Bootstrapping Mobile PINs Using Passwords

Markus Jakobsson
Information Risk Management
PayPal
San Jose, CA
majakobsson@paypal.com

Debin Liu
Information Risk Management
PayPal
San Jose, CA
deliu@paypal.com

***Abstract*—We describe a method of deriving PINs from passwords. The method is useful to obtain friction-free user onboarding to mobile platforms. It has significant business benefits to organizations that wish to introduce mobile apps to existing users – but which are reluctant to make the users authenticate with passwords. From the user's perspective, a PIN is easier to enter than a password, and a *derived* PIN does not need to be remembered – assuming the user can recall her password. The use of tiered authentication – relying on both PINs and passwords – hardens systems against compromise. This is because transactions relying on PINs can have lower transaction limits and flagging thresholds than transactions authenticated using passwords. Even though our PINs are derived from passwords, they do not contain sufficient information about the passwords to make the passwords easy to infer from compromised PINs. We quantify exactly how much information about the passwords and the derived PINs contain, and how much information is *lost* – based on real-life password distributions. We also assess the usability of the proposed method using one 25-subject qualitative study and one 100-subject quantitative study.**

***Keywords*-bootstrapping, dropbox, entropy, malware, password, PIN.**

I. Introduction

Consumers do not like passwords – especially if they have to be entered on a handset [13]. In a recent study [11], it was shown that the time to enter a typical strong password is on the order of 2-3 times longer on a handset than a regular keyboard.

Financial service providers *also* do not like passwords for handsets, but for a different reason: Handsets are believed to be more likely to be targeted by fraudsters than PCs are – at least in the near future. This is based on a shift from phishing to malware [2], [8], an explosive growth of the mobile market [10], and structural vulnerabilities in the current malware defense paradigm [12] for battery-constrained devices. Trusted computing [15] may help address the problem, but has so far not done so. Handsets are also vulnerable to threats that PC are not currently exposed to – such as malware propagation over Bluetooth [4]. Although the threat is nascent, it is starting to be given attention [9].

Having *both* passwords and PINs allows for a tiered authentication approach where certain types (and sizes) of transactions can be performed on mobile devices, and rely on PINs for authentication – while more secure devices and stronger authentication could be required for other transactions. In a way, PINs therefore *improve*[1] security relative to only using passwords: they can be used in contexts where passwords should not be exposed, and such PIN-authenticated transactions could be treated differently by the backend risk model than password-authenticated transactions would. This is contrary to the traditional wisdom that PINs, being shorter than passwords, are simply less secure.

As a result of user preferences and provider concerns, there is an industry trend towards using PINs instead of passwords on handsets, gaming consoles, store checkouts, and other appliances with similar constraints and security profiles. Many phone apps ask for a PIN instead of a password.

While PINs offer improved convenience to users and have security benefits in comparison to passwords, using PINs instead of passwords come with a multitude of drawbacks. From a business point of view, *any* increase of the end-user friction is worrisome. Having to force millions of existing users to create PINs is not desirable at all. Moreover, PINs are not as easy to remember as passwords, unless the number represents something to the user. As a result, approximately one user in five selects their birthday as a PIN [3]. This not only means that database-scraping attackers have an edge, but also leads to a restricted distribution of PINs: There are only 365 days in a year. It is also well understood that people commonly reuse PINs, just as they reuse passwords. This means that if one PIN is compromised, then that may put others at risk as well. For those who do not use any of these tricks to remember PINs, it is relatively common to *forget* them: Roughly one in ten report having forgotten their PINs [3]. Forrester research [5] estimates the average costs of help-desk assisted reset at $70 per reset, making forgotten

[1]There are other ways, of course, in which PINs offer lower security than passwords do. This paper does not argue in favor of either, but simply addresses the issue of how to create and manage PINs – if we are to use them. However, it should be noted that PINs are often considered *sufficiently* secure. This is since financial service providers – and many other service providers – already use various backend security features to detect account take-overs. As a result, PINs provide sufficient security for many common types of transactions – such as paying a vendor who has previously been paid by the user in question.

credentials a commercially very expensive security problem.

This paper describes a method that *bootstraps* the generation of PINs to create an automated onboarding of an overwhelming majority of users. We achieve this by deriving the PINs from already established passwords – without explicit user involvement. The user would, in fact, not even be aware of the PIN creation until she is told that she *has* a PIN, and should use it to log in. The PIN is set as the first four characters of the password, mapped to a numeric keypad. This mapping is analogous to how alphanumeric phone numbers are mapped to a 10-button keypad – like 1-800 CALL ATT *becomes* 1-800 225 5288. Similarly, the password "Blu2thrules" becomes the PIN "2582"[2].

We propose to use mapped password prefixes as PINs, and describe how to derive these on the backend. We also show that the resulting system is usable, and analyze how much information a derived PIN reveals about the underlying password. This is relevant to consider in order to understand the consequences of a PIN being compromised. The analysis is based on tens of thousands of actual passwords, and their associated derived PINs. We show that derived PINs have approximately the same entropy as traditional PINs do, and hence, do not reduce security.

Outline: We begin with a review of related work (section II). We then detail our proposed method, both from the user's perspective and in terms of the backend processing (section III). In section IV we describe how we determined the entropy of passwords and PINs derived from these, after which we detail the security implications of our proposed method (section V). In appendix A, we briefly report on a study we performed aimed at determining how people select PINs, assessing the approximate security of PINs based on those results.

II. Related Work

Florêncio and Herley [6] reported that typical users hardly use uppercase and special characters. Our findings support that claim. Florêncio and Herley [6], [7] also computed the entropy of each password by treating passwords as strings generated uniformly at random: They assumed that the characters of one password are *independent* of each other, and conclude, as a result, that typical 8-character passwords containing numerical, upper case and lower case characters would have around 42 bits of entropy. Unfortunately, and as evidenced by our findings, this is not so: There is a *lot* of correlations between characters of typical passwords. We assess the amount of entropy in the first four characters of passwords, and find that it is much lower than postulated by Florêncio and Herley, but a bit higher than estimated by NIST [14].

As a result of potential independent interest, we also compare the approximate entropies of the first four characters of passwords for different domains – a large financial service provider (we will refer to this as *FSP* for Financial Service Provider), a large social networking provider (we will refer to this as *SNP* for Social Network Provider), and a collection of mixed domains. These entropies are not the same, suggesting that password rules and differing security mentalities effect the strength of passwords in a noticeable manner.

Passwords are not stored in cleartext on the backend, as this would be a great liability, should the backend database be compromised. Instead, each password is concatenated with a random value (the so-called *salt*), then hashed using a function such as SHA-2 [1]. The output of the hash function is stored, along with the associated salt value. This approach drastically increases the cost for an attacker of exhaustively searching the password space, determining matches using a compromised password database. This is because it is not possible to build a dictionary of hashed password values given a dictionary of plaintext passwords, and determine what the entries of a compromised password database correspond to. This, of course, is because each entry depends not only on the password, but on the salt value as well. PINs can be stored in the same way.

III. Proposed Method

The User Perspective. Users would not need to know that PINs are derived from their passwords. It will happen[3] without their knowledge, as they authenticate to the system using a password. When a user for whom a PIN has been derived arrives to the mobile authentication interface (or another PIN authentication interface), she would be told that she has a PIN – and how to derive it from her password. As a result, users do not have to remember the PIN – it is enough to remember the password. A screenshot of a possible user interface is shown in figure 1. Relying on user experiments, we support that it is straightforward for individual users to perform the required mapping.

Derivation Approach. Given how passwords are stored on the backend, it is not possible to derive PINs from *stored* passwords. However, it is possible to derive PINs from passwords when these are temporarily available in plaintext – as they are each time a user logs in using a password. Therefore, we can compute the derived PIN from the plaintext password once it has been verified to be correct – and right before the plaintext password is erased on the

[2]While we think of PINs as four-digit elements, it is straightforward to modify our techniques to derive six-digit PINs. However, while this leads to a reduced risk of PINs being guessed, the amount of information in the PIN about the associated passwords increases. This has negative effects in situations where PINs are compromised.

[3]The PIN derivation will not be performed for users who already *have* PINs; it also will not be done for a small fraction of users who have passwords that cannot be mapped in a manner that would be simple to explain to the user. Correspondingly, the messaging to users of these two types would be different from that to users with derived passwords.

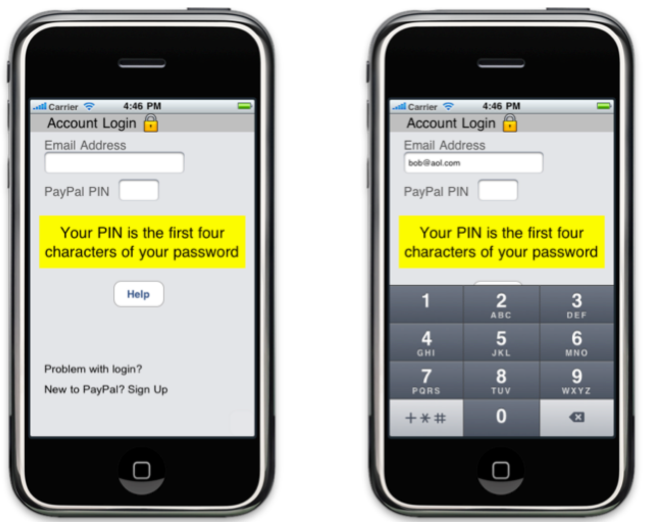


Figure 1. The figure shows the user interface. To the left is an image of what the user might see when arriving at the authentication stage; in some contexts, the user name may be auto-filled or obtained from a drop-down menu. The image to the right shows what she would see after tapping on the PIN window. To log in, the consumer would simply enter the first four characters of her password, using the numeric keypad. If a password character is "2", "A", "a", "B", "b", "C" or "c", then the user presses the 2-button – we are not case sensitive. A password starting with "Blu2" would correspond top the PIN 2852.

backend. The derived PIN can then be salted, hashed, and stored along with the salted and hashed password record.

Managing Special Cases. There are several special cases that need to be addressed for our approach to be practically viable. We list these here, along with our suggested approaches for dealing with them:

1) **Unmappable characters.** An unmappable character is a character that is not present on a typical numeric keypad. An example of an unmappable character is \$. In section IV, we determine that the frequency of passwords containing unmappable characters is marginal – 1.4% for FSP passwords, and lower for the other password sources studied. This special case can be addressed in at least two ways: (a) one can map all special characters to one digit (e.g., to "0"), or (b) one can simply "disqualify" passwords containing unmappable characters among the first four characters. The latter would force the owners of such passwords to create a PIN manually – in the old-fashioned way – or to update his/her passwords. One approach to increase automatic enrollment is to augment password strength checkers to reject unmappable characters in the first four positions of passwords that are created. Given the very low rates of unmappable characters in the first four positions, this has no practical impact on the security of the passwords.
2) **Strong passwords, weak PINs.** While all credentials are equally good from a theoretical point of view, some PINs are so commonly used that they are considered weak – "1234" is one such PIN. A credential is considered *strong* if it is sufficiently uncommon – otherwise *weak*. There are strong passwords that result in weak PINs. Example password such as "1234GreyFrieS#" and "1BeGood" both correspond to a derived PIN "1234".
   Whereas many password strength checkers will reject passwords with "weak" sequences – independently of the strength of the remainder – not all will. For example, many password strength checkers (like that of FSP's) will already reject the above example password. This is since it contains the sequence "1234", which is considered an *indicator* of a weak password – whether the password is actually weak or not. Similarly, many password strength checkers reject passwords that contain the name of the registered user, his or her zip code, and similar elements.
   To avoid weak derived PINs, one could reject the associated password (as is commonly the policy); accept the password but not derive a PIN; or derive a PIN but demand that it is updated on its first use. The approach taken is a matter of policy.
3) **Password changes.** If a PIN is derived from a password, then used for some time, then the user may either think of the PIN in terms of the password it is derived from or she may learn the PIN. In the latter case, the PIN would become independent of the password in the mind of the user. The backend cannot know whether this has happened or not. Therefore, if the user were to change her password, the backend could create two "parallel universes" – with the *old* derived PIN and the *new* derived PIN. Both are valid until one of them is used, at which point the other one is erased.
4) **Hardware keyboards.** Some handsets, like BlackBerries, have hardware keyboards. If the user is asked to enter the beginning of her password, this keyboard should not be used, as it would not cause the characters to be mapped as they are entered, and it is not reasonable to ask the end user to perform this mapping without the visual aid of the PIN pad. Instead, the hardware keyboard would be momentarily turned off, and the PIN entry would be done using the touch-screen keyboard[4].
   Other handsets, such as the typical 4-5 year old phone, have hardware keyboards with the proper mapping; however, people are used to pressing each button one or more times in a row to advance to the right character. This can be detected and compensated for

[4]Old BlackBerries do not have touch screens. Since apps are platform aware (if not platform specific), apps on non-compliant platforms would simply not offer users to log in using a derived PIN, but would require them to create a PIN out of band – as is currently required for *all* users.

on the backend, or the user can be instructed to only press each button once for each character.

Messaging. The backend "knows" for what users a PIN has been derived, and messages those users – but only those – when they arrive to the mobile portal (or another portal where PIN entry is preferred by the system.) The messages must be clear and instructive, and yet, be short and concise (to account for the limited screen size, and to avoid being ignored). We have tested a variety of messages, and have found that users understand the message "Your PIN is the first four characters of your password. Please enter your PIN."

IV. Analysis of Passwords and Derived PINs

It is not possible to determine the strength of passwords simply from their lengths or rules on what constitutes valid passwords. The reason is that passwords are not uniformly distributed within their associated ranges. To determine what the true distributions are and what the resulting degrees of security are, we analyzed a large number of real passwords obtained from raided dropboxes. A *dropbox* is the type of file used by fraudsters to temporarily store stolen credentials. Many financial organizations and security vendors attempt to raid dropboxes – i.e., locate them and copy their contents. This is done to restrict access to compromised accounts. We analyzed passwords from such dropboxes.

Some of the dropboxes were used by phishers, others by malware authors. One difference is that sometimes – very rarely – would-be victims of phishing attacks are aware of being targeted, and instead of entering their real credentials enter insults. This is not happening for malware dropboxes, where victims are more "sincere". Another difference is that phishing dropboxes typically only have credentials for one particular site (the spoofed site), whereas malware dropboxes contain any credential that the malware agent could obtain. (The latter also leads to a way of distinguishing between phishing and malware dropboxes.)

We analyzed the contents of three dropboxes, corresponding to FSP accounts (phishing, 8359 credentials); SNP accounts (phishing, 2873 credentials); mixed credentials (malware, 16192 credentials.) We refer to these as three *groups* of passwords onwards – corresponding to a major Financial Service Provider, a major Social Network Provider, and to mixed domains with passwords harvested by malware. We do not combine the contents of the groups further since the distribution of each class is different – we determined this using the Kolmogorov-Smirnov test and Wilcoxon Rand Sum test. The difference in distribution may be attributed to different rules of what a good password is on the various domains, and to differing user security mentalities for the three different types of domains. (The malware dropbox appears to have a large percentage of gaming passwords.)

Methodology for Assessing Entropy. We wish to assess the entropy of passwords of each group, corresponding to the three groups described above. For all the credentials in the dropboxes, we truncated them to the four first characters, resulting in what we refer to as pwd_4. We then counted – within each group of passwords – the relative frequencies of each value pwd_4. For example, the password "Blu2thrules" would be truncated to become $pwd_4 =$"Blu2", and a counter for this very string would be increased. After that, given the relative frequencies, we would estimate the entropies for the different groups.

There are at least $26^4 \approx 457000$ possible four-character combinations of the truncated passwords (not even taking case, numerical values and special characters into consideration). In contrast, we have between 2000 and 17000 samples for each group, which causes a rather low density of occurrences. This, in turn, results in complete lack of observation of some combinations that may occur in larger password populations, and a slightly inaccurate estimate for the frequencies in general. As a result, we this leads us to underestimate the associated entropies.

We then perform what we refer to as the *phone pad mapping* for all the truncated passwords pwd_4. For example, this would map the pwd_4 string "Blu2" to "2582", using the mapping resulting from the common interface shown in figure 1. As a result, there are exactly 10000 possible mapped strings. For each of these, we determine the relative frequency, given the mapped and truncated passwords of the three groups. Like the entropy assessments from the unmapped passwords, the entropy assessments of the mapped passwords will be slight underestimates (but closer to the truth given that the mapping increased the density.)

Entropies. Figure 2 shows the entropies of the first four characters of *unmapped* passwords and the derived PINs – i.e., the *mapped* and truncated passwords. The figure also shows the information loss during the mapping process.

These entropy estimates are 12, 10.5 and 9.7 for the three groups, and for the derived PINs, they are 10.9, 10 and 9.2. The groups, again, correspond to FSP passwords, SNP passwords and passwords from mixed domains.

We note that the sample sizes play a substantial role when producing entropy estimates – especially for the unmapped samples, where the density is sparser than for the mapped samples. To determine the effects of this, we sampled down the first group (FSP, 8359 samples) to the size of the second group (SNP, 2873 samples). We did this ten times for different randomized samples and averaged the associated entropy measures. The result is 10.9 for the down-sampled FSP, to be compared to the 10.5 of the SNP group, and 12 for the full FSP. (While – with similar sample sizes – FSP and SNP appear to have a similar entropy, we recall that we found that the samples are not drawn from the same distribution.)

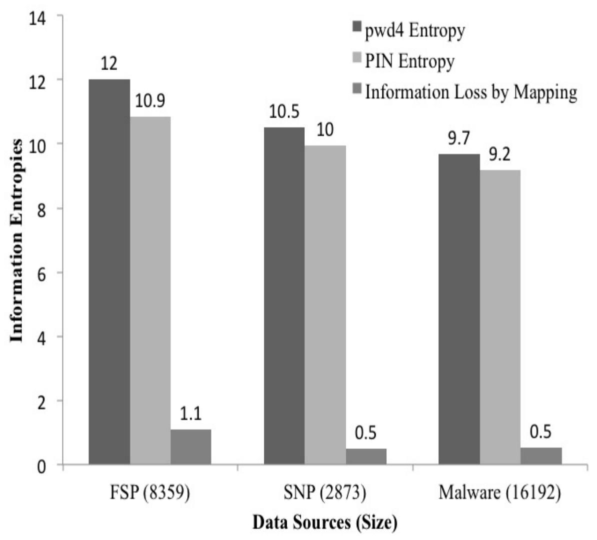


Figure 2. The figure shows the entropies of the first four digits of the studied passwords; the entropies of the corresponding PINs; and the information loss during the mapping process from passwords to PINs. The numbers are different for our three sources of passwords; most probably due to differing rules governing what passwords are accepted, and different security mentalities among end users. The first two groups correspond to phishing dropboxes for FSP and SNP, while the third group corresponds to a malware dropbox containing credentials to any domain that was accessed from the infected machines.

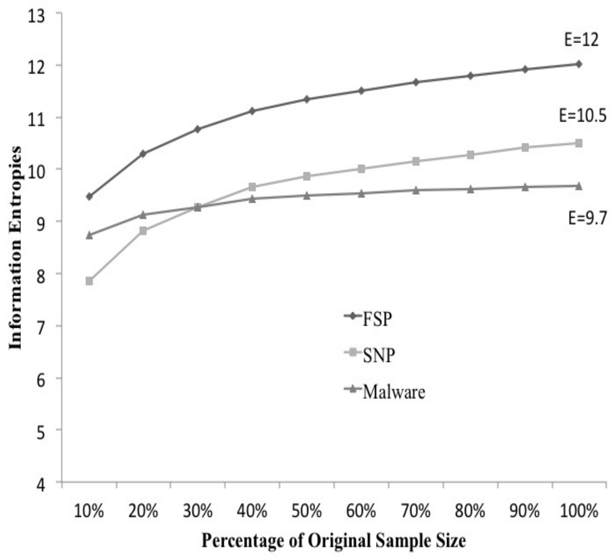


Figure 3. The figure shows a strong sample size dependency of entropy estimation for all three data sources. We know that the true entropies of the three sources are *at least* as big as the estimates for the 100% sample measurements, but extrapolation of these curves tell us that still bigger sample sizes would most likely have shown yet higher source entropies. It is an interesting open problem how to extrapolate estimates of this kind to assess the true entropy, given only samples of rather small sizes.

We sampled three data sources from 10% to 100% of their original size. Figure 3 shows a strong dependency on the sample size when estimating the entropy. This is due to the fact that low densities of samples create an artificially low estimate of the entropy, given how entropies are estimated.

We know that even larger sample sizes are likely to further increase the estimates of entropy. Therefore, we note that our estimates are *lower* estimates.

The *differences* between the entropies of the unmapped and mapped password collections correspond to the amount of information that is removed by the phone pad mappings. These are 1.1, 0.5 and 0.5 for the three groups. (The reason they are not the same are due to different sample sizes, different requirements on passwords, and potentially, due to different efforts among the users to select good passwords for the associated domains.)

Improving on NIST's Entropy Estimates. In addition to performing a four-character truncation of passwords (which we referred to as pwd_4), we also perform three-character, two-character and one-character truncations. We refer to these as pwd_3, pwd_2 and pwd_1. Using the same methodology as described above, we estimate the entropies. The entropy contributed by the first character is the entropy of pwd_1, or $E(pwd_1)$. The entropy of the *second* character is the difference $E(pwd_2) - E(pwd_1)$; that of the third character is $E(pwd_3) - E(pwd_2)$, and that of the fourth, not surprisingly, $E(pwd_4) - E(pwd_3)$. These estimates take correlation into consideration. They are specific to the various groups, since these are governed by different distributions.

We find that the entropies contributed by the first four characters are 5.1, 3.5, 2.4, 1 for FSP passwords; 5.1, 2.9, 1.9, 0.6 for SNP passwords, and 5.4, 2.6, 1.3, 0.4 for the mixed "malware" passwords. Figure 4 shows the entropies contributed by each of the first four characters. This is in slight contrast to the estimates made by NIST [14]. NIST suggests that the distribution of passwords corresponds to an entropy of 4 bits for the first character, 2 bits for the next 7 characters, and 1.5 bits per character for the 9th to the 20th characters, and 1 bit for the remaining of the password. 6 bits of entropy is added when the user is *forced* to use both upper case and non-alphabetic characters, but that does not apply to the FSP or SNP passwords.

Figure 5 compares two entropy estimates for each of first four characters for FSP passwords. Unlike our estimates, the "random" estimates do not take into consideration the correlation between positions. It is computed based solely on the relative frequencies of every character at each of the first four positions. The difference between our estimates and the random estimates illustrate that there are notable correlations between two positions, and the first four characters are not random strings. Passwords from SNP and Malware exhibit similar patterns.

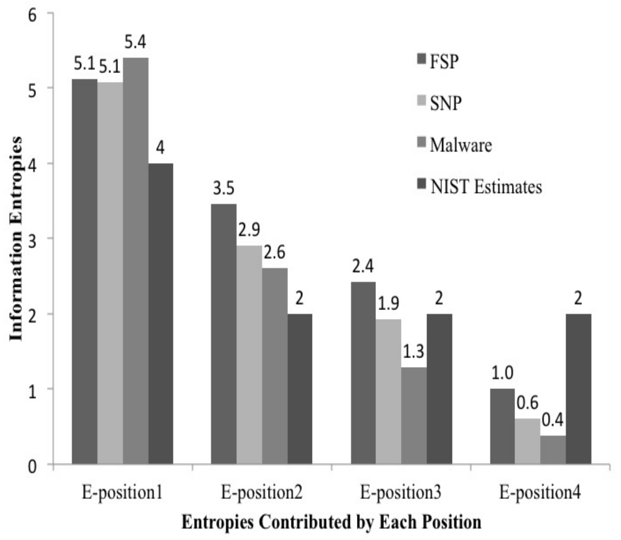


Figure 4. The figure shows the entropies contributed in each of the first four positions for all three data sources. These are *conditional* entropies, as they reflect the fact that the distributions of characters is conditional on previous password characters. This follows from the fact that passwords often are words, or parts of words, and words are not random strings.

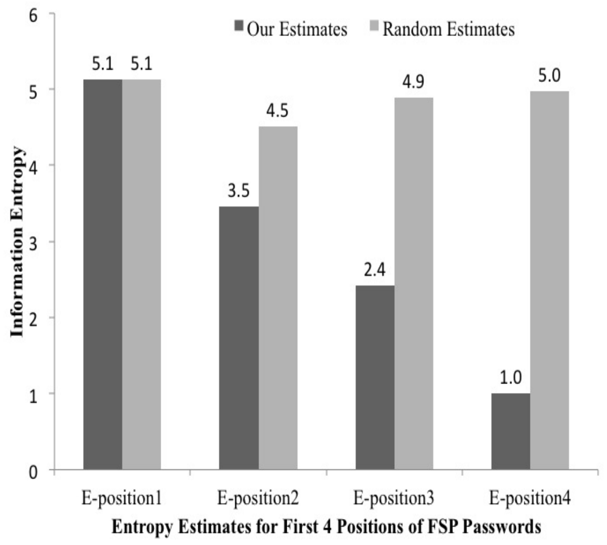


Figure 5. The figure shows the conditional entropies contributed in each of the first four positions, for the FSP password group. It also shows the entropies in each of the positions, which is an estimate that does not take correlation into consideration. The latter estimate is only meaningful when we consider a given position in isolation. The differences between these two estimates suggest that FSP password prefixes are not random strings.

Commonality of Special Characters. Among the passwords we reviewed, on average approximately 32% had both upper and lower case among the first four characters; only 3.2% had at least one non-alphanumeric character.

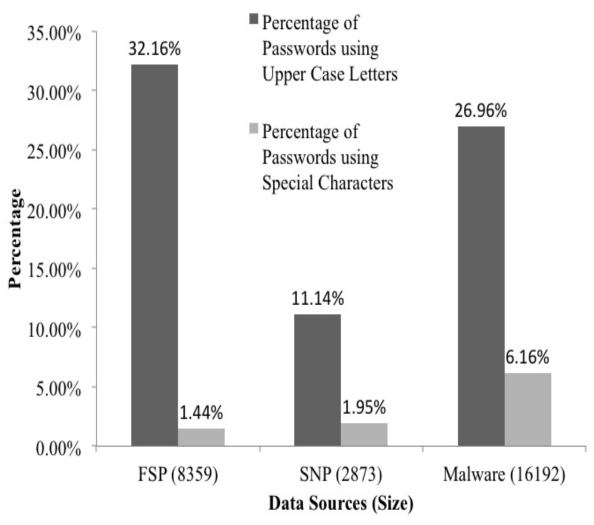


Figure 6. The figure shows the percentages of the passwords which contain upper case and special characters at any of the first four positions. Passwords with special characters in the first four positions can either have such symbols mapped to 0 or 1, or these passwords could be excluded from the PIN derivation process.

Our principal approach only considers the mapping of alphanumerical strings, and does not account for special characters (such as “@”, “$”, “+”) among the characters to be mapped to generate the derived PIN. However, we find that the frequency with which one or more of the first four characters is “unmappable” is rather marginal: It is 1.4% (FSP), 1.9% (SNP) and 6.2% (mixed passwords). Figure 6 shows the percentages of passwords using upper case and special characters.

Character Type. Florêncio and Herley [6] found that typical users hardly use upper case and special characters. We have the same observation for the first four characters of the passwords we study. All three sources exhibit similar character type distributions: lower case characters dominate among all positions; numerical characters are consistent around 10%; upper case characters are used more often as the first digit around 7%, then reduced to $2-3\%$ at other positions. Figure 7 shows the character type distribution in the first four positions of FSP passwords.

V. Security Analysis

Security Impact of Compromise. From the entropy estimates of section IV, we know that an adversary gains on average 10 bits of information about a password if he compromises the corresponding PIN – or 10.9 (FSP), 10 (SNP) and 9.2 (malware). The mapping itself destroyed 0.7

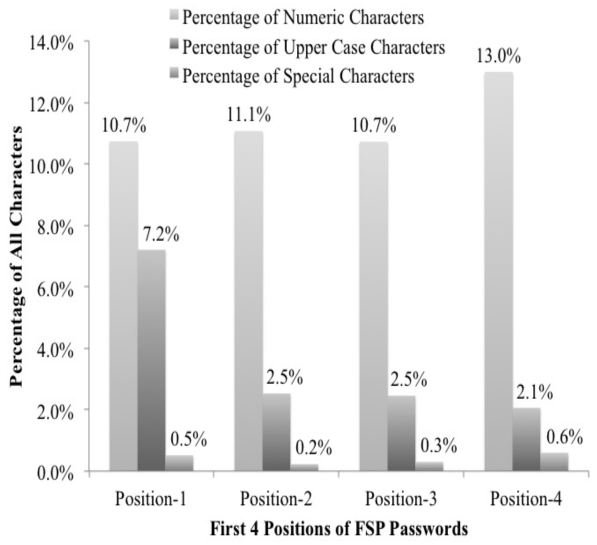


Figure 7. The figure shows the character type distribution at first four positions of FSP passwords. It is interesting to note that upper-case is much more common in the first position than elsewhere. The percentage of numeric characters is fairly even for the different positions.

bits of information on average[5], and the truncation the rest. The exact amount of information lost by the truncation depends on the length of the password. The average password length among the FSP and SNP passwords we analyzed were roughly 8 characters, which according to NISTs entropy estimates [14] corresponds to an entropy of 18 bits.

Our findings suggest that the actual entropy is a bit higher than what was estimated by NIST; however, these findings only apply to the first four characters. If we extrapolate our findings about the first four characters to the entire password and simply argue that each character has 7% more entropy than estimated by NIST (as supported by our findings for the first four characters), then our estimate of the entropy of the entire password is $1.07 \times 18 = 19.3$ bits instead of 18 bits.

We see that the attacker learns on average 10 bits out of the on average 19.3 bits of information, resulting in a *conditional entropy* of between $18 - 10 = 8$ bits (using NIST's estimate) to $19.3 - 10 = 9.3$ bits (using our extrapolated estimate). This corresponds to the security of a password for which the derived PIN has been compromised. While this is not terrific, it ought to be compared to the much starker situation in which the entire password is compromised, should passwords be used on insecure devices instead of derived PINs.

Comparing Derived PINs and Traditional PINs. In section IV, we determined that the entropies of derived PINs were 10.9, 10.0 and 9.2 for the FSP, SNP and mixed domains. This should be compared to the general estimates of the entropy of PINs that are *not* derived from passwords. An upper bound of the traditional PIN entropy is provided in Appendix A. According to those upper bounds, traditional PINs have an entropy of no more than approximately 10.2 bits. Therefore, we see that our derived PINs do not reduce security, but have approximately the same average security as traditional PINs. (There is anecdotal evidence suggesting that the lower quartile of derived PINs are notably more secure than the lower quartile of traditional PINs, though. This is due to the lack of very common combinations for derived PINs, and the relatively common use of years and dates as traditional PINs.)

Moreover, we note that if the (beginning of the) password is not reused, then the derived PIN is independent of other PINs created from other related passwords.

[5]The lost information due to the mapping for the three groups are 1.1 (FSP), 0.5 (SNP) and 0.5 (malware).

VI. Usability Experiments

We wanted to determine whether users would understand how to use derived PINs. We performed two studies: one qualitative, using an iPhone app and 25 subjects; and one quantitative using 100 subjects.

Qualitative Study. We performed a qualitative study using an iPhone app (shown in figure 1), using 25 subjects in the age group 30-50 years old. 16 were men, 13 were employed in the technology sector.

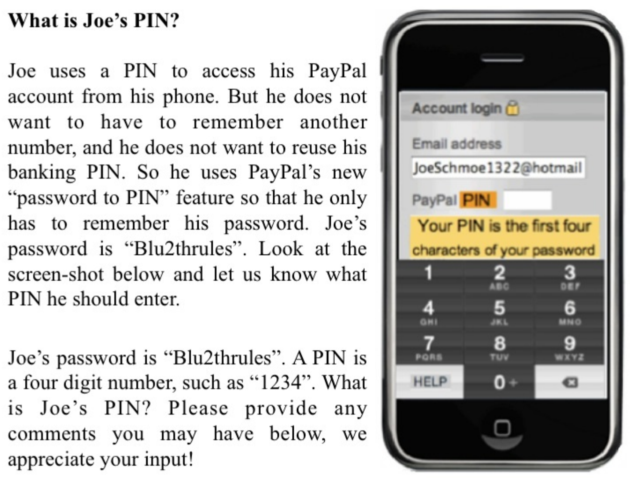
16 of the subjects entered the expected PIN with no hesitation. Four of these cited similar user experiences for dialing phone numbers or for entering last names using the number pad. The remainder offered no particular explanation of why they knew what to do.

Another six subjects (all of them men in the technology sector) took more than a minute to determine what to do, but then, correctly entered the PIN. Several of them argued that it would have been difficult to succeed if they had used a special character in the first four of the password.

Three of the subjects failed initially. All of them understood the process when given an explanation corresponding to what was later added under the help button.

Quantitative Study. We ran a user experiment on Amazon Mechanical Turk in which we recruited 100 subjects, all from the U.S., and only subjects with at least 95% positive feedback. They were paid $0.15 each for responding to the question shown in figure 8.

Out of the 100 responses, 68 were correct ("2582"), 22 were wrong but corresponded to likely successful attempts ("Blu2"). Those who responded "Blu2" were asked in a follow-up interaction what the PIN would be if they could only use digits, and all responded correctly. This amounts a total of 90% of who would have successfully entered the correct PIN. Of the remaining responses, one was a likely typo ("2182"); two were "1322" (the numerical string in the



What is Joe's PIN?

Joe uses a PIN to access his PayPal account from his phone. But he does not want to have to remember another number, and he does not want to reuse his banking PIN. So he uses PayPal's new "password to PIN" feature so that he only has to remember his password. Joe's password is "Blu2thrules". Look at the screen-shot below and let us know what PIN he should enter.

Joe's password is "Blu2thrules". A PIN is a four digit number, such as "1234". What is Joe's PIN? Please provide any comments you may have below, we appreciate your input!

Figure 8. The figure shows what subjects on Amazon Mechanical Turk are asked. Out of the 200 subjects who took the test, 68% responded with the expected answer, "2582" and another 22% with an answer ("Blu2") that indicates that they misunderstood the question but would have passed the authentication.

user name); two suggested that the subjects thought only letters mattered ("Blut"). Three believed "JoeS" (parts of the user name) was the correct response, and one cannot be explained. It is not clear what portion of these 10% of clearly unsuccessful logins would have occurred in a more realistic setting in which the users were more motivated to log in.

In a previous version of the experiment, the text on the screen read "Your PIN is the first four *letters* of your password" (emphasis added). In that version, 22 out of 33 respondents *still* responded with "2852". Two users answered "2882" and three "Blut" – which corresponds to 2882 but not mapped. Another two responded "Blu2".

VII. Conclusions

In this paper, we described a method of deriving PINs from passwords which is useful to obtain friction-free user onboarding to mobile platforms. We quantified exactly how much information about the passwords and the derived PINs contain, and how much information is lost based on real-life password distributions. We also assessed the usability of the proposed method using human-subject studies.

Acknowledgments

Many thanks to Lisa Kelly for helping us obtain access to raided dropboxes. Thanks to Bill Leddy for developing the iPhone app that was used in the qualitative experiments, and for providing advice on the presentation and flow. Also, many thanks to Jeff Hodges, Yang Liu, Alon Nir and Sid Stamm for providing helpful feedback on an earlier draft of the paper.

References

[1] N. S. Agency. SHA-2, US Certification: FIPS PUB 180-2, First published 2001.

[2] M. Barrett. Cybercrime – and what we will have to do if we want to get it under control, July, 2008.

[3] S. Berry. One in five use birthday as PIN number, Daily Telegraph, 27 Oct 2010.

[4] L. Carettoni, C. Merloni, and S. Zanero. Studying Bluetooth malware propagation: The BlueBag project. *IEEE Security and Privacy*, 5(2):17–25, 2007.

[5] A. Cser, J. Penn, P. Stamp, A. Herald, and A. Dill. Identity Management Market Forecast: 2007 To 2014: Provisioning Will Extend Its Dominance Of Market Revenues, February 6, 2008.

[6] D. A. F. Florêncio and C. Herley. A large-scale study of web password habits. In *WWW'07*, pages 657–666, 2007.

[7] D. A. F. Florêncio and C. Herley. Where do security policies come from? In *SOUPS'10*, pages –1–1, 2010.

[8] Georgia Tech Information Security Center. Emerging cyber threats report for 2009, October, 2008.

[9] S. Hansell. How hackers snatch real-time security ID numbers, August 20, 2009.

[10] S. Havlin. Phone infections. *Science*, 324(5930):1023–1024, 2009.

[11] M. Jakobsson and R. Akavipat. Rethinking passwords to adapt to constrained keyboards, www.fastword.me, Accessed 2010.

[12] M. Jakobsson and K.-A. Johansson. Retroactive Detection of Malware With Applications to Mobile Platforms. In *HotSec 2010*, Washington, DC, August 2010. USENIX, USENIX.

[13] M. Jakobsson, E. Shi, P. Golle, and R. Chow. Implicit authentication for mobile devices. In *HotSec'09: Proceedings of the 4th USENIX conference on Hot topics in security*, pages 9–9, Berkeley, CA, USA, 2009. USENIX Association.

[14] NIST Special Publication 800-90. Recommendation for Random Number Generation Using. Deterministic Random Bit Generators.

[15] J. Wiens. A tipping point for the trusted platform module?, June, 2008.

Appendix

We conducted an Internet survey on Amazon Mechanical Turk to study how people choose PINs. In the survey we wanted to estimate a first-order approximation of the entropy of a 4-digit PIN number. We recruited 100 subjects, all from the U.S., and only subjects with at least 95% positive feedback. They were paid \$0.26 each for answering one simple question: "How do you choose PINs to make them easy to remember?"

Some survey participants provide more than one answer. We finally collected 124 valid responses. These responses are further categorized into 4 groups: PINs generated using years, PINs generated using dates, PINs generated by keypad mapping, and PINs generated by other methods.

We list the definitions as well as the illustrative samples of each PIN generation method below.

1) **Using Years:** People tend to choose their PINs using the year during which something special happened. It could be someone's birth year or an anniversary. For example, one participant said "I normally use the years that my favorite movies come out. Say the movie hackers, came out in 1995. Then my pin would be 1995."
2) **Using Dates:** People tend to choose their PINs using the date when some special events occurred. It could be somebody's birthday or graduation date. For example, there is one participant who chose her PINs "by using loved ones birthdates (2 digit month, 2 digit day)."
3) **Using Keypad Mapping:** Many people choose PINs that spell out the first four letters of their hometown or someone's name. For example, one participant said "I choose my PIN as my son's name, Matt, and chose the corresponding number on a keypad."
4) **Others:** We consider all other PIN generation methods as one group of "other strategies". An example of a PIN in this group would be "I choose my PIN as the last four digits of an old phone number I used to have when I was a little kid."

We counted the number of responses and the percentages for each PIN generation method. We then compute estimates of the entropies of a 4-digit PIN using the different PIN generation methods. These are *upper bounds* since we do not know the distributions within each type, and therefore assume uniform distribution, which gives a higher entropy estimate than will actually occur.

For simplicity, we assume people can randomly choose a year between 1900 and 2010 as their PINs. This would give us an entropy value of $E_1 = log_2(110) = 6.6$ for any PIN chosen as a year. We also assume there are 365 days a year. People can randomly choose a month and a date to create a 4-digit PIN whose entropy value would be $E_2 = log_2(12 \times 30) = 8.5$. According to the previous section, a 4-digit PIN mapped from a password has an entropy value of $E_3 = 10$. We use this value for any PIN created by keypad mapping. This is based on the approximation that the mnemonics that are mapped will have a similar distribution as passwords. We treat all the other PINs which are not generated by years, dates, or keypad mapping as *uniformly random* PINs, which again, of course gives us an upper bound on the entropy. Those PINs have an entropy value of $E_4 = 4 \times log_2(10) = 13.3$.

The percentages and entropy estimates are presented in table I.

Table I
HOW TO CHOOSE PINS?

Method	Percentage	Entropy
Using Years	21%	6.6
Using Dates	27.4%	8.5
Keypad Mapping	12.9%	10
Others	38.7%	13.3

Therefore, a not-very-precise estimate of the average entropy for a 4-digit PIN is 10.15.