated periodically.

For partial password and partial PIN checks, there are several parameters which vary between implementations, summarised in Fig. 2. The parameters of interest are:

- The **password format**. This constrains the set of allowed passwords. We suppose that the password consists of n characters chosen from an N character set. For partial passwords, this set is often deliberately small, in particular typically not including non-alphanumeric symbols and not distinguishing upper and lower case letters. This may be for better usability, e.g., by reducing the size of pull-down menus for implementations that use them.
- The **challenge format**. The challenge asks for characters in m unique password positions, where $1 \leq m \leq n$. In all but one case we surveyed, the positions are queried in ascending order. This reduces the number of possible challenges but, we suppose, it may be easier for the user to step through the letters in order from left to right rather than hop around.
- The **number of guesses**. If the user responds incorrectly to a challenge, they may be allowed more attempts; we will use β to stand for the maximum number of tries allowed before the account is locked in some manner. This is also the number of guesses an online attacker may make.

We gathered values for β but they are not shown in the table as they are difficult to corroborate without owning an account with the bank in question, and even then can vary in unexpected ways from our "black box" view. Smaller limits give better security, but inconvenience the user: it has been suggested that ten attempts at password entry should be allowed to provide good usability [13]. It seems reasonable that this limit would also be applied to partial passwords, and we found some implementations do use β=10. For PINs there may be concerns about allowing too many guesses because of the smaller answer space; implementations of protocols behind ATMs typically use β=6 as an upper limit [9].

In Fig. 2 we have listed the second credential used by each bank, whether a full password, full PIN, response to a challenge question, or multi-purpose information such as a credit card number and date of birth. In several cases, the banks also offer alternatives to their customers (e.g., tokens and card readers), though the option to not use these remains. The failure behaviours differed considerably between implementations, e.g., what kind of feedback was provided when the user made a mistake, and whether failure required re-entry of other credentials. These choices affect the overall security of the system so that, for example, for a bank that uses a password as a second credential, a guessing attack would need to compromise both the full and the partial password. Some work has been undertaken to investigate the overall security in this case [14], but in this paper we focus on the security of the partial password stage.

There is another system parameter not listed in the table: the system's schedule of issued challenges. Without knowledge of the system implementation, we can only view the challenge schedule externally and make assumptions about the sequence. A plausible working assumption that fits with our observations is that challenges are chosen uniformly at random from the $\binom{n}{m}$ possible challenges.

The survey motivates two typical parameter cases which we use for empirical study: PINs with N=10, n=6, m=2 and β=6 ($\binom{n}{m} = 15$) and alphanumeric passwords with N=36, n=8, m=3 and β=10 ($\binom{n}{m} = 56$).

Bank (UK based unless noted)	character set size, N	password length, n	challenge size, m	second credential
ING DiBa (Germany)	10	6	2	PIN
Cooperative	10	4	2	question
Tesco	10	6	2	password
Smile	10	6	2	question
Nationwide	10	6	3	password
AIB	10	5	3	question
Bank of Ireland (Ireland)	10	6	3	date of birth
Nat West, step 1	10	4	2	(see next row)
Nat West, step 2	36	6–20	3	(see prev row)
HBoS	36	6–15	3	password
3DSecure, B. of Ireland (UK)	36	8–15	3	credit card #
Standard Life	36	8–10	3	none
Skipton	36	8–30	3	question
First Direct	36	6–30	3	question
Barclays	52	6–8	2	PIN
HSBC (Canada)	62	8	3	question

Fig. 2. Survey of partial password parameters (as of 25 Sept 2012)

3 Guessing attacks

In a guessing attack, the attacker attempts to answer the next challenge, or find the whole password, by selecting from a set of possible answers, possibly using background data such as a dictionary. The plain *brute force* attack uses only knowledge of the response alphabet. Assuming a uniform distribution, the probability of guessing a randomly chosen user's complete password is $\frac{1}{N^n}$, which is 10^{-6} for the PIN case and approximately 10^{-12} for the alphanumeric. The probability of guessing the next challenge is $\frac{1}{N^m}$, so $\frac{1}{100}$ for PINs and approximately 2×10^{-5} for alphanumeric; already a reduced baseline.

The attacker usually can make more than one guess, however. The *β-success rate* [8,15] is a useful measure of the effectiveness of online attacks: it describes the (maximal) proportion of a data set that can be covered by a fixed number β of guesses. A *trawling attack* repeats guessing against many accounts; assuming these are selected randomly from the same distribution without replacement, we expect to break a fraction of them given by the success rate.

For a uniformly distributed password set we would expect to break the proportion $\frac{\beta}{N^n}$ of accounts by guessing whole passwords, or $\frac{\beta}{N^m}$ guessing partial challenges. A system security designer wants to keep the β-success acceptably low. With brute force attacks, the β-success rates goes up to 1 in 5000 for guessing the next challenge in the alphanumeric case (β=10) and 6% for PINs (β=6).

Guessing non-uniformly can change things dramatically. Real passwords are often chosen from common names, dictionary words and their variations; certain PINs are also very common, chosen from dates, or simple numeric or keypad-layout sequences. Dictionary attacks exploit this; their success depends on their

coverage. For offline (or unbounded online) attacks, enormous cracking dictionaries are pre-calculated and stored [16] or generated on-the-fly from lists of words and rules for making variations by mutation, or by approximate methods [17]. For online attacks where rate-limiting applies, the attacker benefits from word ranking, so he can guess the most likely (i.e., most commonly chosen) password first, then the second most likely, and so on, up to the rate limit.

3.1 Letter position frequency attack

We can improve on brute force by using letter frequencies to choose guesses. Compared with storing or generating a large dictionary, attack code with letter frequencies can be small and efficient, thus easily run in many places or on small devices. The attack could simply use overall frequency of letters, but the partial password challenges give us position information too. So we can exploit calculated letter-position frequencies to skew choices when guessing letters.

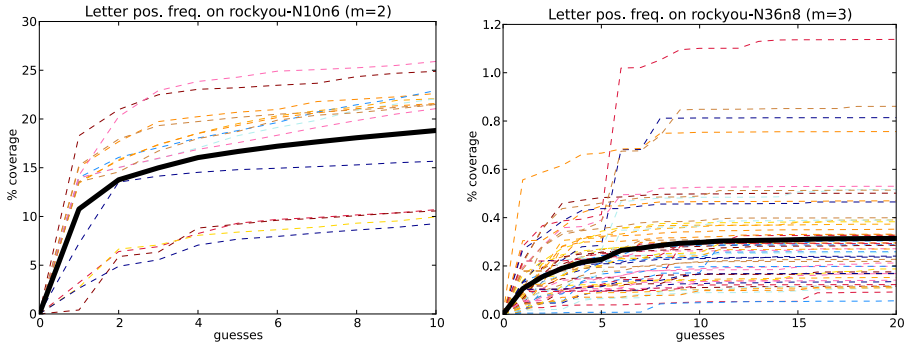

Fig. 3. Position frequency attack for RockYou PINs and passwords

It's easy to generate tables of relative letter frequencies from a dictionary. For example, in RockYou (a large set of leaked passwords described further below in Sect. 3.3), while the letter 'a' occurs 8% of the time in 8-character alphanumeric passwords, in position 2 it occurs 18% of the time. Similarly, for 6 digit PINs taken from RockYou, the digit 1 occurs on average 17% of the time, but almost 40% in position 1. (see Appendix A.2 for pictures that illustrate these numbers.) To attack using these relative frequencies we can guess characters randomly but in proportion to their frequency of occurrence by position, or, more simplistically, in a fixed order guessing the most common letters in each position in turn. For example, in the RockYou case above, the first guess for position 2 in any challenge will always be 'a'. The ideal success rate of this latter strategy is illustrated by Fig. 3, where we measure the attack against the same sets used to generate the frequencies, and show the proportion that are broken for increasing numbers of guesses β. Each dotted line on the plot indicates success rates for a different

challenge; the bold line indicates the success rates averaged over all challenges. For PINs, after 6 guesses we get an average success rate of 17% whereas for passwords, we get an average of 0.3% after 10 guesses. Because we assume that challenges are issued uniformly at random for different accounts, we take these average figures as the overall success rates.

This attack is obviously poor because it ignores correlations between letter positions; if we choose position 2 to be 'a', this changes the distributions of letters that may appear in other positions. We can improve it by using a dictionary that contains the most common answers to the projected positions.

3.2 Projection dictionary attacks

Using a dictionary D during the attack we could draw words at random; if the dictionary matches the target set of passwords, the success rate would be $\frac{\beta}{|D|}$. But for guessing an answer to the next challenge, there is a better strategy. Because many words may share the same projections onto the challenged positions, some responses are more frequent than others. We can pre-compute for this attack, by taking an ordinary dictionary and building a *projection dictionary* that contains combinations of positions from words and ranks the results by frequency.

For example, taking a 11,660 word dictionary of 8-letter English words[3], we find there are 2,736 possible answers to the positions 2, 3 and 6 (about 16% of the possible $26^3 = 17,576$) and 1,793 answers to challenges of the first three positions (only 10% of all possible). The projection dictionary tables give the top β guesses for each challenge, ranking them by number of occurrences, and giving the cumulative fraction covered so far:

Challenge 2 3 6:				Challenge 1 2 3:			
1.	r a i	79	0.69	1.	c o n	116	1.00
2.	r e i	77	1.34	2.	d i s	88	1.75
3.	r o i	63	1.88	3.	p r o	83	2.46
4.	l o i	59	2.38	4.	o v e	75	3.11
5.	l a i	57	2.87	5.	p r e	74	3.74

So, for example, for the challenge 1 2 3, the prefix "ove" occurs in 75 words, and the first four answers cover 3.11% of the set. The β-success rate is given by the cumulative coverage at position β. For β=10 and m=3 we get rates of 5.1% and 6.3% for the two challenges shown; these are close, but there wider variations: the most successful (weakest) case is 6 7 8 which achieves 30% after 10 guesses; this is skewed by the very common ending "ing".

Fig 4 shows success rates of the projection dictionary attack for larger numbers of guesses. By comparison, a brute force attack would take up to $N^m = 17,576$ guesses; an ordinary (unordered) dictionary-based attack is also shown on the graph as a diagonal line, this does better than brute force (omitting responses that do not occur), but not as well as the ordered attack using projections.

[3] more precisely: `/usr/share/dict/words` on Ubuntu 12.04, converted to lower case.

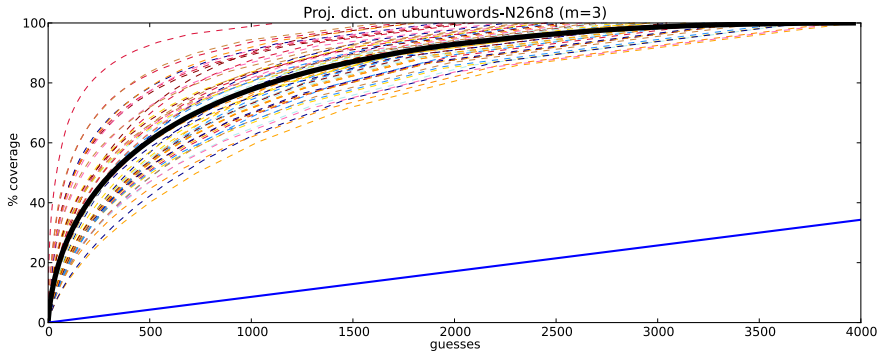

Fig. 4. Projection dictionary attack for an English dictionary

3.3 Projection dictionaries from password distributions

We could use the English projection dictionary, but a better attack is possible by starting with a dictionary that more closely matches the likely target distribution of passwords, such as a real leaked password list. Many leaked password sets have been analysed by researchers, to measure the success rates of attacks that use them [10,12] and also by practitioners, to build better cracking tools [18]. The RockYou data set (leaked in 2009) is one of the most useful, with over 32 million entries. Although it is captured from a social gaming website and one might hope that users choose better passwords to secure their bank accounts than for online gaming, there is some evidence that passwords used to secure financial information are not necessarily better chosen [15]. Moreover, some bank implementations actually encourage weak choices for partial passwords by not enforcing any password composition rules and by restricting the character sets available (presumably for usability reasons).

If we take the 8-letter alphanumeric passwords from the leaked RockYou data set, including frequency counts for the words, we get a ranked dictionary:

RockYou password frequencies (N=36, n=8):		
1. password	59462	1.01
2. iloveyou	49952	1.85
3. princess	33291	2.41
4. 12345678	20553	2.76
5. babygirl	15163	3.02

again, showing frequencies and cumulative percentage coverage of the whole set. There are about 5.9m 8-character alphanumeric passwords in RockYou, with 2.5m distinct, only 0.00009% of the 36^8 possible combinations; the top ten covers 3.88% of the whole set. For PINs, there are about 2.3m 6-digit passwords appearing in RockYou, but only 390,000 are distinct. The top 6 choices cover 15.3% of the whole set, although this is very skewed by the top choice "123456" (12.8%). In trawling attacks on data sets sharing the same distributions this

means 1 in 26 alphanumeric passwords could be broken by making the top 10 guesses, and 1 in 7 PINs could be broken within 6 guesses.[4]

Responding to a partial password challenge we can do even better. We build a projection dictionary taking frequencies into account; by construction, this can only improve the success rate. For example, for alphanumeric passwords:

Challenge 2 3 6:				Challenge 1 2 3:			
1.	a s o	64819	1.10	1.	i l o	76508	1.29
2.	l o y	52074	1.98	2.	p a s	66758	2.42
3.	r i e	47833	2.79	3.	m a r	58058	3.40
4.	2 3 6	24857	3.21	4.	b a b	52785	4.30
5.	a r e	21192	3.56	5.	p r i	46565	5.08

The effect of combining words to rank projections is apparent by comparing this with the previous table. For example, the most frequently occurring 8-letter password in RockYou is "password" but responding to a challenge on the first three positions, an attacker will succeed slightly more often with the letters i l o, the projection on the second most common word "iloveyou". This is because there are more variations that share those first three letters. (See Appendix A.3 for some examples with PINs.)

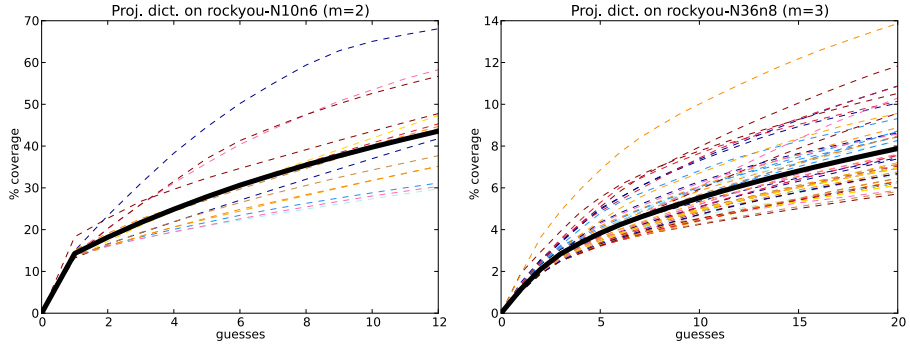

Fig. 5. Projection dictionary attacks for RockYou PINs and passwords

Fig. 5 shows the success rates of the projection dictionary attack on RockYou PINs and passwords themselves for smaller numbers of guesses. For PINs, the rates at our limit point β=6 vary from 22% to 50% with an average of 30.6%; for passwords at β=10, the rates vary from 4.2% to 10%, with the average of 5.5%. These are our best success rates for guessing the answer to the next challenge.

[4] In [9] the success rate for guessing 4 digit PINs with 6 guesses is given as 12.29%, based on almost 1.8m PINs assembled from *all* 4 digit sequences appearing anywhere in a password; the RockYou set only contains 20,661 4-digit passwords.

4 Recording attacks

The attacks considered in the last section do not require the attacker to eavesdrop. But if an attacker can record previous challenges and responses, he will be able to gradually learn the password, as well as make better informed guesses.

4.1 Pure recording attacks

A *recording attack* consists of observing k previous protocol runs and using the recorded challenge-response pairings to either learn the full password, or to respond to a new challenge. And once $k > 1$ runs have been recorded, an attacker can respond to an increasing number of challenges. For example, after recording the responses corresponding to the challenges 1 3 5 and 2 4 5, new challenges such as 1 2 4 can be correctly answered.

How quickly are positions learned? The following recurrence defines the probability $p_n^m(i,k)$ of observing exactly i different positions after k runs:

$$p_n^m(i,k) = \begin{cases} \frac{1}{\binom{n}{m}} \sum_{j=0}^{m} \binom{i-j}{m-j}\binom{n-(i-j)}{j} p_n^m(i-j,k-1) & m \le i \le n,\ k \ge 1 \\ 1 & i = 0,\ k = 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

We can use Equation 1 to recursively compute a probability for run k as a function of the number of positions observed at run $k-1$, by summing over the number of fresh positions j that are learned in the kth run and considering how the remaining $m-j$ positions may be chosen from positions already observed.

Fig. 6 (left) shows the probability $p_n^m(n,k)$ of observing all n positions after k runs for different choices of n and m. For both our PIN (n=6, m=2) and alphanumeric cases (n=8, m=3), after k=6 recorded runs an attacker has a greater than 50% probability of learning the full password. This is purely recording with no guessing, so it is unaffected by the N or β parameters.

In terms of guessing the next challenge after recording k runs, if the attacker knows $m \le i \le n$ password positions, then the proportion of challenges to which they can immediately respond can be computed as a fraction $s_n^m(i)$. We can then compute the proportion of challenges learned after k runs as $\overline{s_n^m}(k)$.

$$s_n^m(i) = \frac{\binom{i}{m}}{\binom{n}{m}} \qquad \overline{s_n^m}(k) = \sum_{i=m}^{n} p_n^m(i,k) s_n^m(i) \tag{2}$$

The results are displayed for different choices of n and m in Fig. 6 (right). For both our PIN and alphanumeric cases, after k=4 recorded runs an attacker has a greater than 50% probability of learning the next challenge. More generally, 50% of challenges can be answered after $3 \le k \le 7$ for password lengths of $8 \le n \le 11$, and common choices of m=2 or m=3.

The attacker may be able to find out the password, or answer the next challenge, quicker still than this, by using partial recording information and then making guesses on unknown positions.

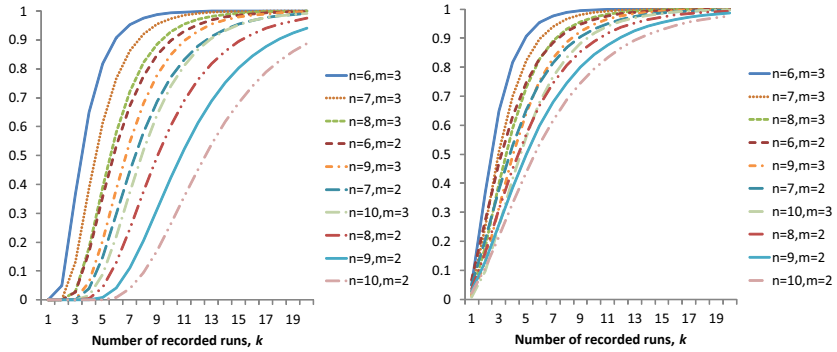

Fig. 6. Probability $p_n^m(n,k)$ of learning password (L), or $\overline{s_n^m}(k)$ next challenge (R).

4.2 Recording plus guessing attacks

Let $0 \leq m' \leq m$ be the number of positions that an attacker might know in a given challenge. If an attacker knows $i \leq n$ password positions, then the proportion of challenges with m' known positions is given by $s_n^m(i,m')$ in Equation 3 (generalizing Equation 2).[5] As before, we can compute the fraction of challenges with $0 \leq m' \leq m$ known positions after $k \geq 1$ runs as $\overline{s_n^m}(k,m')$:

$$s_n^m(i,m') = \frac{\binom{i}{m'}\binom{n-i}{m-m'}}{\binom{n}{m}} \qquad \overline{s_n^m}(k,m') = \sum_{i=m}^{n} p_n^m(i,k)s_n^m(i,m') \tag{3}$$

When an attacker knows $0 \leq m' < m$ positions in a challenge after k runs, guessing can be used to determine remaining $m-m'$ response positions. We can calculate the β-success rate to give us the fraction of responses that are correctly learned after β guesses, as shown in Equation 4.

$$\sum_{j=0}^{m} \overline{s_n^m}(k,j)w_j \tag{4}$$

Here, the new term w_j represents the success rate of guesses when $j \leq m$ positions are known in a challenge, which depends on β and N and varies according to the guessing method. The sum weights w_j by the proportion of challenges with $0 \leq j \leq m$ matches to recorded positions.

First, we combine recording and brute force guessing. If m' positions are known in a given challenge, then at most $N^{m-m'}$ guesses are required to brute force guess the remaining $m-m'$ positions. Here w_j is the probability of guessing correctly with β guesses, which is 1 when the number of guesses is less than or equal to the maximum allowed, β:

[5] The computation for $s_n^m(i,m')$ was used by Goring et al [5] to determine the expected number of positions known in subsequent challenges.

$$w_j = \begin{cases} 1 & \text{if } N^{m-j} \leq \beta \\ \frac{\beta}{N^{m-j}} & \text{otherwise} \end{cases} \tag{5}$$

The results are displayed in Fig. 7 which show different choices for n and m. For the PIN case (third line up on left graph of Fig. 7) the proportion of guessable challenges exceeds 50% after only k=2 runs, while for the alphanumeric case (third line down on right graph in Fig. 7) it takes k=3 runs.

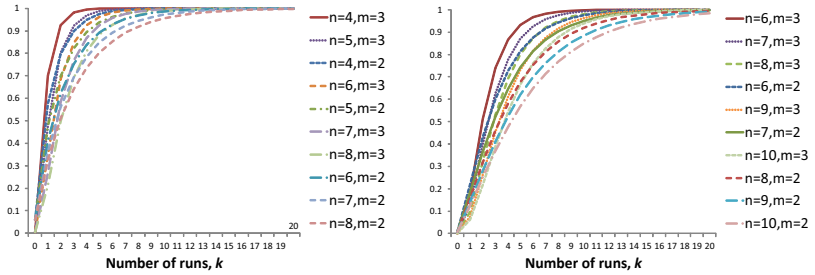

Fig. 7. Success rates for recording and brute force guesses, N=10 and N=36

We can improve the performance beyond brute force guessing by using our letter position frequency and projection dictionary attacks of Section 3. We provide a lower bound for the success rates in these cases by taking the best values for each w_j from Equation 4 for our example cases. In the alphanumeric case, w_0 represents the number of guesses when j=0 challenge positions are known; we earlier calculated a success rate of 5.5% using a projection dictionary. For w_1, guessing the two remaining positions, we take the success rate from a projection dictionary with m=2 sized challenges, which is 12% (versus 4% for the letter position frequency option). This is a rough lower bound: it treats the two remaining challenge positions as a lone challenge without taking into account any dependence upon the known password positions, which would reduce the answers possible (but would require the attacker to use more background dictionary data). Based upon a letter position frequency attack we computed a success rate of 60% for w_2 where we guess one remaining position in the challenge. Again, this success rate would be improved if knowledge of the two known positions is used. Finally, w_3=100% since all three positions are known.

For the PIN case, we similarly chose values $w_0 = 30.6\%$, $w_1 = 77.35\%$ and $w_2 = 100\%$. By plugging these values in Equation 4 we can compute the success rate for recording and best guessing attacks. The results for both k=1 and k=4 recorded runs are shown in our summary table in Fig. 8.

5 Summary

Partial passwords, introduced to prevent a telephone operator learning a user's password, have taken on a broader role in securing the online accounts of many banks. In addition to their likely susceptibility to guessing attacks, they do not appear to be adequate even to mitigate against a small number of recorded protocol runs, at least for typical choices of parameters used today.

Fig. 8 displays a summary of our attacks, showing success rates responding to challenges on partial PINs and passwords with the typical parameters sizes we found. For guessing attacks, our results are relative to data from the RockYou leak, and cannot be taken to be accurate for real PINs and passwords actually used in online banking. Nevertheless, these success rates are worryingly high, especially for banks that allow weak passwords and do not use a second credential or rely upon a second credential that may be easily obtainable such as a credit card number. Within the limited scope of our survey, Standard Life and 3DSecure for Bank of Ireland look at risk here, although, like some other banks, it appears that user ids are an additional credential for Standard Life as they are bank-issued and, we assume, unpredictable if they are not recorded.

Attack type	**parameters**	**percentage success rate**	
		PINs: β=6, N=10, n=6, m=2	alphanumeric: β=10, N=36, n=8, m=3
Brute force		6	0.002
Letter position frequency	RockYou	17.2	0.3
Dictionary	RockYou	15.3	3.9
Projection dictionary	RockYou	30.6	5.5
Recording	k=1 (k=4)	6.7 (63.1)	1.8 (59.0)
Recording + BF Guessing	k=1 (k=4)	41.1 (83.8)	9.6 (69.1)
Recording + Best Dictionary	k=1 (k=4)	60.2 (90.4)	25.2 (81.2)

Fig. 8. Summary of next-challenge attacks on partial passwords

Once recording takes place, attack success rates rapidly increase. It is claimed that half of online banking users access their account at least weekly [19,20]. Malware detection and removal, if operating, has a similar frequency, suggesting that if a key-logger is installed on a user's machine (or a public terminal), there is a good chance that at least one run of a partial password will be recorded. With k=1, only the PIN case yields a >50% success rate (k=2 for the alphanumeric case), so it can be argued that the partial mechanism provides *some* improvement over normal password authentication where an observer learns a complete password in a single step. As more runs are recorded, all success rates exceed 50%, and there is question over whether the partial scheme adds any useful security; reliance on the additional authentication mechanisms (where applicable) becomes primary.

Nonetheless, since the scheme is already widely used, it seems worthwhile for banks to enforce parameter choices that might improve the resistance to attack. There is an optimisation problem to solve here, to balance parameters both for sufficient usability and security. While a larger N improves resistance to guessing, it does not affect recording. So it may be worth first looking at the graphs in Fig. 6, which show that, unsurprisingly, larger password lengths n and smaller challenge sizes m increase the attack difficulty (the latter because it reduces the rate of revealing characters). But reducing m too far raises the success rate of guessing. Keeping m=3, but with $n = 16$, it requires recording 8 runs before the success rate exceeds 50% for an alphanumeric partial password. Longer passwords may be supportable by using passphrases (with a suitable interface) or even physical cards to store longer passwords; the latter is done with some products in a two-dimensional matrix [21].

Further work. There are several areas we want to investigate further. We have started to extend our study to the case considered by Goring et al [5] where the observer captures the response but not the corresponding challenge, which is the case when using a hardware key-logger. Varying the challenge format or sequence can have an effect. For example, repeating positions or giving positions in any order, not only ascending (one of our surveyed banks does this) greatly increases the number of challenges which would thwart the hardware key-logger case. We also want to extend our analysis to model *adaptive* projection dictionary attacks whereby guesses use the information previously learned. However, this requires the attacker to have the full dictionary, not just the top β answers for each challenge. Finally, we also plan to explore the usability of partial passwords, to explore the impact of parameter choices on users, as well as the effect of different presentation formats (e.g., drop-down menus versus text boxes) and challenge schedules. In the case of the latter, it may be interesting to examine how users react to different schedules, such as those generated maliciously in order to quickly reveal as many password positions as possible.

Acknowledgements. We're grateful to a student helper who spent several months gathering the survey data used in Section 2 and to Ronald Bowes for hosting some of the password sources we used on his blog [18].

References

1. UK Consumers Association: Bank websites: How safe is yours? Which? Magazine (September 2011) 24–27
2. Matsumoto, T.: Human identification through insecure channel. In Davies, D.W., ed.: EUROCRYPT. Volume 547 of LNCS., Springer (1991) 409–421
3. Li, X.Y., Teng, S.H.: Practical human-machine identification over insecure channels. Journal of Combinatorial Optimization **3**(4) (1999) 347–361
4. Hopper, N., Blum, M.: Secure human identification protocols. In Boyd, C., ed.: Advances in Cryptology ASIACRYPT 2001. LNCS 2248. Springer (2001) 52–66

5. Goring, S., Rabaiotti, J., Jones, A.: Anti-keylogging measures for secure internet login: An example of the law of unintended consequences. Computers & Security **26**(6) (September 2007) 421–426
6. Berkman, O., Ostrovsky, O.: The unbearable lightness of PIN cracking. In Dietrich, S., Dhamija, R., eds.: Financial Cryptography and Data Security. LNCS 4886. Springer (2007) 224–238
7. Focardi, R., Luccio, F.: Guessing bank PINs by winning a mastermind game. Theory of Computing Systems **50**(1) (2012) 52–71
8. Bonneau, J., Just, M., Matthews, G.: What's in a name? Evaluating statistical attacks on personal knowledge questions. In Sion, R., ed.: Financial Cryptography. LNCS 6052, Springer (2010) 98–113
9. Bonneau, J., Preibusch, S., Anderson, R.: A birthday present every eleven wallets? The security of customer-chosen banking PINs. In: Financial Cryptography and Data Security. LNCS 7397. Springer (2012) 25–40
10. Weir, M., et al.: Testing metrics for password creation policies by attacking large sets of revealed passwords. In: Proc. 17th ACM conference on Computer and Communications Security. CCS '10, ACM (2010) 162–175
11. Kelley, P.G., Komanduri, S., Mazurek, M.L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L.F., Lopez, J.: Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In: IEEE Symposium on Security and Privacy, IEEE Computer Society (2012) 523–537
12. Malone, D., Maher, K.: Investigating the distribution of password choices. In: WWW, ACM (2012) 301–310
13. Brostoff, S., Sasse, M.A.: Ten strikes and you're out: Increasing the number of login attempts can improve password usability. In: Proceedings of CHI 2003 Workshop on HCI and Security Systems, John Wiley (April 2003)
14. Just, M., Aspinall, D.: On the security and usability of dual credential authentication in UK online banking. In: 7th International Conference for Internet Technology and Secured Transactions (ICITST 2012), IEEE (December 2012)
15. Bonneau, J.: The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In: IEEE Symposium on Security and Privacy, IEEE CS (2012) 538–552
16. Yan, J.J.: A note on proactive password checking. In: Proc. 2001 New Security Paradigms Workshop. NSPW '01, ACM (2001) 127–135
17. Narayanan, A., Shmatikov, V.: Fast dictionary attacks on passwords using time-space tradeoff. In: Proc. of the 12th ACM CCS, ACM (2005) 364–372
18. Bowes, R.: SkullSecurity blog, passwords page. `http://www.skullsecurity.org/wiki/index.php/Passwords`. Accessed September 2012.
19. Mahmood, Z.: Attitudes towards the use of e-banking: Result of a pilot study. Communications of the IBIMA **8** (2009) 170—174
20. Thepaypers.com: UK consumers prefer online banking - survey. (May 2011)
21. Voice, C.B., Chiviendacz, M., Pillman, E.: United states patent: 8060915 - Method and apparatus for providing electronic message authentication (November 2011)

A Appendix

In this appendix we provide more detail behind our calculations and data sets.

A.1 Bank survey resources

We gathered data directly from personal accounts and also consulted the following demo and information pages (see collected parameters in Fig. 2):

- ING DiBa: `https://www.ing-diba.de/kundenservice/banking-und-brokerage/#!01091`
- First Direct: `http://www1.firstdirect.com/1/2/banking/ways-to-bank/online-banking`
- Smile: `http://www.smile.co.uk/images/flash/smiledemo/`
- HBoS: `http://www.onlinebankingdemo.co.uk/launchbos.html`
- Nat West: `http://www.rbs.co.uk/personal/online-banking/g1/existing-customers/problems-logging-in.ashx`
- 3DSecure: `http://www.bank-of-ireland.co.uk/3dsecure/`
- AIB: `http://www.aibgb.co.uk/onlinebankingdemo/index.html`
- Bank of Ireland: `https://personalbanking.bankofireland.com/online-banking-demo/`
- Standard Life: `http://www.standardlife.co.uk/html/demo/home1.html` and `http://www.standardlife.co.uk/1/site/uk/help/faqs/online-servicing`
- Cooperative: `http://www.co-operativebank.co.uk/bankdemo/2008-06-04/IB_Demo_v2.html`
- Nationwide: `http://media.nationwide.co.uk/swfs/Demos/default.htm?DemoId=7`
- Skipton: `http://www.skipton.co.uk/demo/`
- Tesco: `http://www.tescobank.com/demos/index.html`
- Barclays: `http://www.barclays.co.uk/online/demo/?WT.ac=coukolbdemo`
- HSBC: `https://www.hsbc.ca/1/2/personal/banking`

A.2 Letter-position frequency data

Fig. 9 shows the relative letter position frequencies from the RockYou set calculated on our two parameter cases: 6 digit PINs and 8 character alphanumeric passwords. These illustrate the statistics used in Sect. 3.1, demonstrating how

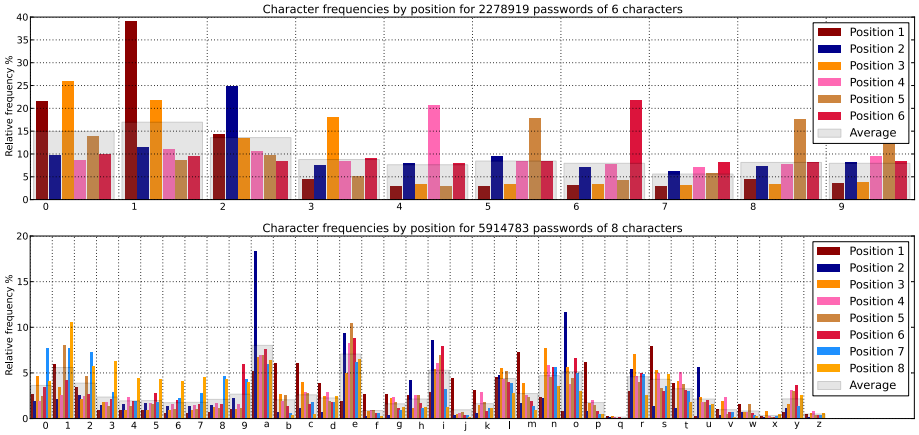

Fig. 9. Letter frequencies by position for RockYou passwords

frequencies of letters vary by position. The ranking of the top 10 characters in each position from this data is shown below, for each case:

1	1	0	2	8	3	9	6	7	5	4
2	2	1	0	5	9	4	3	8	6	7
3	0	1	3	2	9	6	8	5	4	7
4	4	1	2	9	0	5	3	6	8	7
5	5	8	9	0	2	1	7	3	6	4
6	6	0	1	3	9	2	5	8	7	4

1	s	m	p	b	c	1	a	l	j	d
2	a	o	e	i	u	r	l	h	2	n
3	n	r	a	o	l	i	s	e	0	t
4	e	a	i	n	t	s	r	l	o	y
5	e	1	i	a	l	2	n	o	r	t
6	e	i	a	o	9	n	r	1	l	y
7	0	1	2	e	a	n	o	r	8	9
8	1	e	a	3	2	s	7	4	8	5

So, for the challenge 2 3 6 the simplistic attack measured in Sect. 3.1 would guess in turn "a n e", "o r i", "e a a" and so on. We measure the success rate by seeing what proportion of the data set is then broken by making β of these guesses. This attack is not particularly successful, but is meant to illustrate a simple improvement using position data that improves over guessing at random.

A.3 Projection dictionary data

Projection dictionaries are built by taking a source dictionary, finding the answers to each of the $\binom{n}{m}$ challenges, and ranking each result by the number of times it occurs. The calculations are performed using some scripts written in Python. For PINs, an example table entry looks like this:

```
# Challenge (1, 3) has 100 distinct responses, and 2278919 total
# Challenge     Response Occurrences    Coverage %   Cum. %
  1. (1, 3)     13       340719         14.951%      14.951%
  2. (1, 3)     00       191577          8.406%      23.357%
  3. (1, 3)     11       175642          7.707%      31.065%
  4. (1, 3)     10       164900          7.236%      38.301%
  5. (1, 3)     01       143302          6.288%      44.589%
  6. (1, 3)     20       127502          5.595%      50.184%
# Metrics for Challenge (1, 3): Beta success rate: 50.18%
```

An attacker following this strategy would, given the challenge (1,3), reply with responses 1,3, then 0,0 and so on. As with the alphanumeric example, this ordering differs from the global ordering, where the top six PINs occurring as RockYou passwords were:

```
Passwords selected from data/rockyou-N10n6.txt
Selected passwords of length  6
Total of 2278919 passwords, with 390529 unique (17.14%)
Most common password '123456' occurs 290729 times
#    Word    Occurrences  Coverage %  Cum. %
  1. 123456  290729       12.757%     12.757%
  2. 654321  13984        0.614%      13.371%
  3. 111111  13272        0.582%      13.953%
  4. 000000  13028        0.572%      14.525%
  5. 123123  9516         0.418%      14.943%
  6. 666666  7419         0.326%      15.268%
# Password metrics: Beta success rate (beta = 6): 15.268%
```