Expert Password Management

Elizabeth Stobert[1] and Robert Biddle[2]

[1] ETH Zürich
Zürich, Switzerland
elizabeth.stobert@inf.ethz.ch
[2] Carleton University
Ottawa, Canada
robert.biddle@carleton.ca

Abstract. Experts are often asked for advice about password management, but how do they manage their own passwords? We conducted interviews with researchers and practitioners in computer security, asking them about their password management behaviour. We conducted a thematic analysis of our data, and found that experts described a dichotomy of behaviour where they employed more secure behaviour on important accounts, but had similar practices to non-expert users on remaining accounts. Experts' greater situation awareness allowed them to more easily make informed decisions about security, and expert practices can suggest ways for non-experts to better manage passwords.

1 Introduction

Security experts are often turned to for advice about password management, but what do experts themselves do to manage their passwords? How are the practices of those who are knowledgeable about computer security different from or similar to those of non-experts?

Little work exists on the password habits of experts, who must be affected by the same problems that affect all users: difficulties choosing random passwords, difficulties remembering passwords, and multitudinous accounts. If remembering large numbers of random passwords is difficult or near-impossible for non-expert users, it should be similarly difficult for experts.

We conducted a series of interviews with researchers and practitioners in computer security, asking them about their password management behaviour. We found that these knowledgeable users described a dichotomy of behaviour where they employed more secure behaviour on important accounts that they deemed more worthy, but employed similar practices to non-expert users on their remaining accounts. The goal of our interviews was to better understand the practices of expert users, and to see how they address the demands of creating and managing large numbers of passwords. Do experts rely on similar coping strategies as non-experts? What kind of tools and techniques do they use? What differentiates experts from non-experts? We hoped to find insight from the practices and coping strategies of experts to help us form recommendations for non-experts.

In the following sections, we describe our study methodology and present our results. Our interviews yielded a set of descriptive quantitative data as well as a richer qualitative data set. We first present an overview of our interview results, before conducting a thematic analysis of experts' descriptions of their password management techniques. We identify four themes, and use these to better understand the ways in which experts differ from non-experts, as well as to form recommendations for non-expert users. We also identify areas of difficulty: password management problems that even expert knowledge cannot solve.

2 Background

Passwords pose a considerable usability challenge for end users, who are asked to create secure, unique passwords for every account, remember each of those passwords for a long time, and remember which password goes with which account for multiple accounts. These security requirements place demands beyond human capability on users' memory, time, and attention [15], and lead users to create passwords which are memorable, but easily guessed by attackers. This is known as the password problem [31]: passwords that are easy to remember are also easy to guess.

The password problem has existed for most of the history of computing. Morris and Thompson [22] describe the problem in a 1979 article about passwords in the UNIX operating system. However, with the introduction of personal computing and the web, the problem has scaled enormously. Current research continues to find users creating weak passwords [4], and instances of leaked or stolen passwords leading to major losses are increasingly common [8]. The password problem results from a mismatch between security expectations and users' abilities [31], and these disconnects can lead to the misuse or avoidance of security mechanisms [1]. Users may avoid password expectations by writing passwords down, or by reusing the same passwords across multiple accounts [27].

Conventional wisdom concludes that users are lazy and unwilling to comply with security advice. Correspondingly, the conventional suggestion is that users should be motivated to try harder to follow security advice, and be better educated about the dangers of poor security practices. However, the quantity of information that users are expected to remember is arguably impossible for users to memorize [15]. Users often end up ignoring security advice, and Herley [19] argues that these decisions are rational. Not only are password expectations impossible for users to meet, but a cost-benefit analysis of following security advice suggests that users should not even try [19].

2.1 Coping Strategies

Reusing Passwords One technique for coping with the demands of multiple passwords and accounts is to reuse passwords across multiple accounts. Reusing passwords carries security risks because an attacker may be able to uncover a

password for one website and then use that password to attack a user's other accounts (e.g., through the leak of a password database). In spite of these risks, almost all studies of password use have uncovered password reuse [14, 16, 18, 24, 25, 32]. Notoatmodjo [24] found that reuse increased with the number of users' accounts, and that most users cited increased memorability as the reason for reusing passwords. Reuse is a simple and intuitive coping technique that scales well to handling password meters [9], and coping with password policies [25].

Empirically tracing the extent of password reuse can be difficult. Das et al. [7] examined leaked datasets from 10 websites, and found that 43% of all passwords in their data set were reused across multiple accounts. They showed that knowledge of password reuse (via cross-referenced usernames) can be leveraged for more efficient password attacks.

Even when users do not completely reuse passwords, they often reuse pieces of passwords, or make minor modifications when using a password on another website. Most transformations take place at the beginning or end of a password, and the most common transformations are to add a number, symbol, or capitalization to comply with a new password policy [7, 30]. Users often retain fragments of existing habits and passwords across the creation of new accounts and changes in policy, leading to long-term reuse [28].

Writing Passwords Down Another coping strategy that users adopt for remembering passwords is to write passwords down. Writing passwords down can allow users to select and remember more complex passwords, as well as a higher number of passwords, but can have security risks if an attacker were to discover the list of recorded passwords.

Many users write down some or all of their passwords. Zviran and Haga [32] asked users about their password recording practices, and found that 35% of their participants wrote down their passwords, and the most common storage locations for recorded passwords were wallets, notebooks, and calendars. No relationship was found between password characteristics (length, composition) and likelihood of password recording, but participants were significantly more likely to write down passwords that were difficult to remember, or used infrequently. Grawemeyer and Johnson [17] found that writing passwords down was a coping strategy used to complement password reuse. They found that users were almost 18 times more likely to record unique passwords than reused passwords.

An important issue for recorded passwords is how they are stored: if securely stored, writing passwords down can be a perfectly acceptable technique for aiding users with passwords. Shay et al. [25] asked users how they protected their recorded passwords and found that about 30% of people did not protect them at all. Of the remaining 70%, strategies were varied, but included hiding the list of passwords, or storing it on another computer or device with a password.

2.2 Security Practices of Experts and Non-Experts

Quite a lot of work has focused explicitly on non-expert users. Wash [29] investigated non-experts' mental models of security, and found that users have often

inaccurate folk models of viruses and hackers that affect how users perceive and react to threats. Wash theorized that botnets behave in ways unanticipated by users' mental models, allowing botnets to propagate unnoticed. Stobert and Biddle [27] interviewed non-expert users about how they create, keep track of, and remember passwords. They found that users' passwords move through a life cycle where they are created, reused, and adapted into subsequent passwords.

Work comparing experts with non-experts has generally found that experts focus on different parts of the problem than non-experts. Asgharpour, Liu and Camp [3] had experts and non-experts participate in a card-sorting experiment to elicit mental models of security. They found that expert users' mental models of security differed from those of non-experts, and that a physical security metaphor was likely to be useful for framing computer security messages. Kang et al. [21] investigated users' mental models of the internet and examined perceptions of security and privacy online. They distinguished lay participants from technical participants and found that technical participants actually took fewer steps to protect themselves online. Their results showed that both lay and technical participants suffered from high levels of uncertainty around how information is collected and shared online. Although technical participants had different concerns than the lay participants, all were somewhat affected by not knowing how to handle the problems.

Ion, Reeder, and Consolvo [20] examined the security practices of expert users in a survey-based study. They examined exclusive practices of experts vs. non-experts and found that experts were more likely than non-experts to install system updates, use two-factor authentication, and use a password manager to stay safe online. Experts were likely to mention "unique" passwords and the use of password managers, while non-experts discussed "strong" passwords and password change policies. Non-experts were also more likely to say that they visited only known websites, changed passwords regularly, and used antivirus programs to stay protected from security threats. Norman [23] reports anecdotal evidence that experts reuse and record passwords to handle the difficulty of remembering secure passwords. He reported that many security professionals told him that they reused two passwords: a strong password and a weak password. For accounts with unusual password requirements, they reported writing passwords down.

3 Study

To investigate how computer security experts manage their passwords, we conducted a series of semi-structured interviews. We interviewed participants about a variety of subjects relating to password management, including creating, reusing, remembering, changing, and forgetting passwords. The interviews were conducted by the researcher, who asked questions and recorded responses. The interviews were also audio-recorded to facilitate further note-taking. We chose our methodology so that participants could reflect on and discuss not only what they do, but why they do it. We encouraged participants to elaborate on incomplete

answers and to pursue alternative discussion paths that revealed the details and complexity of their password management strategies. To allow comparison with earlier research on coping strategies and the password life cycle, we used the same interview script and elicitation techniques as in [27]. Participants were not given the interview questions in advance.

The interview had two parts: a short self-administered demographics questionnaire, and the password interview. The study was approved by the ethics committees at Carleton University and at ETH Zürich. We emphasized to participants that they should not share their passwords with us, and that all interview questions were optional. The interviews took approximately 30 minutes. Participants were not paid, but were happy to participate because of their interest in the topic.

We interviewed 15 expert users, recruited from the community of industry security practitioners and from among the information security research groups at ETH Zürich. Reflecting the gender distribution of the security community, the majority of our participants were male (13 participants). Participants ranged in age from 24 to 35, with a median age of 29. All participants except two had a graduate degree in computer security and all were employed as researchers, graduate students, or practitioners in information security.

Our interviews resulted in two datasets: a quantitative dataset of participants' specific responses to yes/no and quantitative questions, and a qualitative dataset of participants' explanations and detailed responses. During the interviews, we took detailed notes about each participant's responses. Later, we returned to the audio recordings to add detail to the notes taken during the interviews and to transcribe quotes. We summarize the results of our interviews in Section 4.

We conducted a thematic analysis of our qualitative data (Section 5), using methodology described by Braun and Clarke [5]. We chose thematic analysis for its flexibility and because it allowed us to explore the depth of our data and better understand the commonalities of participants' discussion and responses. We familiarized ourselves with our data by listening to the audio-recordings while reviewing and adding to the notes made during the interviews. We then began the process of open coding, where we identified ideas present in the data and assigned each idea a code. Next, we identified themes resulting from the coding process. We copied our codes onto post-it notes, and manipulated them on a whiteboard, where we could draw around them and use mind-map techniques to identify and refine themes. Finally, we considered our themes in relation to each other, and how they fit into the overall story of the data.

4 Results Overview

The expert participants in this study had a median of 64 accounts, and reported using a median of ten accounts in an average week. They reported wide ranging numbers of unique passwords, from 4 to 200, with a median of 58.

We were very clear that participants should not share their passwords with us, and experts were understandably private about their exact password creation strategies. Several participants mentioned algorithmic password creation strategies that integrated different pieces of information into passwords. All but one of these participants mentioned using this technique alongside reused passwords, and the remaining participant relied exclusively on this kind of algorithmic scheme. This participant had an elaborate password-generation algorithm that included a component related to the website, a random seed, and a personal evaluation of the required security level of the website.

Although participants did not discuss the exact components of their passwords, most participants said that their passwords were rarely rejected for failing to comply with password policies, indicating that these experts were including special characters, digits, and capital letters in their passwords.

> All the passwords have capital letters usually... it's more sometimes they say "okay, you have two strange characters", like unsupported special characters and you have to delete this, and it's a bit annoying. – E08

Although password reuse is a technique often criticized by security experts, the majority of our participants (12 out of 15) said that they reused passwords on at least some of their accounts. Of those participants who reported reusing passwords, all said they reuse multiple passwords. The median number of reused passwords was 3.5. Most participants described a careful strategy for reuse. Participants often mentioned they did not reuse all of their passwords, but that they had one or two passwords that they consistently reused for "throwaway" accounts. Participants mentioned reusing specific passwords for specific purposes, such as single-use websites, or seldom-visited websites.

> [Do you reuse multiple passwords?] Yep. [How many?] Four. Four different ones that have different behaviour in terms of complying to bullshit regulations like numbers, or punctuation, or ... – E03

Conversely, participants also described restricting password reuse for accounts.

> What I perceive as important, which is typically the four or five accounts that I use on a very regular basis, I use unique passwords for all of them. And I believe that these passwords are strong. But on the other hand, I use a common password for ... a lot of services that badger you to create an account at times. – E10

When discussing the kind of password that they reused, participants were clear that they had "their" password, often naming it (e.g., "my bootstrap password" – E14). Multiple participants referenced having had their password since they began using computers and one mentioned having had their password since high school.

We asked participants about how they stored passwords, and most (12 out of 15) reported storing their passwords in a computer program. Of these, six participants reported using a dedicated password manager, and the rest reported

storing their passwords in a web browser. Nine participants told us that they wrote their passwords down. Eight of these specified that writing their passwords down was something they did rarely, and only when unavoidable (e.g., in the case of an assigned password that they could not remember); the remaining participant treated his list as a kind of password manager, but also said one of the purposes of his list was to give to family members in case of emergency.

Several participants said that they relied on their password manager to generate passwords for accounts, but others said that they did not use this functionality (in spite of using a password manager to save passwords). Some participants described only generating random passwords for certain accounts, and most often said that they used this functionality for high-importance accounts.

Participants reported that they enter their passwords on a variety of device types, including smartphones, tablets, laptops and desktops, but most said that when creating passwords, they did not consider the entry device. Two participants mentioned shortening their passwords, or avoiding special characters when they knew they would be entering the password on a smartphone. One participant said that when using a regular keyboard, they tried to create their passwords so that all characters requiring the use of the "shift" key were next to each other.

Slightly less than half of the participants (7 out of 15) reported that they will enter their passwords on computers belonging to friends or family members, but most qualified the statement by mentioning that they would only log into certain accounts on other people's computers. Those who said that they would not enter their passwords on systems not managed by them said that this was a deliberate and strict policy for them.

5 Thematic Analysis

We began our thematic analysis with the process of open coding. We traversed the notes from our interviews, assigning codes to the data. To gain greater familiarity with the data, we relistened to the audio recordings of the interviews and took additional notes, which we then coded. We identified a total of 30 codes, and a list of all the codes used is included in Table 1.

Following the process of open coding, we began the process of identifying themes and relationships in the data. We identified four broad themes in our data, each of which answers some aspect of our research question: how do experts manage passwords?

5.1 Expert Awareness

During the interviews, it was clear that a key strategy for expert participants was to have consistent and pre-planned strategies. Experts were able to speak knowledgeably and fluently about their password management and security strategies. They were familiar with what they do to address security and often anticipated subsequent questions in the interview. While this familiarity is no doubt due to

Table 1. Complete list of all the open codes used in the analysis, organized by theme.

Code name	Description
Expert Awareness	
Consistency	Showed evidence of consistent habits (between accounts, or over time).
Algorithmic/deterministic	Generates passwords according to deterministic strategy or algorithm.
Exceptions to the rule	Discussed situations where consistency is damaged because of other factors.
Threat awareness	Shows awareness of specific security risks.
Family/friend trust	Describes special trust for friends or family.
Resets as coping strategy	Uses the password reset mechanism instead of remembering passwords.
Combining Strategies	
Variations on a theme	Describes creating passwords that are slight variations on each other.
Go-to password	Describes a particular password that is often reused.
Combination of strategies	Describes combining strategies.
Password manager as coping strategy	Uses password manager to cope with some difficulty of passwords.
Writes as backup	Writes passwords down as an insurance strategy (rather than to use often).
Records on paper	Writes passwords down on paper.
Records electronically	Writes passwords down in an electronic document (e.g., email, word document).
Personal Assessment of Risk	
Personal categorization	Organizes accounts by some "personal" strategy.
Security categorization	Organizes accounts by security.
Service-based categorization	Organizes accounts based on the website service.
Financial categorization	Organizes accounts based on money-based considerations.
Frequency-based categorization	Organizes accounts based on frequency of use (both frequent or infrequent).
Hidden category	Has accounts that belong to a category based on their non-dominant categorization.
Personal assessment of risk	Indicated that their assessment of risk was specifically applicable to themselves.
Usability Problems	
Usability problems	Describes usability problems.
Privacy	Describes privacy concerns.
Lack of control	Describes situation where control is lost.
Memory problems	Describes problems remembering information.
Password manager usability	Describes usability problems with password managers.
Username problems	Describes problems with usernames.
Limited online presence	Describes minimizing their online presence to avoid coping with security problems.
Broad online presence	Describes having many accounts.
Change of behaviour	Describes a situation where they changed their practices.
Self-dictionary attack	Guesses at own passwords.

the fact that these participants spend large amounts of their lives considering security, it also seemed to highlight the *a priori* nature of the expert approach. These participants referenced specific policies, and as in the following quote, were emphatic about avoiding certain situations.

> [Do you ever enter your passwords on computers that don't belong to you?] No. This is something I really try to avoid. – E08

Experts were specific about how they create and adapt passwords, and when asked the same question in different contexts, they often showed confusion about why the question was being asked again. Our interview asked about password creation when creating a new account vs. resetting a forgotten password, and at the second question, many participants gave us answers such as:

> [If you do have to reset a password because you don't remember it, how do you pick the new password?] Uh, I mean [it] is the same technique as I used before. – E02

Experts also showed awareness of specific threats in the interviews. When we asked about password changes, several experts referenced having changed their passwords in response to Heartbleed, a security bug in the OpenSSL library that necessitated widespread password changes [6].

> Well, there's, there's been a couple of incidents like, uhh, my laptop got stolen at one point, or... Or maybe you hear, like, a serious vulnerability like Heartbleed, and that's when you think that, that this might be a time to change passwords.– E07

The experts in our study sometimes mentioned planning for failure. Many participants reported using the password reset feature on a regular basis, and participants often planned to rely on this mechanism rather than going to the trouble of keeping track of an unusual password (e.g., one that deviated from their predictable password algorithms.) In these situations, participants were effectively planning on forgetting their password, relying on other existing mechanisms to save them. For accounts used infrequently, the trade-off of login time against convenience appeared to be worthwhile.

Planning for security can be made difficult by the myriad other pressures and unexpected situations that can arise, and experts did mention these situations that forced them to deviate from their preferred strategies. Among the situations described in the interviews were the pressures of friends and family, as well as unforeseen circumstances where information needed to be retrieved. The social and contextual pressures that affect everyone also affect computer security experts.

> I can be as paranoid as I want, but you know, in the real world I have a family and stuff, so sometimes you have to make compromises. – E15

5.2 Combining Strategies to Remember Passwords

Participants described a number of strategies for managing their passwords and accounts, and unexpectedly, many participants described using more than one technique, depending on the account.

Almost half of the participants said that they wrote some passwords down, and all of these described it as a kind of backup strategy. One participant said he wrote down passwords that were difficult or impossible to change. Another said that when he was issued assigned passwords, he often kept the piece of paper that came with the password (e.g., a letter with a PIN sent by the bank). One participant said that he wrote down most of his passwords, but was explicit about how his strategy was intended as a backup strategy for infrequently-used accounts.

> I just keep them written down just in case, and there are those more throwaway accounts that I use once every ... a few times a year, but I need then to check. – E04

Some participants described writing down other pieces of information as a backup strategy. One participant who had an algorithmic password generation strategy said that he sometimes wrote down the year that he had created the password for a specific account. Together with his memorized algorithm, this small piece of information was sufficient for him to regenerate the passwords.

Twelve participants described using some kind of password manager to save passwords. Six participants told us they used dedicated password managers, and eleven participants reported saving passwords in the web browser or in applications. Most participants mentioned using more than one tool, and even users of dedicated password managers reported using them alongside the browser-based managers.

Several participants described using a combination of strategies. In particular, multiple participants mentioned using password reuse in combination with password managers. One participant said that he used a password manager to randomly generate and remember passwords for important accounts, but that he opted to reuse passwords instead of storing them in the password manager for insignificant accounts.

> I don't store everything in a password manager. [Why not?] Because I, I dunno, because that's kind of incon... It's just another layer of inconvenience to use a password manager, and I, for me personally, it's not worth the investment to store it there. And it also kind of clogs my database, I guess, if I would store it in there, the password manager. – E01

In this quote, the participant describes the inconvenience of the password manager. Although he uses the manager, he weighs the inconvenience of the password manager against the significance of the account before deciding if he will use the manager for that account. Another participant described the same technique, but said that he made his decision on whether the website collected financial information. Yet another participant described a kind of thresholding process for determining which accounts got added to his password manager:

> If [password resets] happens more often than, I don't know, a bunch of times, then I will just use 1Password to remember that password. – E09

5.3 A Personal Assessment of Risk

Experts often explicitly mentioned the personal assessment of risk that played a role in their password management and creation strategies. One of the problems of computer security is that it can be difficult to know how well an account is protected, and to what level an account needs protection. Even with the additional experience and knowledge that accompanies expertise, it is hard to know exactly how specific decisions and choices will affect the protection of an account. In the following quote, the participant corrects himself to clarify that his classification of his two passwords as secure is based on his own judgment:

> I have two passwords that are, um, that I *consider* to be more secure, and that I use for only few things, but yeah, I consider more valuable. – E04

This idea of personal assessments of security came up repeatedly in the interviews, often in the discussion of a categorization strategy for accounts. Participants remarked on a number of categorization factors, including money/financial information, service-based categorization, or simply "importance".

> The first ingredient is the security level of the service, that I personally think it falls into this category. So for Amazon I would identify the security level I think Amazon should have in my world and then this is the first ingredient of the password. – E05

These strategies were often vaguely defined, and experts sometimes acknowledged their own inconsistency.

> I actually buy train tickets with this [password], but, yeah, I am contradicting myself because buying a train ticket involves money but I don't really care! – E11

Experts were clear in the interviews that objective assessments of security are difficult to make, and almost every description of a password management strategy mentioned this in some way. Experts did not express hesitation or concern about these decisions, but they were quick to clarify that many of their security assessments were particular to them. Having the awareness and ability to make these decisions quickly and relatively accurately is a hallmark of expert password management.

5.4 Usability Problems

Even though passwords are presumably a subject of interest for people employed as security experts, our participants still described difficulty and frustration with password management. One participant described assigned random passwords as "ridiculous string[s] of horror"(E03). Participants described a number of ways

in which they anticipated and experienced usability problems with passwords. Several participants said that they did not expect to remember passwords that were modified to comply with unusual password policies, and one participant described problems remembering the usernames associated with passwords.

One participant gave a long description of usability problems resulting from an unusual password policy and his reliance on the password reset mechanism.

> If all of them reject the password policy, then I would take the simplest one and do minimum compliance to make it fit their policy, and then any time I ever want to use it again, I wouldn't remember it, because of this, and if they told me this was their policy I would remember but as it is I wouldn't have any clue and I would get angry and frustrated and say "remind me my password" and they would send it to me and I'd be like "oh right, I forgot about this silliness." Or actually, no, what would happen is they would say "ok, reset your password" and then I would click the reset password and I would try to enter the simple password, it would reject it and explain the policy, and then I would remember what it was, the old one, but it was already too late because I had said reset the password and need to enter a new one. Yeah.– E03

This quote describes not only anger and frustration, but the additional time and effort that result from invisible password policies. In this description, the user enters multiple known passwords, creates a new password, revisits the website, logs into his email, clicks a reset link, and chooses another new password. This is a lot of work to log into an account!

Only a few expert participants described changes of behaviour related to security, but when they did, changes related to usability problems rather than to security concerns. One participant said they started using a password manager when they could not remember all of their passwords, but another participant said that they had stopped using a password manager because the built-in browser password manager filled their needs.

> [Do you use any kind of dedicated password manager?] Not at the moment, no. [Have you in the past?] I have tried. [What didn't work out?] Ummm. I guess I would say that in the situation as now, my browser remembers my passwords and that's somehow sufficient for me. My mobile remembers my passwords, so I, at the moment, I don't really feel the need for a separate password manager. – E07

A few participants described making efforts to minimize their online presence to avoid dealing with security and passwords. One participant told us how he avoided creating and managing passwords by relying on his spouse:

> I try to shove off all my passwords to let [my partner] manage it. – E10

The usability problems of passwords also lead experts to make mistakes: experts mentioned a number of practices with obvious security vulnerabilities. Since experts are presumably aware of these weaknesses, it is telling that they

have chosen to trade off security for usability in certain situations. Two participants said that they sometimes created passwords using dictionary words from their non-English mother tongue. Dictionary words in any language are easy for an attacker to guess.

> I sometimes do pick words from my native language because they almost look like a garbled set of characters in English, and then it's highly unlikely that somebody gets it. – E10

Another insecure practice mentioned by experts was guessing at their passwords. If an attacker is collecting password entries, guessing multiple passwords can quickly leak many passwords to an attacker. More than one participant referenced this technique, though most did clarify that they would only turn to it for low-value accounts.

> Since those belong mostly to throwaway accounts, I will just try another variation or try another one of my standard set of passwords. – E01

6 Discussion

Experts make use of many of the same coping strategies that are well-documented for non-experts. They reuse passwords, write passwords down, and create new passwords by making slight variations of older passwords. However, they combine these possibly insecure strategies with more careful habits for accounts where they are strongly concerned about security. One way they accomplish this is by using a password manager to generate and store passwords for high-value accounts, while reusing old passwords across other, lower-value, accounts.

The segmentation of strategies and clear division between important and unimportant accounts is what distinguishes expert behaviour from non-expert behaviour. Experts carefully plan to treat certain accounts more carefully than others. However, other studies [27, 20] have shown that non-experts try to use similar strategies. What allows experts to be more successful than non-experts?

Defining expertise is problematic, but it is usually agreed that an expert is someone with high knowledge in a certain domain and who is successful in that domain [12]. For example, an expert in chess is someone who is deeply familiar with the rules and strategy of the game, and is able to use this knowledge win many of their games. However the notion of success is less clear in personal practice with passwords. How exactly can it be shown that someone is more successful at managing their passwords than another person? How can we know that a lack of security breaches is due to good management and not due to luck?

In "ill-structured problems" [26] such as computer security, Endsley [11] argues that expertise comes from skilled decision making, which is enabled by *situation awareness*. Situation awareness is "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future."– [10, p.97]. Along with specialized skills and high knowledge in a domain, strong situation awareness contributes to expertise.

Experts with high situation awareness have learned knowledge and skills, schemas for prototypical situations, mental models of the domain, and automatic processes in the domain [11]. In our interviews, experts demonstrated all of these characteristics. They had high knowledge of the security domain and awareness of specific threats. They recognized the kind of password-related scenarios they had encountered in the past, and remembered their behaviour in those situations. They had mental models of threats and defences for those threats, as well as for which accounts were susceptible to which threats. Finally, the experts in our study had clear and automatic processes for how to create, remember, and reuse passwords in prototypical situations.

Although someone may have good situation awareness and expertise in one area of their domain, they may not maintain that awareness and expertise when handling novel situations [11]. Most of the password management scenarios discussed in our interviews were fairly routine, but participants repeatedly mentioned the frustration of situations where (for example) an unusual password policy forced them to change their password creation algorithm. In these situations, experts are no longer experts because they have lost some of their situation awareness.

6.1 What Do Experts Do Right?

The purpose of the expert interviews was to better understand how experts are managing passwords, but also to see what can be learned from the practices of experts and adapted to help non-expert users manage their passwords.

Together with other studies [27, 20], our interviews suggest that both experts and non-experts treat accounts with different requirements differently, but that the experts' consistency gives them an advantage in managing passwords. The experts in our study used password managers in combination with password reuse and other less secure coping strategies. They acknowledged the additional effort of using a password manager, but had selected the accounts where this effort was worthwhile. By using the password manager only on those accounts, they were effectively budgeting their time and effort to protect their most valuable accounts. Experts' additional situation awareness of security allowed them to make effective decisions about where to place their time and energy, and how to prioritize good security practices for important accounts.

Many of the habits and behaviours described by experts are accessible to anyone. Experts mostly mentioned using existing tools (open source and commercially available password managers) that are easily available online. We cannot expect that every user will be able to create a robust password generation algorithm, but many of the expert behaviours were similar to the practices of non-experts, and the additional software they used was available to anyone. But how we can help non-experts develop situation awareness to help them make informed decisions about security?

One way in which we can increase situation awareness for end users is to make security policies as transparent as possible. The potential presence of an attacker complicates this, but often-seen strategies such as obscuring the password policy

do little to discourage attackers, while complicating the situation for end users. Presenting information such as password rules and policies at password creation, and making log information about the time and location of logins available to users could potentially help them better manage their accounts.

Helping users develop schemas and good mental models for security is more difficult. Security is a secondary task and users are typically uninterested in the topic. Security and password management tasks are also distributed across many websites and accounts, with no central place through which to monitor them. Password managers create a central place through which passwords are created, saved, and monitored, and this could potentially help end users' situation awareness of their own passwords. Password managers provide users with a list of all their passwords, so users can see where they are reusing passwords, and in the case of a known vulnerability, make it easier for users to change affected passwords. Password managers could also help users by bringing vulnerabilities and compromises to users' attention. Some commercial password managers already do this: 1Password provides a service called Watchtower that allows users to identify services that are vulnerable to Heartbleed [2], and LastPass has a "Security Audit" feature that identifies passwords that occur in leaked datasets [13].

We suggest that end users should be able to develop consistent strategies to strongly protect the accounts they care about most, while not wasting effort on other accounts. The process of setting up a password manager can be daunting, but by selecting a small set of accounts for initial setup, the task is made significantly smaller. For example, users could select three important accounts, install a password manager, and add those accounts to the manager. Instead of attempting to solve their whole password problem, users should focus on the accounts that matter most to them. This incremental approach is scaleable, and it is possible that once the password manager is set up and in use, the user may want to use it for other accounts.

6.2 What Do Experts Do Wrong?

Though the habits and knowledge of experts can help address some of the issues with passwords, other problems still remain. Here, we highlight a few issues that were identified as problems during the interviews.

Password changes were a source of tension for most users. Experts were more likely to say that they changed their passwords than non-experts [27], but most experts said they changed their passwords only rarely and commented on the difficulty of the process. Participants commented on how the password change process can look different for every website, and finding the correct page and going through the password change process can be time-consuming. Password changes also affect the usability of password managers. To take advantage of the random password generation functions in most managers, passwords for existing accounts must be changed. This process can discourage all kinds of users from adopting password managers. Making password changes simpler could both encourage the adoption of password managers and improve security for users who want to change their passwords. A lurking theme in our interviews was the

usability of password managers. Although none of the experts in our study complained about the usability of their password managers, their unwillingness to make them their default password management strategy seems to point to some kind of issue with their usability or usefulness. Since their usefulness is evident, the issue is likely usability. Since there were no specific complaints about particular managers, the issue might simply be that password managers require an additional effort and a few extra clicks when logging into websites. Another possible issue here might be trust. A few participants did mention trust, and those that did had a personal rationale for why they did or did not choose to trust password managers. Interestingly, these personal rationales were not particularly similar to each other, and experts clearly put value on different parts of the security ecosystem.

7 Conclusion

Password management can be a struggle for everyone, even experts in computer security. Our interviews with experts about their password management habits showed that they use a combination of password management strategies to carefully allot appropriate security to individual accounts. Several experts relied on password reuse and other less secure coping strategies for lower-value accounts, but used a password manager to generate and remember random passwords for high-security accounts. Experts' increased situation awareness allowed them to more easily make informed decisions about their password management tasks.

The expert approach suggests that all users could improve their password management strategies by increasing their situation awareness of security. One way to do this might be to use a password manager for their most valued accounts. Explicitly identifying a small number of high-priority accounts is a natural extension of end users' existing strategies, and the comparatively small effort to better protect those accounts could significantly improve users' security. Additionally, this incremental approach could scale to protect more than just the most valuable accounts and could foster better password habits for all accounts.

Of course, expert knowledge does not solve all usability issues with passwords. Problem areas for password management include the usability of password managers and the ease of password changes. Although the expert approach cannot remedy all password management problems, it can suggest practical advice and strategies to help end users manage passwords in their daily life.

Motivations are complex, and it is difficult to know how individual biases and perspectives may affect our results. A limitation of interview studies is that we do not examine users' actual behaviour in the real world, and it can be difficult to know how factors such as reputation affect participants' responses. However, by probing responses and encouraging participants to thoughtfully examine and explain their comments, we hope that we have provided an initial perspective on the area, and that these results can be used to help inform security solutions for both end users and experts.

8 Acknowledgements

We would especially like to thank all of the computer security experts who lent their time, experience, and insight to our interviews. We also acknowledge support from the Natural Sciences and Engineering Research Council of Canada: Discovery Grant RGPIN 311982-2010.

References

1. A. Adams and M. A. Sasse. Users Are Not The Enemy. *Communications of the ACM*, 42(12):40–46, Dec. 1999.
2. AgileBits. 1Password Watchtower, 2015. `https://watchtower.agilebits.com`.
3. F. Asgharpour, D. Liu, and L. J. Camp. Mental Models of Security Risks. In *Financial Cryptography (FC)*, pages 367–377. Springer, 2007.
4. J. Bonneau. The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. In *Proceedings of the 33rd IEEE Symposium on Security and Privacy*, pages 538–552. IEEE, 2012.
5. V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, Jan. 2006.
6. Codenomicon. The Heartbleed Bug, Apr. 2014. `http://heartbleed.com`.
7. A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang. The Tangled Web of Password Reuse. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, Feb. 2014.
8. eBay. eBay Inc. To Ask eBay Users To Change Passwords, May 2014. `http://www.ebayinc.com/in_the_news/story/ebay-inc-ask-ebay-users-change-passwords`.
9. S. Egelman, A. Sotirakopoulos, I. Muslukhov, K. Beznosov, and C. Herley. Does My Password Go Up to Eleven?: The Impact of Password Meters on Password Selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 2379–2388. ACM, 2013.
10. M. R. Endsley. Design and Evaluation for Situation Awareness Enhancement. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, pages 97–101, 1988.
11. M. R. Endsley. Expertise and Situational Awareness. In K. A. Ericsson, N. Charness, P. J. Feltovich, and R. R. Hoffman, editors, *The Cambridge Handbook of Expertise and Expert Performance*. Cambridge University Press, Cambridge, 2006.
12. K. A. Ericsson. An Introduction to the Cambridge Handbook of Expertise and Expert Performance. In *The Cambridge Handbook of Expertise and Expert Performance*, pages 3–20. Cambridge University Press, Cambridge, 2006.
13. J. Fitzpatrick. How to Run a Last Pass Security Audit (and Why It Can't Wait), Dec. 2013. `http://www.howtogeek.com/176038/how-to-run-a-last-pass-security-audit-and-why-it-cant-wait/`.
14. D. Florencio and C. Herley. A Large-Scale Study of Web Password Habits. In *International World Wide Web Conference (WWW)*. ACM, May 2007.
15. D. Florencio, C. Herley, and P. C. van Oorschot. Password Portfolios and the Finite-Effort User: Sustainably Managing Large Numbers of Accounts. In *Proceedings of the 23rd USENIX Security Symposium*. USENIX, Aug. 2014.
16. S. Gaw and E. W. Felten. Password Management Strategies for Online Accounts. In *Proceedings of the 2nd Symposium on Usable Privacy and Security (SOUPS)*. ACM, July 2006.

17. B. Grawemeyer and H. Johnson. Using and Managing Multiple Passwords: a Week to a View. *Interacting with Computers*, 23(3):256–267, May 2011.
18. E. Hayashi and J. Hong. A diary study of password usage in daily life. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, May 2011.
19. C. Herley. So Long, and No Thanks for the Externalities: The Rational Rejection of Security Advice by Users. In *Proceedings of the 2009 Workshop on New Security Paradigms (NSPW)*. ACM, Sept. 2009.
20. I. Ion, R. W. Reeder, and S. Consolvo. "...no one can hack my mind": Comparing Expert and Non-Expert Security Practices. In *Proceedings of the 11th Symposium on Usable Privacy and Security (SOUPS)*. USENIX, July 2015.
21. R. Kang, L. Dabbish, N. Fruchter, and S. Kiesler. "My Data Just Goes Everywhere:" User Mental Models of the Internet and Implications for Privacy and Security. In *Proceedings of the 11th Symposium on Usable Privacy and Security (SOUPS)*. USENIX, July 2015.
22. R. Morris and K. Thompson. Password Security: a Case History. *Communications of the ACM*, 22(11):594–597, Nov. 1979.
23. D. A. Norman. When Security Gets in the Way. *ACM SIGCSE Bulletin*, 16(6):60, Nov. 2009.
24. G. Notoatmodjo. Exploring the 'Weakest Link': A Study of Personal Password Security. Master's thesis, The University of Auckland, New Zealand, Nov. 2007.
25. R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. M. Mazurek, L. Bauer, N. Christin, and L. F. Cranor. Encountering Stronger Password Requirements: User Attitudes and Behaviors. In *Proceedings of the 6th Symposium on Usable Privacy and Security*. ACM, June 2010.
26. H. A. Simon. The Structure of Ill-Structured Problems. In *Models of Discovery*, pages 304–325. D. Reidel Publishing, Dordrecht, 1977.
27. E. Stobert and R. Biddle. The Password Life Cycle: User Behaviour in Managing Passwords. In *Proceedings of the 10th Symposium on Usable Privacy and Security (SOUPS)*. USENIX, July 2014.
28. E. von Zezschwitz, A. De Luca, and H. Hussmann. Survival of the Shortest: A Retrospective Analysis of Influencing Factors on Password Composition. In *Proceedings of the 14th International Conference on Human-Computer Interaction (INTERACT)*. Springer, July 2013.
29. R. Wash. Folk Models of Home Computer Security. In *Proceedings of the 6th Symposium on Usable Privacy and Security (SOUPS)*. ACM, July 2010.
30. M. Weir, S. Aggarwal, M. Collins, and H. Stern. Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*. ACM, Oct. 2010.
31. S. Wiedenbeck, J. Waters, J.-C. Birget, A. Brodskiy, and N. Memon. PassPoints: Design and Longitudinal Evaluation of a Graphical Password System. *International Journal of Human-Computer Studies*, 63(1-2):102–127, July 2005.
32. M. Zviran and W. J. Haga. Password Security: An Empirical Study. *Journal of Management Information Systems*, 15(4):161–185, 1999.

Catena Variants

Different Instantiations for an Extremely Flexible Password-Hashing Framework

Stefan Lucks and Jakob Wenzel

Bauhaus-Universität Weimar, Germany
{stefan.lucks, jakob.wenzel}@uni-weimar.de

Abstract. Catena is a password-scrambling framework characterized by its high flexibility. The user (defender) can simply adapt the underlying (cryptographic) primitives, the underlying memory-hard function, and the time (λ) and memory (garlic) parameters, to render it suitable for a wide range of applications. This enables Catena to maximize the defense against specific adversaries, their capabilities and goals, and to cope with a high variation of hardware and constraints on the side of the defender. Catena has obtained special recognition of the Password Hashing Competition (PHC), alongside of the winner Argon2.
In addition to the default instantiations presented in the PHC submission, we want to use this document to introduce further variants of Catena, or rather, further instantiations of the Catena framework. Our instantiations use different hash functions, and we evaluate their influence on the computational time and the throughput. Next, we discuss how instantiations of the memory-hard graph-based algorithm influence the computational time and resistance against low-memory attacks. Furthermore, we introduce possible extensions of Catena accommodating strong resistance against GPU- and ASIC-based adversaries, e.g., by providing sequential memory-hardness due to a data-dependent indexing function. At the end, we combine particular instantiations discussed so far to construct full-fledged variants of Catena for certain goals. Hence, this document can be seen as an additional guide to the PHC submission of Catena when considering its usage under certain restrictions.

Keywords: Catena, Instantiations, Password Hashing Competition

1 Introduction

Today, it is common wisdom to store a one-way hash of a password instead of the password itself. Nevertheless, due to the large number of leaked password hashes [23] and the fact that the quality of passwords did not increase significantly over the last years [9], the evolution of password hashing had to bring up a new class of algorithms which is able to thwart "modern" attack types, i.e., cache-timing attacks [5,14], garbage-collector attacks [12], and low-memory attacks [6,8]. Thus, in 2013, the Password Hashing Competition (PHC) [2]

was brought to life with the goal to find one or more suitable successors of PBKDF2 [21], `scrypt` [25], and `bcrypt` [28].

During the progress of the PHC, there was a development in the properties and security goals a password-hashing scheme should satisfy depending on its operational environment. A comparison of the PHC submissions regarding to their functional properties, security, and general properties can be found in [12]. More properties were considered in certain specification documents of the PHC submissions. Examples are the resistance against ASICs [11], the delegation feature [27], and attacks on iterative compression functions [6].

In this work, we focus on the password-scrambling framework Catena [15] which, alongside of Lyra2 [30], Makwa [27], yescrypt [26], and the winner Argon2 [6], is one of the specially recognized entries of the PHC [3]. Catena allows to simply replace the underlying (cryptographic) primitives, the graph-based layer structure, the order of function calls in the core function, and adapt its time- and memory-cost parameters. Further, based on the default instantiations, it provides resistance against cache-timing and garbage-collector attacks.

Here, we want to go far beyond the default instantiations of Catena by considering several configurations based on the choice of the underlying hash functions, the graph-based layer structure, and extensions to provide certain features/properties. We provide benchmarks of computational time and memory usage for each configuration and discuss the security of different graph-based algorithms in terms of low-memory attacks [8].

Since we want to guarantee a certain level of security for all instances presented here, we always instantiate the underlying hash function H with a cryptographic hash function. Therefore, we fix H to BLAKE2b [4] and consider different instantiations only for the possibly reduced hash function H'. Obviously, a reduced hash function for H' can be deployed for performance reasons making it possible to increase the memory usage. Nevertheless, we advice against to use any variant of Catena for key derivation where H' is not a cryptographic hash function, since then it may be impossible to ensure suitable random-oracle security (which is a requirement for a key-derivation function (KDF)).

Outline. In Section 2, we present a slightly generalized variant of the Catena password-hashing framework and introduce all necessary notions which are used throughout this work. In Section 3, we consider different instantiations of the underlying hash function H'. In Section 4, we show and discuss certain graph-based algorithms. In Section 5, we introduce further extensions of the underlying structure of Catena. In Section 6, we first discuss the goals which we want to achieve and second, combine certain instantiations discussed in the former sections to build variants of Catena fulfilling these goals. Section 7 concludes our work.

2 Preliminaries

For clarity and to ease the understanding of the paper, we borrow and extend the notions from [15] (see Table 1). Note that for simplicity, we refer to n when

talking in general about the output size of the underlying hash function and to k for the output size of a particular instance of H'. Further, we restate the formal definition of the Catena framework which is used as a basis for our discussed instantiations (see Algorithm 1). Additionally, the definitions of (λ)-memory-hardness and (weak) garbage-collector attacks, as given in [15], can be found in Appendix A.

2.1 Notational Conventions

Identifier	Description
pwd	password
s	salt (public random value)
λ	security parameter of F (depth)
t	tweak
γ	public input (e.g., salt)
μ	secret input (password-dependent value)
$g_{\text{low}}, g_{\text{high}}$	minimum garlic; current garlic with $G = 2^{g_{\text{high}}}$
H	underlying cryptographic hash function
H'	arbitrary hash function (e.g., reduced version of H)
m	output length of Catena
n	output length of H
k	output length of H' (with $k \geq n$)
F	memory-hard function
Γ	function depending on the public input γ
Φ	function depending on the secret input μ
ρ	indexing function used to determine a certain graph instance
BRG_λ^g	(g, λ)-Bit-Reversal Graph
SBRG_λ^g	Shifted (g, λ)-Bit-Reversal Graph
DBG_λ^g	(g, λ)-Double-Butterfly Graph
$\text{GR}\ell_\lambda^g$	(g, λ, ℓ)-Gray-Reverse Graph
CTA	resistance against cache-timing attacks
PCA	resistance against precomputation attacks (tradeoff attacks)
KDF	key-derivation function

Table 1. Notations used throughout this document.

2.2 Catena

First, the tweak t[1], the password pwd, and the salt s are processed by the underlying hash function H. Then, to provide resistance against weak garbage-

[1] Note that the tweak is given as the hash value of an instantiation-specific identifier, a mode, λ, the output size and the size of the salt in bit, and the associated data. Thus, the tweak is unique for each system and each user within a system.

Algorithm 1 Catena

Require: pwd : password, t : tweak, s : salt, g_{low}: min. garlic, g_{high}: garlic, m : output length, γ : public Input
Ensure: x : hash of the password
1: $x \leftarrow H(t \mid\mid pwd \mid\mid s)$
2: $x \leftarrow flap(\lceil g_{\text{low}}/2 \rceil, x, \gamma)$
3: **for** $g = g_{\text{low}}, \ldots, g_{\text{high}}$ **do**
4: $\quad x \leftarrow flap(g, x \mid\mid 0^*, \gamma)$
5: $\quad x \leftarrow H(g \mid\mid x)$
6: $\quad x \leftarrow truncate(x, m)$
7: **end for**
8: **return** x

collector attacks [12], the designers decided to apply a twofold strategy where (1) the password should be removed from memory after the first Step (Line 1 of Algorithm 1) and (2) the output of H is fed into a function called $flap$ (with reduced memory cost) to ensure that the value x (which is directly derived from pwd) is overwritten. The second step is useful in cases where a compiler ignores the deletion of a direct derivative of the password due to optimization. A formal definition of the function $flap$ can be found in Algorithm 2, which is a slightly adjusted variant of its original version presented in [15]. The adjustment was required to allow state words of arbitrary size, which will become useful if the underlying hash function H' is instantiated with a function processing state words of $k > n$ bit in size, e.g., the compression function CF of Argon2 (see Section B.1 in Appendix B). The function H_{init} (see Algorithm 2 (right)) processes an input x of n bit size and outputs two initial state words (v_{-2}, v_{-1}) of k bit each.

Algorithm 2 Adjusted Function $flap$ of Catena (left), H_{init} (right)

Require: g: garlic, x : value to hash, γ : public input
Ensure: x : intermediate hash value
1: $(v_{-2}, v_{-1}) \leftarrow H_{\text{init}}(x)$
2: **for** $i = 0, \ldots, 2^g - 1$ **do**
3: $\quad v_i \leftarrow H'(v_{i-1} \mid\mid v_{i-2})$
4: **end for**
5: $v \leftarrow \Gamma(g, v, \gamma)$
6: $x \leftarrow F(v)$
7: **return** x

Require: x : n-bit value to hash
Ensure: v_{-2}, v_{-1} : two k-bit outputs
1: $\ell = 2 \cdot k/n$
2: **for** $i = 0, \ldots, \ell - 1$ **do**
3: $\quad w_i \leftarrow H(i \mid\mid x)$
4: **end for**
5: $v_{-2} \leftarrow (w_0, \ldots, w_{\ell/2-1})$
6: $v_{-1} \leftarrow (w_{\ell/2}, \ldots, w_{\ell-1})$
7: **return** (v_{-2}, v_{-1})

The rest of $flap$ remains unchanged, i.e., it consists of a mandatory initialization layer which sequentially fills the whole state in memory for the first time. Then, it calls the (optional) function Γ and the (mandatory) function F, where Γ is called the random layer and F is a memory-hard function (usually based on an

underlying graph structure). The memory-accesses employed within Γ depend on the public input γ, where γ can be freely chosen; also $\gamma = s$ is possible. In Section 5, we provide an extension of Catena given by a call to an additional function Φ following F to provide sequential memory-hardness. An instantiation of Catena is characterized by the following user-chosen parameters:

(Cryptographic) Hash Function H. Can also be non-cryptographic with the cost of reducing security, i.e., non-cryptographic hash functions may not support preimage or collision resistance or pseudorandomness.

Memory-Hard Function F. Usually a graph-based structure. Depending on its choice, an instance of Catena provides memory-hardness, λ-memory-hardness, or sequential memory-hardness.

Time- and Memory-Cost Parameter. The values g_{low} (min. garlic), g_{high} (garlic), and λ (depth) influence the required memory and computational time. For simplicity, and as in all default instantiations of Catena, we set $g_{\text{low}} = g_{\text{high}}$. The parameter λ denotes the depth of the underlying graph-based algorithm denoted by F, e.g., the number of stacks of a bit-reversal graph (BRG^g_λ).

Extension to the Function $flap$. Since Catena is designed to be a framework, it is possible to plug-in arbitrary functions to the core $flap$. For example, and as already shown in the specification paper of Catena, one could add a layer/function Γ which uses randomly distributed memory accesses depending on a public input, to update the internal state.

Depending on the instance of the underlying (cryptographic) primitive and the graph-based structure, as well as on possible extensions of the core, an instance of Catena provides a certain level of security. All considered security properties and how they are achieved are briefly discussed below, where P denotes a password-hashing scheme.

Preimage Resistance. Given a value $h = P(pwd)$, it is infeasible to find pwd. Let m denote the entropy of the password source. Then, an adversary can always guess a (weak) password by trying out about 2^m password candidates, leading to a success probability of $q/2^m$ for q queries. Otherwise, following from Algorithm 1, an adversary has to find a preimage for H in Line 5, i.e., for a value $x \leftarrow H(g \mid\mid x)$. If H is a cryptographic hash function and the output of H is chosen large enough, e.g., 256 bit, this becomes infeasible.

Resistance against Garbage-Collector (GC) Attacks. Knowledge about the internal state lying in memory after the invocation of P should not significantly reduce the runtime of testing a password candidate. This can be guaranteed if the depth (λ) of the underlying graph-based structure is at least 2 or the extension Φ is used – implying that the internal state is overwritten at least twice.

Resistance against Weak Garbage-Collector (WGC) Attacks. A value directly derived from the password (or the password itself) must not lie in memory during a significant time of the invocation of P. To provide security against WGC attacks, the password should be removed from memory at an early stage of P and no value directly (efficiently) derived from it should be used in a late stage of P. Each instance of Catena provides the latter by Line 2 of Algorithm 1, whereas the former has to be guaranteed by the particular implementation.

Memory-Hardness. At least a quadratic time-memory tradeoff (standard memory-hardness) is guaranteed by all considered graph-based instantiations. In Sections 4 and 5 we also discuss variants of Catena which provide stronger forms, i.e., λ-memory-hardness and sequential memory-hardness.

Resistance against Cache-Timing Attacks (CTA). All memory accesses conducted during the invocation of P must not be data-dependent. Since we consider instantiations of Catena using the extension Φ (conducting a data-dependent indexing function), we cannot guarantee this feature in general.

Random-Oracle Security. The outputs of P must be indistinguishable from random values of the same size. This feature is required for the usage of Catena as a key-derivation function and highly depends on the instantiation of the underlying hash functions H and H'. More detailed, the specification of Catena only provides a decent security analysis when $H = H'$, where H is a cryptographic hash function. Thus, we cannot guarantee random-oracle security if $H \neq H'$.

3 Hash-Function Instantiations

Note that depending on the choice of H and H', an algorithm might be vulnerable to the iterative compression function based attacks introduced by Khovratovich et al. in [6]. In addition to the instantiations of H', we also discuss the function G of BlaMka (G_B) (presented in [30]) as a possible variant of the underlying permutation of such an instantiation. For simplicity, we denote by G_O, G_L, and G_B the G function as used in original BLAKE2b and in Lyra2 [30], respectively, whereas G_B is a variant of G_L extended by 32-bit multiplications. Due to space issues, we moved the explanation of the discussed instantiations to Appendix B.

Comparison. We provide performance values based on the instantiation of the underlying hash function H'. All measurements shown in Table 2 are done using the tool Catena-Variants written by Schmidt [16] and its extended version by Schilling [20]. Due to the fact that the compression function of Argon2 (CF) supports state words of $k \ggg n$ bit, we fix the memory usage to 128 MB and let the garlic g to be a variable. Moreover, we fixed $\lambda = 2$ (as recommended by the designers of Catena), we set the random layer (Γ) to the identity, and do not

use any extensions. All measurements were done using an Intel(R) Core(TM) i7-3930K CPU @ 3.20GHz. For SHA-512, we used the implementation from the OpenSSL version 1.0.2d (9th July 2015).

Fast HF (H')	**Garlic** (g)	**Time (s)**	**Throughput (GB/s)**
BLAKE2b	21	1.55	0.08
BLAKE2b-1 (G_O)	21	0.37	0.34
CF (G_B)	17	0.25	0.50
CF (G_L)	17	0.18	0.69
GF	21	0.64	0.19
MultHash	21	0.30	0.42
$\mathrm{P}_{\mathrm{compress}}$ (G_B)	21	0.42	0.29
$\mathrm{P}_{\mathrm{compress}}$ (G_L)	21	0.32	0.39
SHA-512	21	4.04	0.03

Table 2. Measurements of Catena-BRG (Catena-Dragonfly) compared with different hash functions H', where we fix the memory usage to 128 MB, $\lambda = 2$, Γ to the identity, and no extensions are used.

4 Using Different Graphs

In this section, we describe different instantiations of the underlying graph-based structure denoted by F in Algorithm 2. The proposed default instantiations of Catena already include two graph-based algorithms called (g, λ)-Bit-Reversal Graph and (g, λ)-Double-Butterfly Graph, where g denotes the logarithm of the memory required per level, and λ the number of levels, i.e., the depth. By elaborating the first generic time-memory tradeoff (TMTO) attack on password scramblers [7,8], Khovratovich et al. have shown that a (g, λ)-Bit-Reversal Graph provides only memory-hardness, whereas it was shown in [22] and restated in [15] that an algorithm based on a (g, λ)-Double-Butterfly Graph provides λ-memory-hardness (see [15] for a definition of λ-memory-hardness). Note that for a $(g, 1)$-Bit-Reversal Graph, there exists a tradeoff analysis in the parallel setting exploiting the fact that the maximum required memory is not occupied during the whole computation time. Thus, multiple computations of a $(g, 1)$-Bit-Reversal Graph in parallel can share some of the memory, leading to a reduced time-memory tradeoff of $O(G^{2/3})$ instead of $O(G^2)$ [1]. Nevertheless, this analysis only holds if $\lambda = 1$ and neither the random layer Γ nor the extension Φ is used.

For each instantiation of F, we require a password-independent memory-access pattern, i.e., the resistance against cache-timing attacks of an instance of Catena must not depend on the instantiation of F. In Section 5, we introduce an extension of Catena that provides a password-dependent memory-access pattern for the sake of sequential memory-hardness (see [25] for a definition).

For now, we disregard the property of sequential memory-hardness for all variants discussed below and focus on the precomputation attack method of Khovratovich et al. shown in [7] and [8]. Thus, we first provide a description of all four instantiations of F and at the end of the section, we discuss their impact on the precomputation attack by presenting penalties regarding to the number of recomputations which have to be done if less than $2^g \cdot n$ memory is available. In our analysis, we exclude the (g, λ)-Double-Butterfly Graph since it is proven to be λ-memory-hard and thus, the precomputation method is not applicable. Since all considered instantiations base on the same generic structure, we can provide a generic graph-based hashing scheme where the difference between the four variants is given by an individual indexing function $\rho(i)$ (see Algorithm 3 (left)). Moreover, to allow state words of arbitrary size, we also include a similar adjustment as shown for the function $flap$ in Algorithm 2. Therefore, the first state value v_0 (denoted by r_0 in Line 2) is now computed using the function H_{first} as defined in Algorithm 3 (right). It takes two inputs of k-bit size and outputs an updated state value r_0 of k bit.

Algorithm 3 Generic (g, λ)-Graph-Based Hashing Scheme (left), H_{first} (right)

Require: g: garlic, v : state array,
λ : depth
Ensure: x : output hash
1: **for** $j = 1, \ldots, \lambda$ **do**
2: $r_0 \leftarrow H_{\text{first}}(v_{2^g-1} \mid\mid v_0)$
3: **for** $i = 1, \ldots, 2^g - 1$ **do**
4: $r_i \leftarrow H'(r_{i-1} \mid\mid v_{\rho(i)})$
5: **end for**
6: $v \leftarrow r$
7: **end for**
8: **return** r_{2^g-1}

Require: v_{2^g-1}, v_0 : two k-bit state words
Ensure: r_0 : k-bit state word
1: $w_0 \leftarrow H(v_{2^g-1} \mid\mid v_0)$
2: $\ell = k/n$
3: **for** $i = 1, \ldots, \ell - 1$ **do**
4: $w_i \leftarrow H(i \mid\mid z)$
5: **end for**
6: $r_0 \leftarrow (w_0, \ldots, w_{\ell-1})$
7: **return** r_0

The input to such an instance is given by the garlic g determining the memory required to be able to compute the fastest algorithm for a graph, a state array v consisting of the current state (of size $2^g \cdot n$ bit) of the password-hashing scheme, and a value λ determining the depth of the graph structure, i.e., the number of invocations of the underlying graph-based hashing operation. The update process of a word v_i of the internal state always depends on its immediate predecessor v_{i-1} (see Line 4 of Algorithm 3 (left) or Line 2 if it is the left-most word of the state). Thus, all graphs follow a sequential nature, thwarting parallel computation of the whole graph.

4.1 (g, λ)-Bit-Reversal Graph

This graph is used in the default instantiation Catena-Dragonfly (see Definition 5.1 of [15]), where $\rho_{\text{BRG}^g_\lambda}(i)$ with $i = i_0, i_1, \ldots, i_{q-1}$ and $i_j \in \{0, 1\}, 0 \leq j \leq q - 1$, is defined by

$$\rho_{\mathrm{BRG}_\lambda^g}(i) = (i_{q-1}, \ldots, i_1, i_0).$$

Thus, the function $\rho_{\mathrm{BRG}_\lambda^g}(i)$ returns the bit reverse of an input i. A property of any BRG_λ^g that was supposed to be exploited in the TMTO attack in [7,8] is its period of 2, i.e., the permutation has a self-inverting structure leading to the same order of elements if applied twice:

$$(0,1,2,3) \overset{\mathrm{BRG}_1^2}{\longrightarrow} (0,2,1,3) \overset{\mathrm{BRG}_1^2}{\longrightarrow} (0,1,2,3).$$

An intuitive countermeasure was to search for variants of BRG_λ^g with an increased period, which we discuss in the next three parts of this section.

4.2 Shifted (g, λ)-Bit-Reversal Graph

An SBRG_λ^g can be adjusted by a constant c determining the period of the underlying permutation. Thus, after $2^g - c$ invocations of an SBRG_λ^g using the indexing function $\rho_{\mathrm{SBRG}_\lambda^g}(i)$ which is defined by

$$\rho_{\mathrm{SBRG}_\lambda^g}(i) = \left(\rho_{\mathrm{BRG}_\lambda^g}(i) + c\right) \bmod 2^g,$$

the original permutation is reached again. See an example with $c = 1$ for an SBRG_1^2:

$$(0,1,2,3) \overset{\mathrm{SBRG}_1^2}{\longrightarrow} (1,3,2,0) \overset{\mathrm{SBRG}_1^2}{\longrightarrow} (3,0,2,1). \overset{\mathrm{SBRG}_1^2}{\longrightarrow} (0,1,2,3).$$

Unfortunately, the SBRG_λ^g does not protect well against the considered tradeoff attack. For example, we have shown for $g = 12$, that even if we consider all possible shift constants c, the penalties do not exceed that of the (g, λ, ℓ)-Gray-Reverse Graph presented in the next section (see Table 3). Based on the structure of an $\mathrm{SBRG}_\lambda^{12}$, the penalties follow a cyclic property, i.e., it holds that $p_c = p_{c \bmod 16}$ for $0 \leq c < 2^{12}$, where p_c denotes the penalty for the shift constant c. Thus, Table 3 shows all possible penalties which exist for an $\mathrm{SBRG}_\lambda^{12}$ with $\lambda \in \{2, 3\}$.
It follows that the instantiation of F with an SBRG_λ^g does not significantly increase the resistance of Catena against the attacks presented by Khovratovich et al. Following from this, we took another approach into consideration (see Section 4.3).

4.3 (g, λ, ℓ)-Gray-Reverse Graph

The indexing function discussed here bases on the combination of $\rho_{\mathrm{BRG}_\lambda^g}$ and the Gray Code [17]. It was initially suggested by Harris on the PHC mailing list [19] to increase the period of the underlying graph-based structure using the following indexing function:

$$\rho_{\mathrm{GR}\ell_\lambda^g}(i) = \rho_{\mathrm{BRG}_\lambda^g}(i) \oplus \left(\rho_{\mathrm{BRG}_\lambda^g}\left(\bar{i}\right) \gg \lceil g/\ell \rceil\right),$$

c	**Penalty**	c	**Penalty**
0	22.00	8	28.31
1	25.03	9	28.78
2	25.50	10	29.25
3	25.96	11	29.71
4	26.43	12	30.18
5	26.90	13	30.65
6	27.37	14	31.12
7	27.84	15	31.59
$\mathrm{GR2}_2^{12}$	54.56	$\mathrm{GR3}_2^{12}$	72.25

c	**Penalty**	c	**Penalty**
0	47.00	8	66.95
1	48.41	9	70.04
2	50.76	10	73.22
3	53.23	11	76.52
4	55.80	12	79.80
5	58.47	13	83.30
6	61.26	14	86.88
7	63.97	15	65.87
$\mathrm{GR2}_2^{12}$	254.93	$\mathrm{GR3}_2^{12}$	223.52

Table 3. Relative costs (penalty) of computing an SBRG_2^{12} (left) and an SBRG_3^{12} (right) depending on the shift constant c, where $c = 0$ corresponds to a BRG_2^{12} and BRG_3^{12}, respectively. The penalties where computed for the case when an adversary has $\lambda \cdot 2^{g-\alpha} \cdot n$ memory available (with $\alpha = g/3$). For comparison, we added the recomputation costs for a $\mathrm{GR2}_2^{12}$ and a $\mathrm{GR3}_2^{12}$.

where $\ell \in \{2, 3\}$ and $\bar{i}$ denotes the bitwise inversion of i. Thus, in comparison to a BRG_λ^g, the period depends on the number of words of the internal state (see Table 4). The values of g in Table 4 are chosen to ease comparison, whereas $g \in \{18, 21\}$ are also used in the default instantiations of Catena.

Garlic (g)	**3**	**4**	**5**	**8**	**12**	**13**	**14**	**15**	**16**	**18**	**21**
$\mathrm{GR2}_\lambda^g$	3	6	6	6	6	6	6	6	6	6	6
$\mathrm{GR3}_\lambda^g$	7	6	21	42	14	42	42	14	42	14	14

Table 4. Period of a $\mathrm{GR2}_\lambda^g$ and a $\mathrm{GR3}_\lambda^g$ depending on the number of internal state words (2^g).

In Section 4.4 we will see that even the 6-cyclic property of a $\mathrm{GR2}_\lambda^g$ is strong in terms of significantly increasing the penalty of an adversary when considering the precomputation attack mentioned before. Moreover, the increased period of a $\mathrm{GR3}_\lambda^g$ does not have a significant influence on the penalty in comparison of a $\mathrm{GR2}_\lambda^g$. Thus, it is an intuitive assumption to say that the increased resistance against the tradeoff attacks of Khovratovich stems more from the general asymmetric structure of a (g, λ, ℓ)-Gray-Reverse Graph than from the increased period. This also explains the weak resistance of an SBRG_λ^g against the tradeoff attacks which were initially designed for a BRG_λ^g.

4.4 Tradeoff Resistance

In this section, we analyze the resistance of the presented graph instantiations against the precomputation method of Khovratovich et al. Therefore, we set the available memory which can be provided by an adversary to $\lambda \cdot 2^{g-\alpha} \cdot n$

(note that n is replaced by k if $k > n$), which delivers an optimal tradeoff for a BRG^g_λ when $\alpha = g/3$ (we also assume $\alpha = g/3$ for all penalty computations independently from the instantiation of F). Based on the available memory, we compute the relative cost (penalty) an adversary would suffer in comparison to the case when it has $2^g \cdot n$ memory available (which would be the case for a defender using Catena). First, we will consider graphs with a depth of 2 or 3, i.e., $\lambda \in \{2, 3\}$. Then, we add the optional random layer Γ (see Line 5 of Algorithm 2) and recompute the penalties to determine the impact of the random layer on the precomputation method. Note that we do not consider the SBRG^g_λ in this analysis due to its weak resistance against those attacks. Nevertheless, we add the BRG^g_λ for the sake of comparison to the original attacks.

Shifting Sampling Points. The precomputation method of Khovratovich et al. with an optimal tradeoff considers $2^{g-\alpha}, \alpha = g/3$, sampling points stored in each level of the underlying graph-based structure. Thus, an adversary is allowed to store $2^{2g/3}$ memory units per level, where one memory unit is given by one word of internal state v. The sampling points are placed on the internal state so that it consists of $2^{2g/3}$ segments of $2^{g/3}$ state words each (beginning with placing the first sampling point on the first state value v_0). First, we were interested in the fact whether shifting the sampling points by a constant amount (which could differ for each layer) would strengthen the resistance of an instance of F against the precomputation attack method. Thus, for all possible $(2^\alpha)^\lambda = 2^{\alpha \cdot \lambda}$ shift configurations of the sampling points (where the sampling points were still ordered with a distance of $2^{g/3}$ state words), we recomputed the penalty an adversary would have. The results can be seen in Table 10 in Appendix D, where we only considered the minimum and maximum values for the penalty. The values behind the penalties (in brackets) denote the shift constants which are applied to the particular layers, e.g., 67.61 $(4, 7)$ denotes that all sampling points in the first layer are shifted by four positions to the right, whereas the sampling points in the second layer are shifted by seven positions to the right. From Table 10, we conclude that shifting the sampling points on the internal state does not significantly favours an adversary in terms of a precomputation attack. Therefore, all following results are based on the same configuration of sampling points as used in the original attack [7,8]. Nevertheless, we leave the question open if an adversary could obtain a significant lower penalty when considering an optimal distribution of all $\lambda \cdot 2^{2g/3}$ sampling points over the whole graph.

Naive Recomputation vs. Precomputation Method. First, we briefly discuss the difference between the naive recomputation approach and the precomputation method, where the former serves as a base for the latter. The configuration of the sampling points is the same for both attack methods, i.e., an adversary stores $\lambda \cdot 2^{g-\alpha}$ sampling points over the whole graph with a constant distance of 2^α state words between each other. For the naive approach, the sampling points already determine the required memory for an adversary, whereas the precomputation method allows to use additional memory in each layer to

speed-up the recomputation. In Table 5, we show the memory requirement for both attacks depending on the garlic g and depth λ. Note that the values shown for the precomputation method are given by the maximum additional memory which is required within one layer. For comparison, we also include the memory requirement for one layer when conducting the naive recomputation approach which solely consists of the sampling points. For simplicity, we set $H = H'$ so that all state words are of n bit in size. Therefore, the values for the precomputation method given in Table 5 denote the number of state words which have to be stored (excluding the fixed sampling points).

Attack	**Graph**	**Garlic** (g)	
		18	21
Precomp.	BRG_2^g	$2^{11.98}$	$2^{13.98}$
	$\mathrm{GR2}_2^g$	$2^{14.95}$	$2^{16.98}$
	$\mathrm{GR3}_2^g$	$2^{17.95}$	$2^{20.98}$
Naive	**all**	2^{12}	2^{14}

Attack	**Graph**	**Garlic** (g)	
		18	21
Precomp.	BRG_3^g	$2^{11.98}$	$2^{11.98}$
	$\mathrm{GR2}_3^g$	$2^{17.95}$	$2^{19.98}$
	$\mathrm{GR3}_3^g$	$2^{17.95}$	$2^{20.98}$
Naive	**all**	2^{12}	2^{14}

Table 5. Memory requirement depending on the garlic g, the attack method, and the particular graph instance, where the values are given for $\lambda = 2$ (left) and $\lambda = 3$ (right). All values refer to the maximum memory required within one layer.

From Table 5, we can deduce that the precomputation method is highly optimized for the application to a BRG_λ^g. We observed that for $\lambda \in \{2, 3\}$, the instance given by a $\mathrm{GR3}_\lambda^g$ massively thwarts an adversary since its memory savings are negligible. For example, for $g = 18$ and $\lambda = 2$, it would have to store 2^{12} state words for the fixed sampling points plus $2^{17.95}$ additional state words, leading to a total amount of about $2^{17.97}$ state words. The same holds for a $\mathrm{GR2}_3^g$, whereas at least some memory is saved when $\lambda = 2$. For the sake of completeness, the relative costs (penalties) of a naive recomputation and a precomputation attack on the instances given above are shown in Table 6. Considering the naive approach, the penalties do not differ between the graph instances since the underlying structure is given by a permutation, i.e., all values from the previous layer are used to compute the current layer.

Attack	**Graph**	**Garlic** (g)	
		18	21
Precomp.	BRG_2^g	32.33	64.33
	$\mathrm{GR2}_2^g$	150.11	334.08
	$\mathrm{GR3}_2^g$	352.75	1387.33
Naive	**all**	352.75	1387.41

Attack	**Graph**	**Garlic** (g)	
		18	21
Precomp.	BRG_2^g	47.81	95.87
	$\mathrm{GR2}_2^g$	1051.77	2344.44
	$\mathrm{GR3}_2^g$	912.11	3666.58
Naive	**all**	5895.25	45411.00

Table 6. Penalties depending on the garlic g, the attack method, and the particular graph instance, where the values are given for $\lambda = 2$ (left) and $\lambda = 3$ (right).

Even with the enourmous extra costs in terms of memory, the application of the precomputation method to a $\mathrm{GR}\ell_\lambda^g$ leads to a significantly higher penalty in comparision to the same attack on a BRG_λ^g. Thus, we can conclude that the instances of F given by a $\mathrm{GR}\ell_\lambda^g$ provide a strong resistance against the TMTO attacks introduced by Khovratovich et al. Nevertheless, there might exist attacks which are specificially focused on such instances, hence, reducing the penalty of an adversary. At the moment, we are not aware of such attacks and leave it as an open research question.

5 Extensions

This section discusses two extensions to the generic core of Catena: (1) a password-independent random layer Γ increasing the resistance of an instance of Catena against ASIC-based adversaries and (2) a password-dependent random layer Φ adding sequential memory-hardness to an instance of Catena. Each extension works independently from the instantiations of the underlying hash functions H and H'. Due to space issues, we discuss both extensions in Appendix C.

6 Discussion and Recommendations

All variants presented and recommended in this section provide preimage resistance as well as resistance against WGC attacks. Resistance against GC attacks is only considered for variants of Catena which are used for password hashing since an adversary launching a GC attack on an instance of Catena must already have access to the RAM used by this particular instance. Thus, reading the internal state of an instance of Catena (required for launching a GC attack) is as easy as obtaining the key generated by this instance. Nevertheless, if a variant makes use of the extension Φ, it directly derives GC resistance since the internal state is at least overwritten twice: at least once during the call to F and one time during the call to Φ. Furthermore, we only consider instantiations of F which conduct a password-independent memory-access pattern. If one would like to use a variant of Catena providing sequential memory-hardness, we refer to the extension Φ (see Section 5). Be aware that, for the variants presented here, the indexing function R used within Φ simply returns the g least significant bit of its input. In Table 7, we list and compare all variants of Catena which we recommend for usage in real-world applications. We always assume $g_{\mathrm{low}} = g_{\mathrm{high}}$ and recommend variants of Catena based on the four goals discussed below, where the -Full versions imply that $H = H' = BLAKE2b$. For each goal, we provide an instance for password hashing and one for key-derivation, whereby we set the maximal runtime to about 0.5 s for the former and about 5 s for the latter.

Note that all four variants discussed below satisfy common security goals for password hashing and key derivation (the -FULL versions), i.e., preimage security, random-oracle security, resistance against (weak) garbage-collector attacks

(except Catena-Horsefly-Full), and at least memory-hardness. Thus, a variant of Catena should be deployed depending on the most probably occurring type of adversary.

ASIC Resistant (*Catena-Stonefly/Catena-Stonefly-Full*). For the application of password hashing, providing strong resistance against ASIC-based adversaries can be achieved (among other things) by a password-dependent memory-access pattern (providing sequential memory-hardness), a public-input-dependent memory-access pattern, and multiplication-hardening. The idea of the password-dependent memory-access pattern can be realized by the extension Φ introduced in Section 5. We highly recommend to place the function Φ at the end of $flap$, since then the advantage of an adversary launching a cache-timing attack (as described for `scrypt` in [13]) against Catena is negligible. We further recommend to use a function conducting a public-input-dependent memory-access at the start of $flap$. Then, an ASIC-based adversary would have to copy the whole state to and from an ASIC if it aims to compute the underlying memory-hard function F on an ASIC (which would be intuitive since it follows a password-independent memory-access pattern and thus, can be computed efficiently on an ASIC). Such a function is given by Γ, which was already introduced in the default instantiation of Catena, where the public input is given by the salt. The goal of multiplication-hardening can achieved by using the function G_B (see Algorithm 5 (right) in Appendix B.2) as the underlying permutation of H'. For performance reasons, we instantiate F with a BRG_2^{18} since we do not aim to provide full resistance against tradeoff attacks for this instance. Catena-Stonefly and Catena-Stonefly-Full can be seen as the Catena complements to the data-dependent variant of the PHC winner Argon2, i.e., Argon2d [6].

High Throughput (*Catena-Horsefly/Catena-Horsefly-Full*). Even if we aim for maximal performance and memory usage, each instance of Catena should still provide a certain level of security. Therefore, we decided against instantiating H' in our recommended variants with MultHash (untrusted, insecure) or the Galois-Field Multiplication (allows tradeoff attacks with significantly reduced memory). Thus, for H', we opted for the compression function CF of Argon2 using G_L. The underlying graph-based structure F is set to a BRG_1^{18} which provides, in comparison to a $\mathrm{GR3}_\lambda^g$ and a DBG_λ^g, the fastest graph-traversing operation. Be aware that $\lambda = 1$ provides high performance but on the other hand allows for GC attacks. Therefore, we refer to $\lambda = 2$ if GC attacks are likely to happen. The instance presented here would be also suitable for the application in a proof-of-space scenario based on a challenge-response protocol.

Tradeoff Resistance (*Catena-Mydasfly/Catena-Mydasfly-Full*). For a tradeoff-resistant instance of Catena we decided to use a DBG_2^{14} as instantiation of F. This decision is based on the fact that we want to provide strong resistance against tradeoff attacks without enabling an adversary to launch a

cache-timing attack on the "first part" of $flap$. To further increase the resistance against tradeoff attacks, we make use of the extension Φ, providing sequential memory-hardness. Since the focus of this instance does not lie in ASIC resistance, we favour CF with G_L over G_B as the underlying hash function H'. Catena-Mydasfly and Catena-Mydasfly-Full can be seen as the Catena complements to the data-independent variant of the PHC winner Argon2, i.e., Argon2i [6].

Hybrid (*Catena-Lanternfly/Catena-Lanternfly-Full*). Here we wanted to aim for the best performance while remaining suitable security against ASIC-based adversaries, tradeoff attacks, as well as cache-timing resistance. Therefore, in comparison to Catena-Stonefly, we disregard the extension Φ but keep the invocation of Γ to provide sufficient resistance against ASICs. We instantiate F by a $\mathrm{GR3}_2^{17}$ which maintains reasonable resistance against precomputation attacks while still providing acceptable performance (in comparison to a DBG_λ^g). The hash function H' is instantiated with CF of Argon2 using G_B, which leads to a good throughput in terms of memory while providing multiplication-hardening. Catena-Lanternfly and Catena-Lanternfly-Full can be seen as the Catena complements to the hybrid variant of the PHC winner Argon2, i.e., Argon2id [6].

7 Conclusion

The inspiration for this work bases on the high flexibility of the password-hashing framework Catena. We have shown and discussed several instantiations of the underlying hash function H' as well as of the graph-based structure F. Furthermore, we considered possible extensions of Catena, called Γ and Φ, whereby the function Φ was introduced in this work. Based on the presented instantiations, we extended the Catena portfolio by new essential variants called Catena-Stonefly, Catena-Horsefly, Catena-Mydasfly, and Catena-Lanternfly, which provide resistance against ASICs, high performance/memory, tradeoff resistance, and resistance against a hybrid approach, respectively. For each of these four variants, we further provide a version suitable for key derivation.

Name	F	H'	Memory	Time (s)	Memory-Hardness	Γ	Φ	CTA	PCA	GCA	KDF
Default Instantiations											
Catena-Dragonfly	BRG_2^{21}	BLAKE2b-1	128 MB	0.36	✓	✓	-	✓	-	✓	-
Catena-Dragonfly-Full	BRG_2^{18}	BLAKE2b	16 MB	0.19	✓	✓	-	✓	-	✓	-
Catena-Butterfly	DBG_4^{16}	BLAKE2b-1	4 MB	0.28	λ	✓	-	✓	✓	✓	-
Catena-Butterfly-Full	DBG_4^{14}	BLAKE2b	1 MB	0.38	λ	✓	-	✓	✓	✓	-
Catena-Dragonfly-Full	BRG_2^{22}	BLAKE2b	256 MB	3.15	✓	✓	-	✓	-	✓	✓
Catena-Butterfly-Full	DBG_4^{17}	BLAKE2b	8 MB	3.76	λ	✓	-	✓	✓	✓	✓
ASIC Resistant											
Catena-Stonefly	BRG_1^{18}	CF (G_B)	256 MB	0.51	sequential	✓	✓	-	-	✓	-
Catena-Stonefly-Full	BRG_1^{22}	BLAKE2b	256 MB	3.12	sequential	✓	✓	-	-	✓	✓
High Throughput											
Catena-Horsefly	BRG_1^{19}	CF (G_L)	512 MB	0.45	✓	-	-	✓	-	✓	-
Catena-Horsefly-Full	BRG_1^{23}	BLAKE2b	512 MB	3.66	✓	-	-	✓	-	-	✓
Tradeoff Resistant											
Catena-Mydasfly	DBG_2^{14}	CF (G_L)	128 MB	0.40	sequential	-	✓	-	✓	✓	-
Catena-Mydasfly-Full	DBG_2^{18}	BLAKE2b	256 MB	4.33	sequential	-	✓	-	✓	✓	✓
Hybrid											
Catena-Lanternfly	GR3_2^{17}	CF (G_B)	128 MB	0.35	✓	✓	-	✓	✓	✓	-
Catena-Lanternfly-Full	GR3_2^{22}	BLAKE2b	256 MB	3.65	✓	✓	-	✓	✓	✓	✓

Table 7. Comparison of variants of Catena. CTA – Resistance against cache-timing attacks, PCA – Resistance against precomputation attacks (tradeoff attacks), GCA – Resistance against GC attacks, KDF – Key-derivation function. All variants mentioned here also provide preimage resistance and WGCA resistance and use BLAKE2b as the underlying cryptographic hash function H. The function Γ for the here mentioned instances is always called right before F due to the given layer structure. All measurements where done using an Intel(R) Core(TM) i7-3930K CPU @ 3.20GHz.

8 Acknowledgement

We would like to thank S. Schmidt and H. Schilling for their work on the reference implementation of Catena as well as on the tool Catena-Variants, E. List for his helpful comments and fruitful discussions, and H. Schilling for his analysis of the underlying graph-based structures. Furthermore, we would like to thank the reviewers of the Passwords 2015 for their helpful comments.

References

1. Joël Alwen and Vladimir Serbinenko. High parallel complexity graphs and memory-hard functions. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 595–603, 2015.
2. Jean-Philippe Aumasson. Password Hashing Competition. `https://password-hashing.net/call.html`. Accessed September 3, 2015.
3. Jean-Philippe Aumasson. Password Hashing Competition – Candidates. https://password-hashing.net/candidates.html.
4. Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. BLAKE2: Simpler, Smaller, Fast as MD5. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS*, volume 7954 of *Lecture Notes in Computer Science*, pages 119–135. Springer, 2013.
5. Daniel J. Bernstein. Cache-timing attacks on AES, 2005.
6. Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Argon2. Password Hashing Competition, Winner, 2015. `https://www.cryptolux.org/index.php/Argon2`.
7. Alex Biryukov and Dmitry Khovratovich. Tradeoff cryptanalysis of Catena. PHC mailing list: discussions@password-hashing.net.
8. Alex Biryukov and Dmitry Khovratovich. Tradeoff Cryptanalysis of Memory-Hard Functions. *IACR Cryptology ePrint Archive*, 2015:227, 2015.
9. Joseph Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *2012 IEEE Symposium on Security and Privacy*, May 2012.
10. Richard P. Brent, Pierrick Gaudry, Emmanuel Thomé, and Paul Zimmermann. Faster Multiplication in GF(2)[x]. In *ANTS*, pages 153–166, 2008.
11. Bill Cox. TwoCats (and SkinnyCat): A Compute Time and Sequential Memory Hard Password Hashing Scheme. `https://password-hashing.net/submissions/specs/TwoCats-v0.pdf`, 2014.
12. Christian Forler, Eik List, Stefan Lucks, and Jakob Wenzel. Overview of the Candidates for the Password Hashing Competition - And their Resistance against Garbage-Collector Attacks. *IACR Cryptology ePrint Archive*, 2014:881, 2014.
13. Christian Forler, Stefan Lucks, and Jakob Wenzel. Catena: A Memory-Consuming Password Scrambler. Cryptology ePrint Archive, Report 2013/525, 2013. `http://eprint.iacr.org/`.
14. Christian Forler, Stefan Lucks, and Jakob Wenzel. Memory-Demanding Password Scrambling. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, pages 289–305, 2014.

15. Christian Forler, Stefan Lucks, and Jakob Wenzel. The Catena Password-Scrambling Framework. Password Hashing Competition, 2nd round submission, 2015. https://password-hashing.net/submissions/specs/Catena-v3.pdf.
16. funkysash. catena-variants. https://github.com/medsec/catena-variants, 2015.
17. Frank Gray. Pulse code communication, March 1953. US Patent 2,632,058.
18. Shay Gueron and Michael E. Kounavis. Intel Carry-Less Multiplication Instruction and its Usage for Computing the GCM Mode - Rev 2.01. Intel White Paper. Technical report, Intel corporation, September 2012.
19. Ben Harris. Replacement index function for data-independent schemes (Catena). http://article.gmane.org/gmane.comp.security.phc/2457/match=grey, 2015.
20. HPSchilling. catena-variants. https://github.com/HPSchilling/catena-variants, 2015.
21. B. Kaliski. RFC 2898 - PKCS #5: Password-Based Cryptography Specification Version 2.0. Technical report, IETF, 2000.
22. Thomas Lengauer and Robert Endre Tarjan. Asymptotically Tight Bounds on Time-Space Trade-offs in a Pebble Game. *J. ACM*, 29(4):1087–1130, 1982.
23. T. Alexander Lystad. Leaked Password Lists and Dictionaries - The Password Project. http://thepasswordproject.com/leaked_password_lists_and_dictionaries. Accessed May 16, 2013.
24. David A. McGrew and John Viega. The Security and Performance of the Galois/-Counter Mode (GCM) of Operation. In *INDOCRYPT, volume 3348 of LNCS*, pages 343–355. Springer, 2004.
25. Colin Percival. Stronger Key Derivation via Sequential Memory-Hard Functions. presented at BSDCan'09, May 2009, 2009.
26. Alexander Peslyak. yescrypt - a Password Hashing Competition submission. https://password-hashing.net/submissions/specs/yescrypt-v1.pdf, 2015.
27. Thomas Pornin. The MAKWA Password Hashing Function. https://password-hashing.net/submissions/specs/Makwa-v1.pdf, 2015.
28. Niels Provos and David Mazières. A Future-Adaptable Password Scheme. In *USENIX Annual Technical Conference, FREENIX Track*, pages 81–91. USENIX, 1999.
29. Mark Shand, Patrice Bertin, and Jean Vuillemin. Hardware Speedups in Long Integer Multiplication. In *SPAA*, pages 138–145, 1990.
30. Marcos Simplicio, Leonardo Almeida, Paulo dos Santos, and Paulo Barreto. The Lyra2 reference guide. Password Hashing Competition, 2nd round submission, 2015. https://password-hashing.net/submissions/specs/Lyra2-v2.pdf.
31. Peter Soderquist and Miriam Leeser. An Area/Performance Comparison of Subtractive and Multiplicative Divide/Square Root Implementations. In *12th Symposium on Computer Arithmetic (ARITH-12 '95), July 19-21, 1995, Bath, England, UK*, pages 132–139, 1995.
32. Bill Cox (waywardgeek). MultHash – A simple multiplication speed limited hash function. https://github.com/medsec/catena/blob/3a3ce823d4c54f2da33757bf8f6389488c31bd93/src/catena-multhash.c, 2014.

A Memory-Hardness and Garbage-Collector Attacks

A.1 Memory-Hardness

In this part of the paper we show the definition of memory-hardness as it was described in [15]. The intuition is that for any parallelized attack (using b cores) the required memory is decreased by a factor of $1/b$ per core, and vice versa.

Definition 1 (Memory-Hard Function). *Let g denote the memory cost factor. For all $\alpha > 0$, a memory-hard function f can be computed on a Random Access Machine using $S(g)$ space and $T(g)$ operations, where $S(g) \in \Omega(T(g)^{1-\alpha})$.*

Thus, for $S \cdot T = G^2$ with $G = 2^g$, using b cores, we have

$$\left(\frac{1}{b} \cdot S\right) \cdot (b \cdot T) = G^2.$$

A formal generalization of this notion is given in the following definition.

Definition 2 ($\lambda-$Memory-Hard Function). *Let g denote the memory cost factor. For a $\lambda-$memory-hard function f, which is computed on a Random Access Machine using $S(g)$ space and $T(g)$ operations with $G = 2^g$, it holds that*

$$T(g) = \Omega\left(\frac{G^{\lambda+1}}{S(g)^{\lambda}}\right).$$

Thus, we have

$$\left(\frac{1}{b} \cdot S^{\lambda}\right) \cdot (b \cdot T) = G^{\lambda+1}.$$

Since we also consider instantiations of Catena which fulfill the definition of sequential memory-hardness (see [25]), we highly recommend the reader to have a look at the discussion *λ-Memory-Hard vs. Sequential Memory-Hard* given in Section 2.2 of [15].

A.2 (Weak) Garbage-Collector Attacks

The motivation behind these attacks is to exploit the fact that a password scrambler may leave its password-dependent internal state in memory for a long time (during its invocation). An adversary then gains benefits if either the memory access to the internal state depends on the password (Garbage-Collector Attack) or if a value directly derived from the password (e.g., using a fast hash function) remains in memory (Weak Garbage-Collector Attack). Based on such attacks, an adversary is able to filter password candidates using significant less time/memory in comparison to the original algorithm.

Definition 3 (Garbage-Collector Attack). *Let $P_G(\cdot)$ be a memory-consuming password scrambler that depends on a memory-cost parameter G and let Q be a positive constant. Furthermore, let v denote the internal state of $P_G(\cdot)$ after its termination. Let $\mathcal{A}$ be a computationally unbounded but always-halting adversary conducting a garbage-collector attack. We say that $\mathcal{A}$ is successful if some knowledge about v reduces the runtime of $\mathcal{A}$ for testing a password candidate x from $\mathcal{O}(P_G(x))$ to $\mathcal{O}(f(x))$ with $\mathcal{O}(f(x)) \lll \mathcal{O}(P_G(x))/Q, \forall x \in \{0,1\}^*$.*

In the following we define the Weak Garbage-Collector (WGC) Attack.

Definition 4 (Weak Garbage-Collector Attack). *Let $P_G(\cdot)$ be a password scrambler that depends on a memory-cost parameter G, and let $R(\cdot)$ be an underlying function of $P_G(\cdot)$ that can be computed efficiently. We say that an adversary $\mathcal{A}$ is successful in terms of a weak garbage-collector attack if a value $y = R(pwd)$ remains in memory during (almost) the entire runtime of $P_G(pwd)$, where pwd denotes the secret input.*

An adversary that is capable of reading the internal memory of a password scrambler during its invocation gains knowledge about y. Thus, it can reduce the effort for filtering invalid password candidates by just computing $y' = R(x)$ and checking whether $y = y'$, where x denotes the current password candidate. Note that the function R can also be the identity function. Then, the plain password remains in memory, rendering WGC attacks trivial.

B Hash-Function Instantiations

B.1 Compression Function of Argon2

Here we consider the underlying compression function of the PHC winner Argon2 [6]. The function CF (within the specification of Argon2, this function is called G, but since we use G already for the variants of the round function of BLAKE2b, we rename it here to CF) is built upon a permutation P using the BLAKE2b round function G_L as defined in Lyra2 and is formally defined in Algorithm 4. Note that the function G_L, in comparison to the original one of BLAKE2b (G_O), does neither consider the message schedule nor handling of the message input. The input state words are directly written to the internal state (a, b, c, d) and then processed as shown in Algorithm 5 (left).

The function CF, using G_L, takes two 8192-bit values X, Y as input and outputs a 8192-bit value Z. The value $R = X \oplus Y$ (Line 1) is represented as an 8×8 matrix consisting of 64 128-bit values $R_0, \ldots, R_{63}$ ordered from top to down and left to right. First, all rows are updated using the permutation P generating an intermediate matrix Q (Lines 2-4). Then, all columns of Q are updated using P (Lines 5-7). The output of CF is then given by the XOR operation of the input matrix R and the intermediate matrix Q. Based on the large state of CF, it is possible to fill memory quite fast, e.g., about 1 GB memory is filled in less than 1 second [6]. For a formal definition of P we refer to the specification of Argon2 [6].

Algorithm 4 Compression Function CF of Argon2 [6]

Require: X, Y : two 8192-bit values fetched from memory
Ensure: Z : intermediate hash (8192 bit)
1: $R \leftarrow X \oplus Y$
2: **for** $i = 0, \ldots, 7$ **do**
3: $\quad (Q_i, \ldots, Q_{i+7}) \leftarrow P(R_i, \ldots, R_{i+7})$ ▷ updating the rows
4: **end for**
5: **for** $i = 0, \ldots, 7$ **do**
6: $\quad (Q_i, Q_{i+8}, \ldots, Q_{i+56}) \leftarrow P(R_i, R_{i+8} \ldots, R_{i+56})$ ▷ updating the columns
7: **end for**
8: $Z \leftarrow R \oplus Q$
9: **return** Z

Even if the function CF requires two calls to the underlying round function of BLAKE2b to process 1024 bit, it has a significant speed-up in comparison to the way BLAKE2b-1 is used within Catena (see Table 2). The speed-up comes from the fact that CF processes 16384 bit per invocation whereas BLAKE2b-1 processes only 1024 bit. Thus, Catena requires significant more loading operations in comparison to Argon2 when processing inputs of equal size.

Note that we provide another instantiation of H' in Section 3 called $\mathrm{P}_{\mathrm{compress}}$. This is a modification of the permutation P, where we split the output of P into

two 512-bit halves and combine them using the XOR operation. This is a simple way to build a compression function out of the permutation P. We introduce $\mathrm{P}_{\mathrm{compress}}$ for comparison with BLAKE2b-1, which includes the regular message schedule, the initialization, and finalization of the round function of BLAKE2b (as shown in Algorithm 6), whereas $\mathrm{P}_{\mathrm{compress}}$ does not. Nevertheless, we do not consider it as a recommended instantiation of H' since, due to the smaller size of the state words, it will never achieve the same throughput as the function CF.

B.2 BlaMka

The function G_B was introduced by the authors of Lyra2 [30]. It described a slightly adjusted variant of G_L (see Algorithm 5 for a comparison) and is there used as the underlying permutation of a sponge-based structure. Due to the missing cryptanalysis of BlaMka, the authors do not recommend its usage. Nevertheless, since it consists of a neat combination of ARX (Addition, Rotation, XOR) and integer multiplication, we also consider it as an alternative to G_L.

The only difference between the functions G_L (see Algorithm 5 (left)) and G_B (see Algorithm 5 (right)) is that each addition $a+b$ is replaced by a variant of the latin-square operation, namely $a+b+2\cdot lsw(a)\cdot lsw(b)$, where $lsw(x)$ denotes the least significant 32-bit word of x. The idea behind the so called multiplication.-hardening is to provide a similar runtime for hardware- and software-based adversaries [29,31].

Algorithm 5 Functions G_L (left) and G_B (right)

Require: a, b, c, d : 256-bit input state	**Require:** a, b, c, d : 256-bit input state
Ensure: a, b, c, d : 256-bit output state	**Ensure:** a, b, c, d : 256-bit output state
1: $a \leftarrow a + b$	1: $a \leftarrow a + b + 2 \cdot lsw(a) \cdot lsw(b)$
2: $d \leftarrow (d \oplus a) \ggg 32$	2: $d \leftarrow (d \oplus a) \ggg 32$
3: $c \leftarrow c + d$	3: $c \leftarrow c + d + 2 \cdot lsw(c) \cdot lsw(d)$
4: $b \leftarrow (b \oplus c) \ggg 24$	4: $b \leftarrow (b \oplus c) \ggg 24$
5: $a \leftarrow a + b$	5: $a \leftarrow a + b + 2 \cdot lsw(a) \cdot lsw(b)$
6: $d \leftarrow (d \oplus a) \ggg 16$	6: $d \leftarrow (d \oplus a) \ggg 16$
7: $c \leftarrow c + d$	7: $c \leftarrow c + d + 2 \cdot lsw(c) \cdot lsw(d)$
8: $b \leftarrow (b \oplus c) \ggg 63$	8: $b \leftarrow (b \oplus c) \ggg 63$

Note that for comparison, we restate BLAKE2b-1 as used in the default instantiations of Catena (see Algorithm 6). It conducts G_O ensuring that 12 invocations of BLAKE2b-1 are quite similar to one invocation of full BLAKE2b.

B.3 Galois-Field Multiplication

We took the Galois-Field multiplication into account since it allows to be used as a really fast compression function, especially when considering binary Galois Fields with an order that is a power of 2, since its coefficients can be represented

Algorithm 6 BLAKE2b-1 (left) and its Function compress (right).

Require: I_1, I_2 : two 512-bit input words, v : Vertex index, S : global BLAKE2b state
Ensure: x : intermediate hash
1: $S.buf \leftarrow I_1 \mid\mid I_2$
2: $S.buflen \leftarrow 128$
3: $increment\_counter(S, S.buflen)$
4: $set\_last\_block(S)$
5: $r \leftarrow v \mod 12$
6: $compress(S, r)$
7: **return** $S.h$

Require: S : Blake2b state, r : Round index, IV : global initialization vector, σ : Message Schedule
1: $v[0 \dots 7] \leftarrow S.h$
2: $v[8 \dots 15] \leftarrow IV$
3: $v[12, 13] \leftarrow v[12, 13] \oplus S.t$
4: $v[14, 15] \leftarrow v[14, 15] \oplus S.f$
5: $s[0 \dots 15] := \sigma[r \mod 10][0 \dots 15]$
6: $v \leftarrow G_O(v, 0, 4, 8, 12, S.buf[s[0]], S.buf[s[1]])$
7: $v \leftarrow G_O(v, 1, 5, 9, 13, S.buf[s[2]], S.buf[s[3]])$
8: $v \leftarrow G_O(v, 2, 6, 10, 14, S.buf[s[4]], S.buf[s[5]])$
9: $v \leftarrow G_O(v, 3, 7, 11, 15, S.buf[s[6]], S.buf[s[7]])$
10: $v \leftarrow G_O(v, 0, 5, 10, 15, S.buf[s[8]], S.buf[s[9]])$
11: $v \leftarrow G_O(v, 1, 6, 11, 12, S.buf[s[10]], S.buf[s[11]])$
12: $v \leftarrow G_O(v, 2, 7, 8, 13, S.buf[s[12]], S.buf[s[13]])$
13: $v \leftarrow G_O(v, 3, 4, 9, 14, S.buf[s[14]], S.buf[s[15]])$
14: $S.h \leftarrow S.h \oplus v[0 \dots 7] \oplus v[8 \dots 15]$

by bits and addition/subtraction is equivalent to the XOR operation. We decided to conduct the multiplication in $GF(2^{128})$ using the reduction polynom $f(x) = x^{128}+x^7+x^2+x+1$ from the well-analyzed and widely used Galois/Counter Mode (GCM) [24]. We provide the optimized version of the right-to-left multiplication as suggested in [10] or, if available, a version that uses the pclmulqdq instruction described in [18]. Note that in contrast to the multiplication used within GCM, we do not use bit reflection.

B.4 MultHash

The function MultHash is an experimental hash-function design by Cox [32] with the goal to max out the memory bandwidth. It processes two ℓ-bit inputs A and B by splitting them into $\ell/32$-bit chunks and then conducts $\ell/32$ 32-bit multiplications of the form $C_i = (C_{i-1} \cdot (A \mid 3) \cdot B) \bmod 2^{32}$, where C_{i-1} denotes the former computed chaining value. Note that, especially in comparison to the Galois-Field multiplication presented before, the MultHash operation should be considered insecure. Nevertheless, due to its neat property of filling the internal state significantly fast, we include it in the comparison of possible instantiations of H' even if it will not be part of our recommended variants of Catena.

C Extensions of Catena

C.1 Password-Independent Random Layer

This part of the section is devoted to the function Γ, which reflects a graph-based structure generated by a password-independent indexing function R as shown in Algorithm 7 (left). As defined in the specification of Catena, it overwrites

$2^{\lceil 3g/4 \rceil}$ randomly chosen state words. In respect of the memory requirement in Table 5, it is intuitive to say that the penalty for a $\mathrm{GR}\ell^g_\lambda$ will not increase significantly since almost the whole state is already stored. On the other hand, when considering a BRG^g_λ, one would expect an increased penalty.

Algorithm 7 The functions $\Gamma(g, v, \gamma)$ (left) and $\Phi(g, v, \mu)$ (right).

Require: g : garlic, v : state,
γ : public input
Ensure: v : updated state
1: $r \leftarrow H(\gamma) \mid\mid H(H(\gamma))$
2: $p \leftarrow 0$
3: **for** $i \leftarrow 0, \ldots, 2^{\lceil 3g/4 \rceil} - 1$ **do**
4: $(j_1, r, p) \leftarrow R(r, p)$
5: $(j_2, r, p) \leftarrow R(r, p)$
6: $v_{j_1} \leftarrow H'(v_{j_1} \mid\mid v_{j_2})$
7: **end for**
8: **return** v

Require: g : garlic, v : state,
μ : secret input
Ensure: z : output of $flap$
1: $j \leftarrow R(\mu)$
2: $v_0 \leftarrow H'(v_{2^g-1} \mid\mid v_j)$
3: **for** $i \leftarrow 1, \ldots, 2^g - 1$ **do**
4: $j \leftarrow R(v_{i-1})$
5: $v_i \leftarrow H'(v_{i-1} \mid\mid v_j)$
6: **end for**
7: **return** v

Algorithm 8 Function $flap$ of Catena with Extension Φ

Require: c : garlic, x : value to hash, γ : public input
Ensure: x : intermediate hash value
1: $(v_{-2}, v_{-1}) \leftarrow H_{\mathrm{init}}(x)$
2: **for** $i = 0, \ldots, 2^g - 1$ **do**
3: $v_i \leftarrow H'(v_{i-1} \mid\mid v_{i-2})$
4: **end for**
5: $v \leftarrow \Gamma(g, v, \gamma)$ ▷ layer with γ-based indexing function
6: $v \leftarrow F(v)$ ▷ memory-hard function
7: $x \leftarrow \Phi(g, v, \mu)$ ▷ layer with μ-based indexing function
8: **return** x

First, we wanted to know if the currently designated position of Γ (invoked before F, see Algorithm 2) does provide the best possible resistance (in comparison to other chosen positions) against the precomputation method discussed in the former section. Therefore, we integrated the function (layer) Γ into F, fixed $g = 12$ (for time reasons), and recomputed the penalties for different positions of Γ within F, different values of λ, and the graph instantiations presented in Section 4. Further, we set the public input γ to the salt and fixed it to a constant value ($\gamma = 0$). We see this as a valid approach since if R is instantiated with a reasonable strong pseudorandom number generator, all indices computed during the invocation of Γ are randomly distributed values. The penalties depending on the positions of Γ are shown in Table 8, where it is easy to see that placing the random layer Γ at the third-last position leads, in general, to the highest penalty.

Graph	Layer Structure	
	$\Gamma F_0 F_1$	$F_0 \Gamma F_1$
BRG_2^{12}	33.34	31.84
$\mathrm{GR2}_2^{12}$	62.02	61.39
$\mathrm{GR3}_2^{12}$	79.68	81.06

Graph	Layer Structure		
	$\Gamma F_0 F_1 F_2$	$F_0 \Gamma F_1 F_2$	$F_0 F_1 \Gamma F_2$
BRG_2^{g}	62.44	103.81	81.27
$\mathrm{GR2}_2^{g}$	270.68	306.48	288.73
$\mathrm{GR3}_2^{g}$	248.48	264.10	257.30

Table 8. Penalties depending on the position of Γ within F and the particular graph instance, where F_i denotes the i-th layer of F and $g = 12$. The values are given for $\lambda = 2$ (left) and $\lambda = 3$ (right).

Thus, we assumed this configuration when computing the impact of the random layer for the recommended values of the garlic, i.e., $g \in \{18, 21\}$. Our first intuition was confirmed by the results shown in Table 9. Note that due to time issues, the values for the precomputation method with $\lambda = 3$ and $g = 21$ are a rough estimation based on the values for $\lambda = 3$ in Table 6.

Attack	Graph	Garlic (g)	
		18	21
Precomp.	BRG_2^{g}	54.11	107.36
	$\mathrm{GR2}_2^{g}$	169.18	374.66
	$\mathrm{GR3}_2^{g}$	365.40	1411.90
Naive	**all**	376.59	1432.87

Attack	Graph	Garlic (g)	
		18	21
Precomp.	BRG_2^{g}	417.03	836.24
	$\mathrm{GR2}_2^{g}$	1386.31	3090.14
	$\mathrm{GR3}_2^{g}$	1163.19	4675.89
Naive	**all**	6456.97	47581.00

Table 9. Penalties depending on the garlic g, the attack method, and the particular graph instance extended by a random layer Γ, where the values are given for $\lambda = 2$ (left) and $\lambda = 3$ (right).

By comparing the values presented in Table 6 and Table 9, we observed that the penalties only clearly increase for a BRG_λ^g. Thus, we can conclude that the additional invocation of Γ does help substantially against TMTO attacks based on the precomputation method, since the penalties given for a BRG_λ^g are still much smaller than that of a $\mathrm{GR}\ell_\lambda^g$. Nevertheless, the main aspect of including Γ in our core function $flap$ was the increased resistance against ASIC-based adversaries, and not to thwart TMTO attacks. Furthermore, the penalties presented in Table 9 only hold if an adversary does not have the additional $2^{3g/4} \cdot n$ bit of memory available.

C.2 Password-Dependent Random Layer

This extension is described by introducing a further graph-based layer below the call to F (see Algorithm 8, Line 7). In general, the function Φ is used to update the internal state v of size $2^g \cdot n$ bit by sequentially updating each state word.

An update of a state word v_i depends on two inputs (see Algorithm 7 (right)): First, the immediate predecessor v_{i-1} (or v_{2^g-1} if $i = 0$) and second, a value chosen uniformly at random from the state, where the index is determined by the indexing function R.

More detailed, the first update depends on the secret value μ, which we set per default to the value v_{2^g-1} of the output state of F. All subsequent updates dependent on the previous computed state value v_{i-1} (see Line 4 of Algorithm 7 (right)). Thus, we basically follow a slightly more generic approach in comparison to the ROMix function used within `scrypt` [25]. The function R can, for example, be given by `xorshift1024star` as used in the default instantiations of Catena [15], or a similar function to the one proposed by the designers of Argon2 [6], which uses the g least significant bits of the previous computed value to determine the new index.

It is easy to see that Φ conducts sequential writes, whereas Γ conducts writes to randomly determined words of the state. Thus, by calling Φ, Catena provides sequential memory-hardness as defined in [25]. Note that if the extension Φ is added to $flap$, the function F must return the full state v (see Line 6 of Algorithm 8) instead of a single value.

D Penalties Caused by Shifting Sampling Points

Graph	**No Shift**	**Min**	**Max**
		$\lambda = 2$	
BRG_2^{12}	25.03	25.00 (0,7)	31.59 (1,0)
$\mathrm{GR2}_2^{12}$	54.56	54.35 (14,12)	67.61 (4,7)
$\mathrm{GR3}_2^{12}$	74.25	72.88 (5,8)	74.25 (0,0)
		$\lambda = 3$	
BRG_3^{12}	47.59	46.70 (0,0,15)	90.13 (1,1,14)
$\mathrm{GR2}_3^{12}$	254.93	246.72 (12,13,4)	296.54 (6,4,11)
$\mathrm{GR3}_3^{12}$	223.52	217.64 (8,6,15)	262.36 (11,4,10)

Table 10. Relative costs (penalty) depending on the shift of the sampling points and the depth λ.

Assessing the User Experience of Password Reset Policies in a University

Simon Parkin, Samy Driss, Kat Krol, M. Angela Sasse

Department of Computer Science, University College London, UK
{s.parkin, samy.driss.14, kat.krol.10, a.sasse}@ucl.ac.uk

Abstract Organisations often provide helpdesk services to users, to resolve any problems that they may have in managing passwords for their provisioned accounts. Helpdesk logs record password change events and support requests, but overlook the impact of compliance upon end-user productivity. System managers are not incentivised to investigate these impacts, so productivity costs remain with the end-user. We investigate how helpdesk log data can be analysed and augmented to expose the user's personal costs. Here we describe exploratory analysis of a university's helpdesk log data, spanning 30 months and 500,000 system events for approximately 10,000 staff and 20,000-plus students. The scale of end-user costs was identified in log data, where follow-on exploratory interviews and NASA-RTLX assessments with 20 students exposed issues which log data did not adequately represent. The majority of users reset passwords before expiration. Log analysis indicated that the online self-service system was vastly preferred to the helpdesk, but that there was a 4:1 ratio of failed to successful attempts to recover account access. Log data did not capture the effort in managing passwords, where interviews exposed points of frustration. Participants saw the need for security but voiced a lack of understanding of the numerous restrictions on passwords. Frustrations led to adoption of diverse coping strategies, for example deliberately waiting to reset a password after reaching the post-expiry grace period. We propose ways to improve support, including real-time communication of reasons for failed password creation attempts, and measurement of timing for both successful and failed login attempts.

1 Introduction

An organisation may have a password policy, which dictates rules about the character composition of passwords for provisioned accounts and how these passwords should be managed (e.g., how often they should be changed). Password policies in organisations can be unusable – and consequently insecure – if the end-user experience is not considered [10]. Policies may dictate that users regularly change their passwords, complicating attempts to remember a password. It is already common to forget passwords [36], especially when managing many at once [6].

To reduce the impact of forgetting passwords, password recovery mechanisms are often provided. These typically take two forms [21]: (1) self-service, authenticating through an alternate factor such as security questions, and/or; (2) helpdesk, where a representative will reset a password on behalf of a user.

IT security policies must accommodate regulatory and economic considerations [16]. Policy decisions are typically based on intuition and security expertise, and consider business impact ahead of any effects upon end-users [19]. This does not necessarily lead to poor choices, but is subjective and enacted without evidence of productivity impacts for end-users who have jobs to do.

For the individual, a password reset can appear to lack a personal benefit for the amount of effort required [20]. This can encourage negative attitudes towards computer security, leading users to subvert security mechanisms [5]. Assertions about the effectiveness of password reset policies are then weakened if end-users feel pressured to respond to the burden in unanticipated ways. It may not be possible for those responsible for maintaining security to observe these unanticipated responses (or otherwise seem obvious for them to monitor behaviours if they have no evidence that policy is not being enacted) [3].

Here we explore what the data collected for typical password support mechanisms can and cannot indicate about the end-user experience, and where they may be augmented to improve both usability and security. A university provided access to helpdesk log data for password reset activity, spanning 30 months, for all users of an authentication system for email and other centrally-managed IT services. This data was analysed using Exploratory Data Analysis (EDA) techniques to find out what it shows of the end-user experience for (un)successful reset or recovery of passwords over time. Where analysis did – or did not – describe user experiences, this informed questions posed during exploratory interviews with 20 students, structured to explore attitudes towards specific aspects of password policy (such as responding to password expiry notifications and adhering to password composition rules). Interview analysis relates the end-user experience to numbers in the helpdesk logs, with an aim to identify candidate extensions to both the logging process and the password support mechanism itself. A contribution of this paper is the application of holistic thinking to the way security infrastructure supports and monitors users and their security behaviours.

The remainder of the paper is organised as follows: In Section 2, we give an overview of related research on password policies in organisations and end-user behaviours around authentication. Section 3 summarises the methodological approaches in our research. Section 4 presents our findings from the analysis of helpdesk log data while Section 5 focuses on the qualitative results from user interviews. We discuss our findings in Section 6 and build on them to provide recommendations for practitioners. Finally, Section 7 summarises our findings and outlines avenues for future research.

2 Related Work

A number of works have analysed organisation log data and engaged with system users to develop understanding of behaviours around password policies and related authentication mechanisms. Adams and Sasse [5] stress that IT security managers must consider that (i) users can decide whether to comply with security policy, and (ii) the decision to comply is shaped by goals, perspectives and attitudes, and the norms that govern behaviours. Security is not at the forefront of users' minds [35], and should not be a barrier to a user's primary task or it will create frustration and promote negative attitudes towards security [10]. Tightening security can not only render systems less accessible and make users less productive – it might also undermine security as users will employ workarounds to carry out their tasks [25]. The same can be said of demands that occur repeatedly or in sum over time [10].

The usability and productivity impacts of security mechanisms are rarely considered when devising policies [10]. The user burden of security can be measured as the impacts they experience, for instance delays to task completion, restrictions to their ability to complete their work, and in the case of passwords the memorability of new information becomes critical [8]. A user may begrudgingly commit effort to compliance with security, if only to avoid being held accountable for a security compromise [10]. Shay et al. [4] look specifically at user behaviours and attitudes towards passwords and password change at a university, providing recommendations for how to manage user responses to policy expectations. Strategies for managing passwords were also identified, where a number of survey respondents remarked that they wrote down their passwords, or otherwise shared them with others or followed a repeatable set of rules for creating a new password such as reusing their previous password or a password from another account. Here we consider the impacts upon users which may be hidden from the view of system managers, even where user behaviour as observed in system logs appears to be compliant with the password policy.

It has been suggested that the design of security systems be based on task and work-flow analysis [28], and that security restrictions be routinely tested with various users to minimise interference and enhance (rather than hinder) productivity [25]. Elsewhere productivity impacts for groups of users have been considered alongside business support costs and recognised efforts to reduce security breaches, in a decision-support paradigm [9] and complementary proposal for tool support [24], with a focus on password policies. The paradigm was well-received by security managers, especially where it was able to relate to decisions and data that managers were familiar with.

Organisations may employ a helpdesk team to help users who need to replace forgotten passwords. Alternatively, users may manage passwords personally through self-service portals, or by using a registered contact point (e.g., having information sent to a registered email account) [26]. The number and complexity of password policies plays a role in increasing the frequency of helpdesk visits for password resets [5], where it is estimated that roughly 30% of help requests relate to password resets [33], the majority as a result of passwords be-

ing forgotten. Password reset requests are most common after holidays or after long periods of inactivity [13]. Additionally, helpdesk calls may result from users being locked out, as it is common for organisations to lock user accounts after three failed login attempts (where increasing this limit can reduce the number of reset requests [13]). In some organisations, the cost of maintaining helpdesks leads security staff to consider the writing down of passwords as a minor infraction [30]. Many organisations then seek to reduce the number of calls to the helpdesk [14]. Self-service solutions however require investment and support to be appropriately integrated into designated systems [31]. When selecting or changing a password, users should be provided with instructions for constructing and memorising strong passwords [29]. Without such information, users make up their own rules to devise more memorable passwords [5]. Training users to create secure and memorable passwords has been identified as a more plausible solution to easing the burden on the helpdesk [17]. Here we also consider where communication of policy can be improved without necessarily changing the rules of the password policy itself.

3 Methodology

A mixed methods approach combined both quantitative and qualitative data. Exploratory Data Analysis (EDA) [34] was performed on the anonymised helpdesk log data. Second, semi-structured interviews were conducted to understand end-user perceptions of system use and related policies. Interviewees completed NASA-RTLX workload assessments [18] for regular tasks, thereby quantifying perceived workload. Interviews and workload responses were then analysed relative to the findings of the log analysis. The study received ethics approval from the Research Ethics Committee at UCL. All quantitative data was anonymised. Participants in interviews were assigned participants numbers and no personally-identifying information was stored during data collection.

3.1 Systems under analysis

Analysis was carried out with a university support team's password reset log data. The data comprises monthly reports of the frequency distribution for various password reset activities (e.g., resetting an expired password) from September 2012 to April 2015, for approximately 10,000 staff and 20,000-plus students. The data contains logs from two systems: Personal Password Management and Personal Password Recovery, where each system has both an online and in-person helpdesk channel. The main dataset consisted of monthly aggregations, to both manage the scale of the data and to preserve anonymity.

The **Personal Password Management (PPM)** is a web-application allowing users at the university to change their account passwords. Here a *password change* is the act of replacing an existing known password with a new one. *Reminder emails* are sent to system users in advance of a password being set to expire – a user will receive a series of timed emails at set intervals until they reset

their password. There is a *grace period* of several days after expiry when a user can still reset their password. New passwords must conform to set rules, which define character composition and prohibit similarity to previous passwords. A password change may take time to propagate across all university IT systems.

The **Personal Password Recovery (PPR)** system allows users to register memorable phrases and associated hints so that they can authenticate themselves to the system if they have forgotten their password or it has expired – a user can then enact a *password reset*, to replace their password. The process can be enacted online personally, or via the helpdesk in-person or over the phone. Users must also provide a limited number of personal details. When answering memorable questions, users are prompted to only enter selected characters from their memorable answers. A *password reset* is not the same as *password recovery*, where a user may indicate on a webpage that they have forgotten their password and follow a direct link (or indirect link, perhaps via email) to a webpage to change their password – such a feature was not part of the system under analysis.

3.2 Helpdesk log analysis

Exploratory Data Analysis (EDA) can identify patterns within a dataset to visually summarise its primary characteristics [34]. The inherent advantages of EDA include that it allows researchers to uncover patterns and structures within the data that are not readily discerned. EDA is applied to the password helpdesk data to carry out (i) data munging, (ii) examine key variables and relationships that may exist between them, (iii) visually describe the data and, (iv) determine temporal patterns and trends that may be of interest to IT security managers. In this study, EDA findings also informed the design of a subsequent user interview exercise.

3.3 User interviews

Semi-structured interviews were conducted, to facilitate probing of responses where they related to unanticipated factors or issues identified from log analysis. Thematic analysis was used here to identify coding themes and sub-themes based on topics emerging from the interviews. The analysis followed the six phases identified in [12]: familiarisation with data, generating initial codes, searching for themes, reviewing themes, defining and naming themes and producing the report. The objective of the analysis was to document and reason about the end-user experience with password resets, helping to explain findings in the log data.

The student population of the university was the focus of the interviews – staff at the university had a wider range of password management options available to them, but all users shared access to the same core set of functions considered in this paper. Also, staff may have had clerical, support, academic, or research roles, which would influence the systems they regularly accessed (where students had access to provisioned systems such as timetabling and teaching resources).

A pilot study was conducted to assess the interview questions. The final set of questions consisted of 9 open-ended questions, related directly to metrics in the helpdesk log data. A pre-screen survey determined eligibility for the study proper – the pre-screen determined that all participants were students at the subject university, and also recorded whether the participant had ever forgotten their password and "lost access to [their university account] for any amount of time as a result". An information sheet was provided at the start of each interview, alongside a consent form to be signed. Each interview lasted approximately 20 minutes and was audio-recorded. Each participant received £5 for their participation.

3.4 NASA Raw Task Load Index (NASA-RTLX)

Interviews were complemented with a NASA-RTLX (Raw Task Load Index) workload measurement [15] for password use, where participants assess a task by providing ratings for perceived workload [18]. The scheme employs six subscales to assess user workload, on a scale of 0 to 100 for each subscale: mental, physical and temporal demands as well as performance, effort and frustration. Users assign a score for each individual factor (with all but performance being measured with a high score equating to a negative impact upon the respondent).

Workload measurements rely on users' recollection of completing a task [7]. Here NASA-RTLX served to promote discussion, and was used to explore potential metrics for user perception of password use. Three RTLX forms were prepared for each interview, assessing workload for password reset, PPR registration and PPR authentication. If an interviewee had not registered/authenticated to the PPR system, related RTLX forms were not included. RTLX responses were reviewed during the interview, as cues for discussion.

4 Results: Helpdesk Log Analysis

In this section we discuss the results of the log analysis and user interviews, relating interviews to log analysis to expose links between user experiences the log data. Specifics of the subject university's password policy are not discussed, so as not to identify the institution.

4.1 Results

Figure 1 shows time-plots of monthly summary data. There are noticeable peaks in resets around September, October and January of each year. These months follow longer periods of inactivity (when students and staff return from summer/winter holidays and begin study). Frequencies increase gradually over the timespan of the data, reflecting an increase in users. It is important to understand how support mechanisms are utilised, to manage the impact of increased activity. The month of September 2013 does not align with the same period in

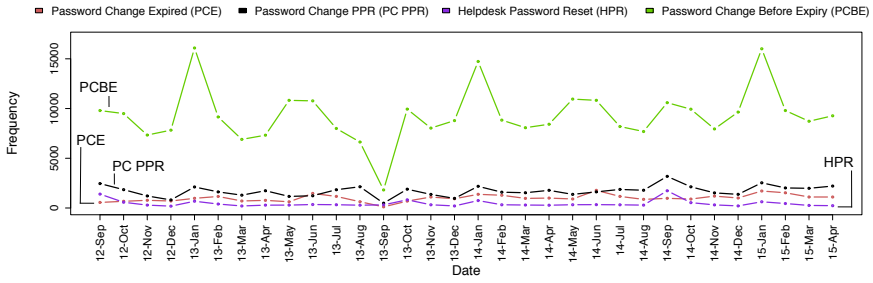


Figure 1. Time plot for password related fields.

other years – where numerous systems interact in complex ways, mapping the relationships between data fields could help to rationalise such one-off events.

Figure 2 shows the different ways in which staff and students have reset their passwords. With a mean of 74.4% the most-used approach to resetting passwords is via the PPM system, signifying that the majority of users follow the channel preferred for limiting helpdesk costs. However, there are a consistent minority of users resetting their passwords within the expiry "grace period", implying either that reminder emails are not completely effective, or that there is always a small proportion of users who are unable or do not want to reset their passwords before expiry. Reasons for using the PPR and how users perceive it is also important for the interviews, as up to a fifth of resets are enacted through recovery mechanisms.

From Figure 2, the provisioned recovery channels (i.e., PPR and helpdesk) are – as intended – not used by the majority of registered users. As illustrated in Figure 2, password changes via the helpdesk are the least common method of choice for users, with an average of 3.6%.

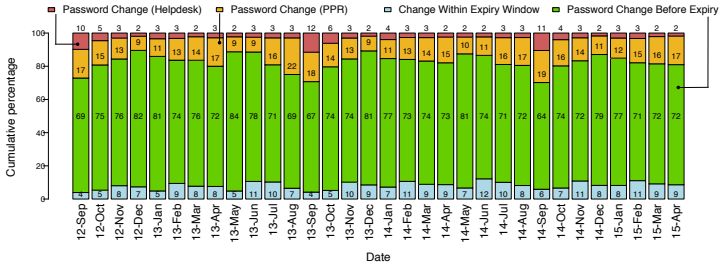


Figure 2. Stacked bar plot for percentage of password change events.

The system that produced the data logs does not record any data to indicate the experience of users *as they are resetting their passwords*. This then becomes

another area of focus for subsequent interviews. It is also unclear from the summary log data how individuals reset passwords in response to reminder emails, especially whether reminders are knowingly ignored (a general behaviour noted elsewhere [19]). How end-users react to notification emails is another subject to explore in interviews. Without recording for example the duration of password reset or recovery events (online or in-person, for comparison), it is unclear whether the helpdesk is little-used because of a comparative convenience in using the PPM system, or because of a lack of awareness about its existence. This is an important point to clarify if system managers ever intend to retire or reduce helpdesk support (for instance to manage costs, a concern touched upon in Section 2).

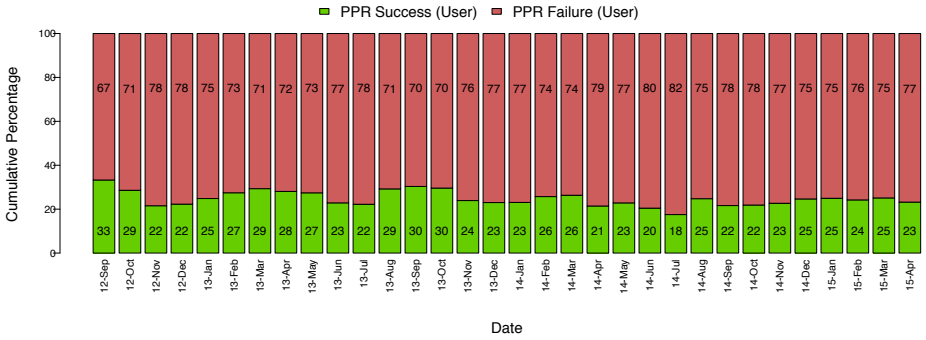


Figure 3. Percentage of attempts that are failures/successes.

The relative proportions, expressed as percentages, between PPR failure and PPR success show that on average 75.2% of attempts to authenticate to the PPR fail. Likewise, the mean success rate stands at 16%. Figure 3 shows a time plot illustrating the relative percentages between PPR failure (user) and PPR success (user). If all such events were attributable to legitimate users the figures could be distressing, considering that based on the data available the lowest failure rate (September 2012) is 66.7% and the highest 82.5% (July 2014). The high failure rates may be due to the *demands* of the recovery process, or may reflect success in managing automated attempts to access the system – other work has for instance highlighted that it can be difficult to distinguish brute-force attacks from other attempts to access user accounts [2]. The PPR failure and success rates also raise the question of how many PPM password reset attempts fail (something which is not recorded in the dataset) – user interviews examine the personal impact and perception of failure to create a new, policy-compliant password.

Going beyond the aggregated data, an event-by-event dataset was examined to explore PPR failure rates. This dataset associated anonymised user identifiers with specific PPR/PPM/helpdesk events, enabling further exploration of

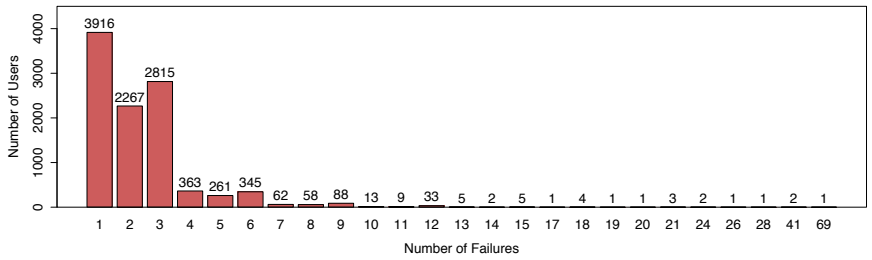


Figure 4. Number of users failing PPR authentication x times within a 24-hour period of using the system.

the PPR failure counts. Figure 4 shows a bar plot of the number of PPR failures for a user within a 24-hour period (representing a unique session of interaction with the PPR). From Figure 4, there are many users who are failing 1-2 times each time they use the system. However, the number of users who are failing 3 times also represents the second highest figure in the chart – this is important as after 3 successive failures access to the PPR is locked for a time. At this point a user may decide to go to the helpdesk or wait before trying to use the PPR again (where the latter may explain the lower occurrences of higher failure rates beyond 3 attempts). Given the number of users at the subject institution this may constitute a large drain on working time for both users and helpdesk staff. The effectiveness of interactions with the PPR system was then discussed in interviews. Understanding responses to failure would also rationalise the distinction between the "noise" of unusable security and genuine attempts to access the system without first-hand knowledge of the associated credentials. There is a long-tail distribution to the tallies of failed PPR use. It becomes unclear as tallies increase whether this represents determined users or automated authentication attempts, where the distinction (and perhaps then the response) is not clear. The high PPR failure rates were then of further interest in subsequent interviews.

5 Results: User Interviews and NASA-RTLX

The goal of the interviews was to explore the perceived impacts that the password reset policy is having on its users, in this study specifically students. Equally the study aimed to determine how user feedback could be used to explain findings identified in the EDA process, with interview questions targeted to explore specific outcomes from the log data analyses.

Interviews were transcribed, and transcripts re-read many times to become familiar with the data, and a priori notes and patterns were recorded to develop codes. An initial list of codes was generated by one of the authors following

data reduction (e.g., subtracting and merging codes) and complication (e.g., adding and splitting codes). An inductive thematic analysis approach derived concepts and perspectives from the interview data. A code book was maintained throughout. This codebook was reviewed at regular intervals by three of the authors.

From the pre-screen responses, 20 participants were selected to be interviewed, all students at the university. 16 were female and 4 male, mean age was 25. Furthermore, 9 participants reported being registered on the PPR system whilst 11 stated they were unaware of it.

5.1 Results

Our analysis identified four overarching themes (with a total of 10 sub-themes) which we described in the following sections.

Impact of stringent policy Many interviewees felt that the password reset policy was restrictive, with many using words "*annoying*" (10 interviewees) and "*inconvenient*" (8) to describe changing their password. Interviewees identified themselves as security-conscious and understanding of the need for secure behaviour, but nonetheless the majority felt burdened by the process.

(A) Difficulty meeting criteria. For the majority of interviewees (17), creating a new password that met the necessary criteria required multiple attempts, and was a process of trial and error. Some (8) felt that the password criteria were too strict, with the perception that no words or names could be used. For some (6), there was an expectation of having problems with every password change. Two interviewees implied that their initial experience informed their impression of how the process would be in future, P8 explained: "*. . . so like when I* [went to] *reset it, like my first password. It was a pain because I had to* [try] *like 10, 15 times in a row and then I went: "Ah okay, this is gonna be fun if I have to do it again."*" Several respondents (9) expressed frustration at having produced passwords, which they felt met the criteria, but which were then rejected by the system. The difficulties of failed or protracted attempts to change passwords were not represented directly in the helpdesk logs.

(B) Overwhelming frequency of password change. More than half of participants (13) felt that password reset requests were too frequent with 6 considering the frequency was more of a problem than meeting the creation criteria. P8 pointed out that passwords are subject to such strict requirements that they should be kept for longer: "*. . . the password, like the requirements are so strict so I don't see the point of changing it like 2 or 3 times a year.*"

When contrasting the policy with other accounts they had used, a number of participants (7) indicated that the university's policy required the *most* frequent changes; they were puzzled as to why the frequency was so high, "[At] *my previous university where I was for 4 years, I didn't have to reset my password at all, during the 4 years!*" (P13).

Effectiveness of support

(A) Lack of information. Some (6) participants saw "*no rational explanation*" for the policy, querying why it was so stringent. P15 explained: "*I mean if there are any pertinent security reasons for having to change it so often, I think you better explain why for non-technical users. Um, cause it's like... I'm just feeling angry that I have to change so often if I haven't understood why.*". Several interviewees (7) felt that insufficient feedback during a password reset made it a "*guessing game*" to satisfy the various rules, as P17 told us: "*It just tells me that it's just too similar or it has to have this and it... I think it tells you one thing at once.* [...] *Instead of explaining everything in one go... What a good password should look like, I have to* [use trial] *and error every time.*" This implies that unique attempts to create a password may take a long time or that there may be many failed attempts in quick succession before a successful password change.

(B) Efficacy of email reminders. 8 participants *knowingly* ignored reminder emails as they were either not near a computer to carry out the reset (a phone screen was regarded as inconvenient to type on), or did not feel a sense of urgency to do so (6); "*I read* [the reminder email] *and... I figured I could do that at some point in time and I didn't really see the urgency and then I got a second reminder and then I thought: "Okay, now I have to do it."*" (P17).

Nearly all interviewees (18) reset their passwords before the deadline, which aligns with the log data analysis (see Figure 2), and implying that the reminder emails were having an effect. For some (5), this was because the email reminder prompted a *fear* of the unknown as to the consequences of missing the deadline. However, three participants did perceive reminders as an effective tool that helped to avoid expired passwords. This highlights that "successful" users of the system may or may not at the same time experience stress in complying with expectations.

(C) Recovering forgotten passwords is not a problem. Some (5) participants had at one time employed the PPR service to reset their forgotten password which all described as "*straightforward*" (what may be regarded as comparable to the log data in Figure 2). Though no experiences were reported of being locked out of an account, 3 had experienced difficulties in recalling memorable answers before successfully authenticating. PPR registration was encouraged when joining the university, where memorable answers may be required some time after registration, challenging the capacity to correctly recall configured answers (or in the case of the study participants, recall whether they had registered for the system at all). Similarly, in a study on user-chosen challenge questions, Just and Aspinall [22] found that 18% of their participants were unable to provide at least one of their answers correctly after 23 days from creation and this despite, as the authors emphasised, the participants being young and potentially having excellent memory.

Of the 20 participants, 9 had registered for the PPR service whilst 3 had heard of it but not registered and the remainder were not aware of its existence.

Those who had visited the helpdesk for queries – a minority, as reflected in the log data represented in Figure 2 – described it as a "*clear cut*" and "*quick*" procedure. These accounts raise the question of how well the support systems are publicised.

Development of coping strategies

(A) Coping with frequency of reset. Participants were divided between resetting upon receipt of the initial email (10) and relative to the expiry date of the password (10). This division is not reflected in the metrics of the helpdesk logs, where a reset anywhere between the initial email and the expiry date is logged as a "Password Change" either way (as in Figure 2).

Two participants deliberately delayed resetting until the last few days preceding the expiry date, to put off the next time that they would have to complete the process. Four participants also felt that delaying the reset allowed them to plan ahead. It is possible that similar strategies can explain password resets during the expiry grace period (as seen in the log data, Figure 2), in that for some users there may be a deliberate effort to delay interactions with the system as long as possible (and not just for instance representing users who experience difficulties in changing a password in time).

(B) Coping with password creation. More than half of the interviewees (12) developed a strategy for creating passwords that were both acceptable and memorable. Most common was a scheme of sequential changes, taking words from a category of items (e.g., colours, city names), and modifying them to meet the policy requirements. A new item was then derived from the category with each change. Such strategies are examined elsewhere [4], where the genuine change in composition (and guessability) across old and new passwords is put into question.

(C) Coping with recalling passwords. The most evident strategy for recalling passwords was for users (6) to write their passwords down [5] which either took the form of physical paper (4) or an electronic document (2). Most of these individuals (4) acknowledged the security risk in doing so yet felt compelled to do so given the complex criteria. With paper-based recall aids, some interviewees (3) admitted to having lost it. These "offline" recall aids would require a further investment of resources for security managers to monitor. Five participants stated that they only work from their personal computers, and hence reduce the burden of remembering by caching passwords in their browser (a behaviour that is not unique and has been noted elsewhere [11]).

Shaping perceptions towards security

(A) Accepting of security, but... Many (10) acknowledged the need for security despite feeling that it was a *pain*. P7 explained: "*Oh yeah, I understand it should be done like that... Because you know it can be something quite personal, with*

my degree and stuff. But um... It just gets annoying, it's like both sides anyways. I find it annoying but I understand it has to be done.". A number of users (7) also felt a sense of immunity to security breaches and as such did not comprehend the level of security being mandated by the policy. P5 told us: "*I only have my emails which are literally just: "Come to the seminar, it's next week." and I have no sensitive information in my email. [...] I think my personal email might... require more of these mandatory password changes*".

(B) The definition of security. 9 participants had the impression that the university takes security "*very seriously*", acting as a model of what constitutes "security". P3 explained: "*I think* [it's] *definitely one of the most secure from that point of view, I've never had to change my password so often.*". For some, this influenced their decisions in managing passwords. 4 participants admitted setting other account passwords to mirror their university password, in part because the university rules seemed to be the most "*secure*".

5.2 RTLX data analysis

All participants completed the RTLX ratings for the password reset task. RTLX for PPR registration and authentication applied to 9 and 5 of the participants respectively.

Figure 5 shows box plots of the subscales for each task (with the scale of Performance inverted so that higher values equating to better perceived task performance). With mental demand, the password reset task was comparatively more challenging than the PPR-related tasks, with a mean of 58 (on a scale of 0 to 100) for the password reset task whilst for PPR registration and authentication, these values were at 31.6 and 43 respectively. The distributions for the PPR-related tasks suggest that some users of the system would have difficulty creating/remembering memorable answers. Temporal demand was higher and more widely-distributed for the password reset task than for PPR tasks (which again would not have been reflected in the log data).

For password changes/resets the performance varied, further highlighting that participants found it difficult to generate a valid password. With PPR-related tasks, perceived performance was high, with users feeling that they successfully managed these tasks – this is in contrast to the EDA findings that apply to the larger base of users.

Frustration levels were the highest measure for password reset tasks, with a mean of 61.8. Frustration was also notable for PPR authentication, perhaps owing to recalling memorable answers – this and the related mental demand may help to explain some of the high PPR failure rate described in the log data analysis results. The low frustration for the PPR registration process complements the perception of the task being "*straightforward*". Regarding the frustration scale for the password reset task, dialogue with interviewees implied that reminder emails and the actual task of creating a new password were all part of an interlinked process – this may create some uncertainty as to what it was that

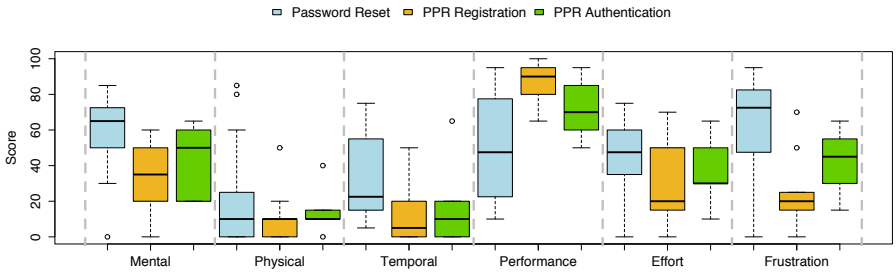


Figure 5. Box plot of average RTLX subscale results for password reset (20 responses) and PPR registration (9 responses) and PPR authentication (5 responses) tasks.

participants assigned scores to, but reinforces that the perceptions of reminder emails and resetting passwords can impact (and potentially reinforce) each other.

The mental demand of PPR authentication was more varied (exhibited a wider span of responses on the TLX subscale) compared to password reset and PPR registration. This could be attributed to needing to recall "memorable" answers which were set potentially months (if not years) beforehand, where some participants used what they regarded as knowingly obvious "memorable answers" and others had difficulty in recalling their answers later on when they needed them.

6 Discussion

Interviewees devised coping strategies to both produce passwords that the system would not deem similar to previous ones, and avoid the effort of resetting (e.g., postpone the next reset for as long as possible). The latter may contribute to – and be masked by – the peaks in resets seen in log data (Section 4.1), warranting further investigation. Coping strategies have been noted during other studies with university participants [4] (especially modification of a previous password), suggesting that this is far from being an isolated case.

Interviewees regarded the stringent policy as the "*most secure*" they had seen, thereby confounding number of rules with security. Participants felt that replicating their university password across their other accounts was appropriate to protect their (personal) data. It may be that students especially may take these habits and understanding with them after their time at university and into workplace environments.

The combination of log analysis and user interviews indicated that the Personal Password Recovery (PPR) facility may have usability issues, as a great proportion of attempts to authenticate to the system resulted in failures (where the significance of authentication failures of genuine users within the data warrants further analysis, as hinted in Figure 4). Use of the PPR system is then counter

to the expectation that self-service mechanisms minimise helpdesk calls [14], although the log data did show that visits to the helpdesk were minimal (as in Figure 2). To the contrary, interviewees who had used the PPR expressed a general consensus that it was "*straightforward*" to use, with relatively low TLX values for both registration and authentication tasks. These results raise the possibility that there is a *subset* of users, under-represented in our interviewee cohort, who consistently have difficulties authenticating to the system. Consideration should be given to how to engage with these "forgotten users".

The log data did not indicate the effectiveness of reminder emails; resets after expiry were low, though non-negligible (Figure 2). Some interviewees felt that reminder emails were too frequent and too early, where they ignored them or developed a distracting sense of *pressure* to change their password. This contributed to the wide-ranging responses for the RTLX "temporal demand" measure for password resets.

For interviewees who replicated their university password across other accounts, the workload of resetting their university password would then be compounded by the number of accounts they have in and outside the university. Not having a coping strategy is also a drain on productivity, see P13: "*And so you spent time thinking of something, you put that in and then it [tells you it's not valid] and then you go: "Alright, I have to think of something else*"". Logging both the *number of attempts* to create a password and the *amount of time* to devise a valid password could indicate the usability of the composition rules and whether predefined coping strategies are being used.

Interviewees lacked comprehension as to why such stringent rules were necessary, which contributed to perceived inconvenience. A perceived lack of reasoning for the password policy may also support the sense of immunity to threats voiced by some participants. It is then important to relate policy to the context in which an account is used, P10 stressed: "*So I'm guessing it's the same policy for all... From undergrad to like PhD and bachelors and all. So I can see why higher up the chain they probably* [need that level of security] *at a high level but at undergrad it might not.*".

6.1 Recommendations for practitioners

1. Provide users with *real-time* feedback during password creation, which may reduce number of attempts, frustration levels, and time expended on creating a new password.
2. Improve the communication and justification of password policies to users. That is, *explicitly* relate potential/existing threats to what they do with their accounts.
3. Do not *assume* that policies are highly usable even if there is evidence that a majority do not require helpdesk support, as users may devise coping strategies which although compliant with policy may be counter-productive to the security of both the individual and the organisation.

7 Conclusions

Password support log data at a university was analysed. Taken by itself, analysis suggested that most users comply with the university's password policy with few problems. However, there may be hidden costs not reflected in typical helpdesk logs – many interviewees saw the policy as too stringent and frustrating, in some cases developing (insecure) coping strategies, tailored specifically towards complying with the policy. Several interviewees delayed password resets and the effort they associated with the process. A number of interviewees interpreted the strictness of the password policy to mean it was a *secure* policy.

Interviewees who had used the Personal Password Recovery (PPR) system indicated through NASA-RTLX workload scores that it was a *straightforward* process. Analysis of both aggregated data and event-by-event log data indicated that most attempts to provide memorable answers for a specific account failed 1-2 times, and that a minority of users may be experiencing trouble using the system (or otherwise that illegitimate attempts to access accounts are difficult to distinguish from genuine attempts, exacerbating the matching of resources to the support of users). The lack of feedback and justification of policy that users saw when creating passwords contributed to frustration.

Future work will continue analysis of the event-by-event log data obtained after initial analysis and reporting, and will engage with students and staff equally. Initial focus will be on responses to reminder emails and individuals' interactions with the PPR facility. It was noted that interviewees treat the entire password reset process as a single task (from initial email reminder to devising a new password), and so workload assessment techniques which acknowledge secondary tasks (such as security) may be necessary.

Findings highlighted that the log data offered limited insight into the user experience of password management, such as habits and points of frustration. Measures of effort and consequences are then important to consider, as these were shown to have potential hidden costs for security managers. NASA-RTLX assessments that quantify perceived workload were a tentative first step. In a similar vein, findings may inform repeatable surveys deployed on a larger scale to reach both students and staff (particularly long-term users).

Acknowledgements

Simon Parkin and Angela Sasse's research is funded in part by EPSRC, grant number: EP/K006517/1 ("Productive Security"). The authors would like to thank the participating university and especially their IT department for providing the data that informed this publication. The authors would like to thank Ingolf Becker for his help with the editing of this paper.

References

1. Florêncio, D., Herley, C., Van Oorschot, P.C.: Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts, *Proc. USENIX Security.* (2014)
2. Florêncio, D., & Herley, C.. Where do security policies come from?. Symposium on Usable Privacy and Security (p. 10). ACM, (2010)
3. Kirlappos, I., Parkin, S., & Sasse, M.A.: Learning from "Shadow Security": Why understanding non-compliance provides the basis for effective security. USEC (2014)
4. Shay, R., Komanduri, S., Kelley, P. G., Leon, P. G., Mazurek, M. L., Bauer, L., ... & Cranor, L.F.: Encountering stronger password requirements: user attitudes and behaviors. Symposium on Usable Privacy and Security (SOUPS). ACM, (2010)
5. Adams, A., Sasse, M.A.: Users are not the enemy. *Communications of the ACM* 42.12 (1999): 40-46
6. Adams, A., Sasse, M.A., Lunt, P.: Making passwords secure and usable. *People and Computers XII.* Springer London, (1997): 1-19
7. Albers, M., Patton, J.T.: Measuring cognitive load to test the usability of web sites. *Annual Conference-society for technical communication.* Vol. 53. (2006)
8. Anderson, J.: Why we need a new definition of information security. *Computers & Security* 22.4 (2003): 308-313
9. Arnell, S., Beautement, A., Inglesant, P., Monahan, B., Pym, D., Sasse, M.A.: Systematic decision making in security management modelling password usage and support. *International Workshop on Quantitative Aspects in Security Assurance.* Pisa, Italy. (2012)
10. Beautement, A., Sasse, M.A., Wonham, M.: The compliance budget: managing security behaviour in organisations. *Proceedings of the 2008 Workshop on New Security Paradigms.* ACM, (2009)
11. Besnard, D., Arief, B.: Computer security impaired by legitimate users. *Computers & Security* 23.3 (2004): 253-264
12. Braun, V., Clarke, V.: Using Thematic Analysis in Psychology. *Qualitative Research in Psychology*, 3 (2006): 77-101
13. Brostoff S., Sasse M.A.: Ten strikes and you're out: Increasing the number of login attempts can improve password usability. *Proceedings of CHI 2003 Workshop on HCI and Security Systems.* (2003)
14. Broome, C., Streitwieser, J.: "What is E-support. Service and Support Handbook". Help Desk Institute, (2002): 31-40
15. Byers, J.C., Bittner, A.C., Hill, S.G.: Traditional and raw task load index (TLX) correlations: Are paired comparisons necessary. *Advances in industrial ergonomics and safety* I (1989): 481-485
16. Coles, R.: Keynote Address, Eighth Workshop on the Economics of Information Security (WEIS 2009), University College London, England, (2009): 24-25
17. Charoen, D., Raman, M., Olfman, L.: Improving end user behaviour in password utilization: An action research initiative. *Systemic Practice and Action Research* 21.1 (2008): 55-72
18. Hart, S., Staveland, L.: Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Advances in psychology* 52 (1988): 139-183
19. Herley, C.: So long, and no thanks for the externalities: The rational rejection of security advice by users. *Proceedings of the 2009 Workshop on New Security Paradigms Workshop.* ACM, (2009)

20. Inglesant, P., Sasse, M.A.: The true cost of unusable password policies: Password use in the wild. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, (2010)
21. Jakobsson, M., Myers, S. eds.: *Phishing and countermeasures: Understanding the increasing problem of electronic identity theft.* J. Wiley & Sons. (2006)
22. Just M. and D. Aspinall. Personal choice and challenge questions: a security and usability assessment. Symposium on Usable Privacy and Security (SOUPS). ACM, (2009)
23. Komanduri, S., Shay, R., Kelley, P., Mazurek, M., Bauer, L., Christin, N., Cranor, L., Egelman, S.: Of passwords and people: measuring the effect of password-composition policies. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, (2011)
24. Parkin, S., Inglesant, P., Sasse, M.A., van Moorsel, A.: A stealth approach to usable security: helping IT security managers to identify workable security solutions. Proceedings of the 2010 Workshop on New Security Paradigms. ACM, (2010)
25. Post, G., Kagan, A.: Evaluating information security tradeoffs: Restricting access can interfere with user tasks. *Computers & Security* 26.3 (2007): 229-237
26. Reeder, R., Schechter, S.: When the Password Doesn't Work: Secondary Authentication for Websites. *IEEE Security & Privacy*, 9.2 (2011): 43-49
27. Riley, S.: Password security: What users know and what they actually do. *Usability News* 8.1 (2006): 2833-2836
28. Sasse, M.A.: Computer security: Anatomy of a usability disaster, and a plan for recovery. *Workshop on Human-Computer Interaction and Security Systems, CHI.* (2003)
29. Sasse, M.A., Brostoff, S., Weirich, D.: Transforming the 'weakest link' a human/computer interaction approach to usable and effective security. *BT Technology Journal* 19.3 (2001): 122-131
30. Sasse, M.A. and Fléchais, I.: "Usable security: Why do we need it? How do we get it?" (2005): 13-30
31. Skaff, G.: An alternative to passwords? *Biometric Technology Today* 15.5 (2007): 10-11
32. Tejaswini, H., Raghav Rao, H.: Protection motivation and deterrence: a framework for security policy compliance in organisations. *European Journal of Information Systems* 18.2 (2009): 106-125
33. Tari, F., Ozok, A.A., Holden, S.: A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords. Symposium On Usable Privacy and Security (SOUPS). (2006)
34. Tukey, J.: Exploratory data analysis. (1977)
35. Whitten, A., Tygar, D.: Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0, *Proc. USENIX Security.* (1999)
36. Zviran, M., Haga, W.J.: A comparison of password techniques for multilevel authentication mechanisms. *The Computer Journal* 36.3 (1993): 227-237

Verification Code Forwarding Attack
Short Paper

Hossein Siadati, Toan Nguyen, and Nasir Memon

New York University School of Engineering
{hossein,toan.v.nguyen,memon}@nyu.edu

Abstract. Major Internet service providers deploy *SMS-based verification* mechanisms to fortify the security of users' accounts for critical actions such as password reset and logging in from a new computer. In this paper, we describe a new type of phishing attack where an attacker triggers the delivery of a verification code from a service provider to a user and lures the user to forward the code to him so that he can bypass the SMS verification process. We call this a *Verification Code Forwarding Attack* (VCFA). The attacker can use VCFA to reset a password of a user's account or to get access to a 2-factor enabled account which he already knows its password (e.g., through leaked databases). We attribute the success of this attack to the lack of an effective and usable means for users to verify the service provider, the lack of context for the message sent, and an assumption about users' understanding of the authentication process. To show the susceptibility of the users to such an attack, we conducted an experiment with 20 mobile phone users and found that more than 25% of users were vulnerable against this type of attack. A semi-structured interview with the subjects of the experiment and a survey of 100 subjects on Amazon Mechanical Turk were done to explore possible causes for the success of this type of attack. We also discuss possible remediation.

1 Introduction

Guessable passwords [3], reusing passwords for different accounts [8, 5], breaches of password databases [12, 2], an abundance of malware and the ease of which the devices get infected by trojans and key-loggers easily give attackers access to passwords. As a result, for critical actions such as password recovery and high risk authentication (e.g., log in from a new device) an auxiliary factor is needed to make the system more secure. One prevalent approach adopted by service providers leverages a resource assumed to be in control of the user, such as a phone number or an email address. In one such example of the scheme, advertised as a 2-step or 2-factor verification, the service provider sends a *nonce* in the form of a *verification code* to the user. The user presents this verification code back to the service provider via a channel requested by the service provider. Since the user who is being authenticated is assumed to be in control of this resource, the address of the resource is likely to be unknown to an attacker, and the nonce is random, the requested authentication is then completed. The rationale is that to circumvent such a mechanism, an attacker has to be in *control* of this auxiliary channel or resource. In this work, we show that authentication schemes that utilize such an auxiliary resource can be potentially circumvented without gaining control of the channel but by other means such as social engineering.

For example, we show that a commonly used authentication scheme where a verification code is sent to the user's mobile phone by an SMS message can potentially be compromised by luring the user to send the code to the attacker. We call this a Verification Code Forwarding Attack (VCFA).

In a VCFA attack, the attacker triggers the delivery of a verification code to a victim and shortly after that, the attacker sends a direct SMS to the victim and phishes him or her to forward the code. If the victim forwards the code, the attacker can successfully bypass the SMS verification and get access to the victim's account.

Attackers can use VCFA in three attack scenarios. The first scenario is *password reset*. In this attack, the attacker initiates a password recovery request on a service provider website by entering victim's username or email address and chooses to receive a verification code via SMS message. Shortly after the verification code is sent to the victim, the attacker will phish the victim to steal the verification code and complete the password reset. The attacker needs to know the phone number of the victim in order to phish him or her. For this, the attacker can easily search the public records, social networking websites, data from leaked databases of information or employ social engineering techniques. In a second scenario, the attacker knows the username and password of a victim (perhaps through leaked databases or other hacking techniques), logs into the victim's account from a new machine and then lures the victim to forward the verification code. The access to the victim's account as a result of this attack is at least for one session, but also can be permanent depending on the victim's account settings. The last scenario belongs to *spam account creation* where a fraudster or spammer creates a verified account without giving out any traceable information. In this attack, the spammer enters a random phone number as his verification number at the time of account creation. Then he follows the described steps to phish the verification code.

In this paper, we study this new type of phishing attack and the root causes of why it is successful. In particular, these are our contributions in this research:

- Using a small scale experiment on 20 subjects, we show that more than 25% of users are susceptible to VCFA.
- By conducting a semi-structured interview, we systematically study the reasons why people fall or do not fall for this attack.
- Using a survey on about 100 Amazon Mechanical Turk workers that have enabled SMS-based verification for their Gmail accounts, we validate our findings on a larger and more diverse pool of subjects.

Paper organization: After briefly introducing background in Section 2, we detail our study procedures and findings in Section 3. We discuss the root causes of the problem and possible remediations in Section 4.

2 SMS-based verification and its security

SMS-based verification is a subset of *two-factor authentication* (2FA) mechanisms where a one-time password is used as a second factor for authentication. SMS-based verification is not able to provide security against a phishing attack [14]. The argument is that in a successful phishing attack, the attacker will

lure a victim to enter the one-time password as well. This attack is deployed by attackers in the wild [4]. SMS-based verification also does not provide protection against the existence of malware on mobile devices or workstations [14, 7] because by using the malware, the attacker can capture the one-time passwords as well as hijacking a session after the authentication process is done. The malware attack on SMS-based verification has been in use by attackers [13, 1]. The SMS-based verification, however, provides protection against *known-password-attack* when the user-chose password is known by the attacker, for example based on a leaked database of passwords. For an account protected by SMS-based verification, an attacker who knows the password still can not log in to the account because he does not have access to the verification code. However, such an attacker can launch a VCFA attack to get the user to forward the verification codes, as discussed in this paper.

Several research work have previously studied social engineering techniques and phishing attacks [9–11, 15]. Dhamija et al. [6] have studied the reasons why phishing is successful. Major reasons are visual techniques that the attackers use to deceive users into believing that the URL and the webpage are authentic.

Although there are similarities between the known email-based phishing attacks, Smishing (SMS-based phishing where a phishing link is sent via SMS), and VCFA, there are several differences concerning the reasons for their success and needed countermeasures against them. Firstly, in a VCFA attack, no URL is included in the phishing messages and victims do not need to visit a phishing website. Secondly, a successful VCFA attack needs a victim to forward only a verification code, mostly, a random sequence of digits. In comparison, a victim of traditional phishing attack has to enter widely known sensitive credentials such as password, credit card numbers, or SSN numbers into a website. Thirdly, there are a few indicators such as the sender's email address or the URL of the phishing website that can be used to verify the authenticity of a phishing message and website. In a VCFA attack, however, the victim only has the phone number of the sender and it is much harder to verify the sender of a message based on that alone. These differentiating elements suggest the study of reasons for success of VCFA attack and its remediation.

3 Study Procedures

We conducted this research in three phases. The first phase was a small scale phishing experiment on 20 subjects. Next, we interviewed the subjects. Finally, we extracted a handful of hypotheses from the interviews and evaluated them in a larger scale by surveying 100 subjects on Amazon MTurk.

3.1 Experiment

For the sake of the experiment, we imitated a VCFA attack using messages similar to Google's messages. We bought two 10-digit U.S.A phone numbers, one for imitating the role of a service provider (e.g., Google in our experiment) and the other one for imitating the role of the attacker (e.g., sending phishing message to subjects). The area code for the phone numbers were Mountain View, CA (the area code for Google's headquarters) to make the first message appear more legitimate and the second one more deceptive. We randomly selected 20 subjects

from the contact list of the experimenters. The subjects included 10 males and 10 females, mostly aged between 25-35. 70% of the subjects were students. We were granted an IRB exemption from our institution for this research. We sent two messages to each subject from two different numbers. The first message was: ***"Your Google verification code is [6-digit code]."*** The style of the message exactly followed the Google's message for verification code. It did not include username or any user identifying information. It also did not include the reason why the user is receiving this message. 30 seconds later, we sent the second message: ***"Please verify that your phone is still associated with your Gmail account by replying to this message with the code we have just sent to you."***

Experimental results. 5 out of 20 subjects forwarded the verification codes. This is translated to 25% success for the VCFA attack.

3.2 Semi-structured interview

We interviewed 10 out of 20 subjects of our experiments, 5 of those who fell for phishing and 5 who did not.

Findings. After completing the interviews, we documented the responses and analyzed them to find themes and significant experiences. Because of the space constrains, we only report some of the findings and refer the readers to the long version of this paper.

Subjects listed different reasons for enabling the SMS-based 2FA for their Gmail accounts. The major reason was improving security of their accounts. One subject mentioned that she has enabled the 2FA because of the need for logging into her account from insecure machines at the university's library and laboratories. Since the verification codes are sent to her phone, she thought it was safe to enter her password on potentially insecure computers.

70% of subjects that we interviewed mentioned that they did not pay attention to the phone number that they received the *Message I* from. Indeed, they believed that the message was sent by Google. Apart from two subjects who did not fall for VCFA, other subjects did not notice that the phone number that was used for *Message II* was different from the first phone number. Another interesting finding was that the subjects have seen Google using different phone numbers with different lengths (i.e., short codes vs. 10-digit numbers) for sending verification codes. They did not have a clear understanding of what a Google phone number looks like. These observations explain the core problems of SMS-based verification since SMS system does not provide any effective and usable means for users to verify the sender of messages.

In general, users found the second message (message sent by the attacker) convincing because Google's message does not include any context or reason why the user has received a verification code. Therefore, the second message can alter users' perception and convince them to forward the code. This lack of context is another problem in design of the verification messages in most of the SMS-based verification systems.

3.3 Survey

Following the interviews, we formed a survey in order to measure the extent and prevalence of the insights we gained from the interviews. Our questionnaire was

composed of questions about demographics (i.e., age and gender), and users' usage of SMS-based verification (i.e., frequency of usage of SMS-based verification including "every log in", "log in from a new computer", and "password recovery"). We also asked about the reasons for using SMS-based verification (i.e., "log in from insecure computers" and "account being hacked before.") We asked if they check the phone number of the sender of a verification message. In addition, we asked two different questions to discover the perceptions of users about a VCFA attack in different stages of the attack. Firstly, we asked users if they have ever received an unwanted verification code and what their perception would be if they received one. Then, we asked them to consider a hypothetical scenario in which Google asks them to forward a verification code. Using these questions, we measured the success rate of the VCFA attack on a larger scale. We ran the survey on Amazon MTurk. We asked MTurkers to take survey only if they have enabled SMS-based 2-step verification for their Gmail account. A cleaning process to exclude the unqualified subjects resulted in 98 reliable responses.

Results. 66% of participants in our survey were male, and 90% of participants were between 18 and 35 years old. 8% of participants use SMS-based verification every time they log into Gmail. 66% use it for logging in from a new computer, and about 22% for password reset. We asked users why they chose to use the SMS-based verification. 62% mentioned that they enabled it because they log in from insecure computers, and 22% have enabled it because their accounts have been hacked before.

We asked the participants if they check sender's phone number of verification messages. 38% of participants reported that they check the phone number to make sure it comes from Google. However, 30% of participants did not have any idea about the length of the phone number that Google uses to send the verification codes (i.e., short code vs 10-digit number). 58% of participants believed that they received the verification code from the same number whereas others thought Google uses different numbers to send verification codes. This demonstrates that current settings for SMS-based verification does not offer any effective and usable mechanism for users to verify the sender of the messages.

We asked participants how they would feel if they received an unwanted verification code. 67% of participants believed that it would mean that somebody is hacking their account, 11% believed this is the result of a flaw in the Google's system, 22% of participants did not know why this may happen. We can see that a considerable number of users are not aware of the possibility of a misuse or an attack based on verification codes.

We asked participants what they would do if Google asks them to forward a verification code via SMS. 20% of participants answered that they would forward the verification code, meaning that they would fall for this attack given the fact that Google never asks users to do so. It is important to notice that we notified the participants about the possibility of an attack by adding a choice to answers as follows: "I think somebody is hacking me". Therefore, the expected yield of this attack might be more than 20% in reality.

4 Conclusion

A noticeable number of the users are susceptible to VCFA attack. We attribute the success of this attack to the lack of an effective and usable means for users to verify the service provider and the lack of context for the message sent. Another reason is the assumption about users' understanding of how this authentication process works and their awareness of the possibility of a misuse based on verification codes. A potential quick fix by service providers would be to use a list of publicly announced phone numbers that users should expect to get their messages from. Possible long-term remediation would be to augment a naming system to SMS system so users can see the name of a service provider who sends a message. Another simple fix is to add context to verification code messages indicating why the user received a verification code. Another fix includes appending a warning text such as "DO NOT FORWARD THE VERIFICATION CODE" to remind the importance of the code. The number of subjects in our experiment and the process of recruiting the subjects in this experiment only suit a pilot study. We are conducting a larger scale study to verify our results and to measure the success of suggested list of potential remediation.

References

1. Bankinfosecurity. Malware bypasses 2-factor authentication. http://www.bankinfosecurity.com/malware-bypasses-2-factor-authentication-a-7090/op-1. Accessed: 2015-08-25.
2. J. Bonneau. The gawker hack: how a million passwords were lost. https://www.lightbluetouchpaper.org/2010/12/15/the-gawker-hack-how-a-million-passwords-were-lost/. Accessed: 2015-08-25.
3. J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *SP*, pages 538–552. IEEE, 2012.
4. Citizenlab. London calling: Two-factor authentication phishing from Iran. https://citizenlab.org/2015/08/iran_two_factor_phishing/. Accessed: 2015-08-25.
5. A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang. The tangled web of password reuse. In *NDSS*, 2014.
6. R. Dhamija, J. D. Tygar, and M. Hearst. Why phishing works. In *CHI*, pages 581–590. ACM, 2006.
7. A. Dmitrienko, C. Liebchen, C. Rossow, and A.-R. Sadeghi. On the (in) security of mobile two-factor authentication. In *Financial Cryptography and Data Security*, pages 365–383. Springer, 2014.
8. B. Ives, K. R. Walsh, and H. Schneider. The domino effect of password reuse. *Communications of the ACM*, 47(4):75–78, 2004.
9. T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.
10. M. Jakobsson and S. Myers. *Phishing and countermeasures: understanding the increasing problem of electronic identity theft*. John Wiley & Sons, 2006.
11. M. Jakobsson, A. Tsow, A. Shah, E. Blevis, and Y.-K. Lim. What instills trust? a qualitative study of phishing. In *Financial Cryptography*, pages 356–361. Springer, 2007.
12. J. Kirk. Dating site eHarmony confirms password breach. http://www.computerworld.com/article/2504089/security0/dating-site-eharmony-confirms-password-breach.html. Accessed: 2015-08-25.
13. N. Perlroth. Hackers find way to outwit tough security at banking sites. http://bits.blogs.nytimes.com/2014/07/22/hackers-find-way-to-outwit-tough-security-at-banking-sites. Accessed: 2015-07-20.
14. B. Schneier. Two-factor authentication: too little, too late. *Commun. ACM*, 48(4):136, 2005.
15. M. Wu, R. C. Miller, and S. L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 601–610. ACM, 2006.

Analyzing 4 Million Real-World Personal Knowledge Questions (Short Paper)

Maximilian Golla and Markus Dürmuth

Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Germany
{maximilian.golla,markus.duermuth}@rub.de

Abstract. Personal Knowledge Questions are widely used for fallback authentication, i. e., recovering access to an account when the primary authenticator is lost. It is well known that the answers only have low-entropy and are sometimes derivable from public data sources, but ease-of-use and supposedly good memorability seem to outweigh this drawback for some applications.
Recently, a database dump of an online dating website was leaked, including 3.9 million plain text answers to personal knowledge questions, making it the largest publicly available list. We analyzed this list of answers and were able to confirm previous findings that were obtained on non-public lists (WWW 2015), in particular we found that some users don't answer truthfully, which may actually reduce the answer's entropy.

Keywords: fallback authentication, personal knowledge question, password recovery, password reset, challenge question

1 Introduction

Personal Knowledge Questions (PKQ) are commonly used for fallback authentication, i. e., to recover access to an account when the primary authenticator is lost. Common examples of personal knowledge questions ask for easy-to-remember facts about a user's life, such as the "mother's maiden name" or the "brand of the first car". Previous studies indicated that these questions have indeed better memorability [17], where likely explanations are that (i) no secret needs to be remembered, and (ii) it bases on a cued-recall problem (where the question asked constitutes the cue), in contrast to, e. g., traditional passwords that base on a (pure) recall problem. However, a number of problems limit the usefulness of PKQs: (i) In some instances, the answers to PKQs are available in public databases [7, 13], (ii) answers may be easily guessable by friends and relatives [15], and (iii) the entropy of answers may be low, due to frequent answers or small answer space [15, 4, 13]. In the past, the study of PKQs was hindered by a lack of publicly available real-world data, the only exception being recent work by Bonneau et al. [3], which studied personal knowledge questions at Google, with the data not being publicly available.

In August 2015, a database dump of an online dating service called Ashley Madison was leaked [11]. This data includes about 3.9 million answers to PKQs, which makes it by far the largest set publicly available. The leak is widely considered authentic and contains answers to four different questions (Mother's Maiden Name, Name of High School, Name of Favorite Sports Team, Last 4 Digits of SSN), with 300,000 to 1,500,000 answers per question. In this work, we provide a first analysis of this data, where we focus on the guessability of the answers. We use statistical entropy measures (partial guessing entropy [3]), which assumes perfect knowledge of the distribution of answers, and thus provides an lower-bound on the security offered. Our findings include:

i) The security of knowledge questions is low compared to other knowledge-based authentication methods.
ii) The country of origin and the age of the user influences the strength of the answer for some questions but not for others.
iii) The question about the sports team is the least secure one.

1.1 Related Work

One of the earlier studies on the security of PKQs was conducted in 1993 [17], which found good usability and security. However, by today's standards, the security of PKQs is rather low, as several studies have shown. Griffith et al. showed [7] that the *Mother's Maiden Name*, which is frequently used as a PKQ, can often be derived from public databases, rendering them insecure. Rosenblum has shown that private information about persons can often be inferred from social networking sites [14]. This information can be used to narrow down potential answers for the security questions. Secrecy of those answers in the age of Facebook was studied by Rabkin [13], whereas Bonneau et al. studied the entropy of names [4]. Schechter et al. demonstrated [15] that for a number of such questions the answers can often be guessed easily. A more general discussion on designing security questions including usability, privacy, and security is given by Just [9]. Alternative and a potentially better domain of security questions, namely questions about *personal preference* similar to those used on online dating sites, where studied by Jakobson et al. [8] and have found to provide better security than most other commonly used questions. In recent work, Bonneau et al. [3] have evaluated real-world PKQs at Google and found that some answers are quite predictable, in part because some users do not answer truthfully, which indeed lowers the overall security. This work is the closest to our work; the biggest difference being that our work is based on a public data source and thus is reproducible.

Alternative forms of fallback authentication use a registered email address or mobile phone of the user [6], where an access code is sent to the registered device if the user lost the regular password and requested a password reset. Fallback authentication by support teams is susceptible to social engineering attacks [12]. Social authentication, or vouching, was proposed by [5], a more recent design is given by Schechter et al. [16]. Problems with social authentication were pointed out in [10].

2 Methodology

Attacker Model and Guessing Metrics: We consider an attacker guessing the answer of PKQs without specific knowledge about the user. We consider an idealized attacker that has exact knowledge about the distribution of answers to the questions, modeled by statistical guessing metrics [2]. We are mostly interested in *online guessing attacks*, where we assume that only a very limited number of guesses can be made before the account is locked, or substantial delays are applied to slow down guessing attempts. Resistance against online guessing attacks can be measured by the fraction λ_β of accounts that covers the β most common answers, i. e., an attacker that can make β guesses before being locked out can compromise roughly λ_β of the accounts. For comparability, we follow Bonneau et al. [3] and also list partial guessing entropy G_α, which measures the resistance against offline guessing attacks.

Datasets: The dating website used four different security questions for password recovery: *What is Your Mother's Maiden Name?* (MMN), *What is the Name of Your High School?* (High School), *What is Your Favorite Sports Team?* (Sport Team), and *What are the Last 4 Digits of Your SSN?* (SSN). In total the dataset contained approx. 32 million entries, where 3.9 million had a security answer set.

It has been noted earlier by Newitz [1] that the leaked dataset contained entries from bots, which were used to chat with users that had few "real" contacts. She found a set of criteria to approximately differentiate between datasets that belong to humans and those that belong to bots. Those criteria concern the used email addresses, indicators for the last contact by mail or chat, IP addresses, as well as the existence of account deletion flags within the password hash strings. After filtering for bots, there were 903,255 entries remaining for MMN (out of 1,576,779 before filtering), 632,484 entries for High School (out of 1,031,416), 650,680 entries for Sports Team (out of 1,011,383), and 186,134 entries for SSN (out of 309,827).

Ethical Considerations: The data analyzed in this paper was leaked to the public before, so attackers already had independent access to the datasets. We took care that the data was not distributed any further, and we only present aggregated results in this work that will not leak information about any specific answer.

3 Strength Evaluation

Table 1 shows the full results for all subsets we consider. It lists both λ_β as well as guessing entropy values G_α, with the same parameters used by Bonneau et al. [3] so that the results are comparable to their work.

Sample Size and Significance: To test for statistical significance of our results, we use a slightly simplified re-sampling approach similar to Bonneau et al. [3]. For each set, we repeatedly sample a random subset of 10% the original size, and

		size	online guessing (success in %) λ_1	λ_3	λ_{10}	λ_{100}	λ_{1000}	offline guess. (bits) $\tilde{G}_{0.1}$	$\tilde{G}_{0.25}$	$\tilde{G}_{0.5}$
mother's maiden name										
all		903 255	3.21	5.12	8.18	22.41	48.10	7.34	8.93	10.94
country	USA	612 890	*3.15*	5.31	8.89	24.41	51.74	7.07	8.64	10.42
	Canada	125 101	-	*4.09*	*6.91*	*21.64*	*48.49*	7.68	9.03	10.86
	UK	26 912	-	-	-	-	-	*6.27*	*7.97*	*9.53*
age	<= 35	169 571	*2.65*	*4.06*	*6.79*	20.73	*46.37*	7.79	9.19	11.18
	36 – 45	293 868	*3.08*	*5.12*	8.08	22.63	48.33	7.35	8.90	10.90
	45 – 55	284 594	*3.48*	*5.55*	8.82	23.38	49.36	7.08	8.79	10.75
	> 55	155 222	*3.58*	*5.50*	*8.91*	23.74	*49.78*	7.07	8.74	10.69
high school										
all		632 484	2.63	5.60	7.86	16.91	37.88	7.78	10.04	11.93
country	USA	461 582	*2.68*	6.03	8.55	18.25	42.61	7.36	9.67	11.39
	Canada	86 465	-	-	*9.83*	*26.79*	*63.56*	6.72	8.32	9.59
	UK	8 740	-	-	-	-	-	-	-	-
age	<= 35	127 707	*2.56*	*5.76*	*7.92*	*16.45*	-	7.79	10.30	*12.28*
	36 – 45	217 157	*2.57*	5.51	7.72	16.88	*37.90*	7.85	10.04	11.92
	45 – 55	194 608	*2.80*	*5.53*	7.87	17.27	*40.21*	7.72	9.90	11.63
	> 55	93 012	-	*5.86*	*8.39*	*18.60*	-	7.43	9.63	11.25
sports team										
all		650 680	2.83	7.84	19.44	62.94	89.68	5.39	5.82	6.58
country	USA	432 129	4.11	10.75	26.51	74.15	93.08	4.79	5.20	5.75
	Canada	85 520	*11.16*	18.61	36.03	74.61	91.44	3.16	4.39	5.30
	UK	12 011	-	*23.23*	*41.93*	*73.49*	-	*3.50*	*3.87*	*4.86*
age	<= 35	128 520	-	*6.03*	16.38	58.78	86.98	5.75	6.16	6.90
	36 – 45	250 901	*2.73*	7.69	19.73	63.39	89.91	5.43	5.77	6.53
	45 – 55	199 321	*3.07*	8.67	21.94	65.83	91.40	5.15	5.60	6.35
	> 55	71 938	-	*9.48*	23.49	66.95	*91.73*	5.05	5.48	6.25
SSN (4 digits)										
all		186 134	-	*5.32*	*7.07*	-	-	*9.91*	*12.15*	*12.70*
country	USA	128 611	-	*5.15*	*6.73*	-	-	-	*12.17*	*12.65*
baseline										
password (RockYou) [3]			0.9	1.4	2.1	4.6	11.3	12.8	15.9	19.8
4-digit PIN (iPhone) [3]			4.3	9.2	14.4	29.3	56.4	5.2	7.7	10.1
4-digit random PIN			0.01	0.03	0.1	1.0	10.0	13.29	13.29	13.29

Table 1. Full listing of the results.

test if the resulting entropy values fall within a 5% or a 10% error band with a confidence of at least $p = 0.98$. Values that are in the 5% band are shown in non-italic, those that fall in the 10% band are shown in italic. All other values are omitted from the table.

Differences in Questions: First, we compare the four different questions available. We found that the results for λ_1 are surprisingly consistent over all tested questions and subsets. However, higher values, e. g., λ_{10}, show measurable differences. For example, we have $\lambda_{10} = 19.44$ for the sports team question, while the other three questions all have a λ_{10} of around 8. This is also reflected in the guessing entropy, with a $\tilde{G}_{0.25}$ of 5.82 for sports team, and 12.15 for the SSN.

Differences in Nationality: When considering different nationalities, we observe that mother's maiden name and high school show surprisingly little variation. For sports team, interestingly, knowing the nationality has substantial influence on the guessability of the answer (we have λ_1 of 2.83 if nationality is unknown, and values between 04.11 and 11.16 once the nationality is known). In addition, for Canada the sports teams are slightly easier to guess than for the US. The SSN is substantially less secure when a person is from outside the US. This is most likely explained by the fact that the SSN is specific to the US, and answers from non-US citizens contain a substantial fraction of fake answers, a behavior which was similarly observed in previous work [3]. For the complete set of the SSN each answer should occur, assuming a uniform distribution, with probability 0.01%, but the most popular answer (1234) occurred with probability 2.23%. Other frequent answers include 1111, 0000, and 6969.

Differences in Age: The data suggests that the age has little influence on the guessability; for mother's maiden name and SSN the differences are practically non-existent. For high school and sports team, however, we find that with increasing age the answers are slightly easier to guess, which means that there is less variability (e. g., for sports team we have $\lambda_{10} = 16.38$ for those under 35 and $\lambda_{10} = 23.49$ for those over 55).

Baseline: As baseline, we added entropy values for passwords (using the RockYou list), for real-world PINs, and for randomly chosen PINs; the entropy values for the first and second one are taken from [3]. These provide an interesting perspective: We see that all knowledge questions in the dataset are substantially less secure than a randomly chosen 4-digit PIN, however, they are slightly more secure than a real-world PIN chosen by a human.

4 Conclusion

We have analyzed the answers to personal knowledge questions given in the data leaked from Ashley Madison in August 2015. We found that favorite sports teams are particularly easy to guess, that the security depends to a certain extent on the age and the origin of a user, and that in general they only offer a low level of security.

References

1. Annalee Newitz – Gawker Media. Ashley Madison Code Shows More Women, and More Bots, August 2015. Online at http://gizmodo.com/ashley-madison-code-shows-more-women-and-more-bots-1727613924, as of December 5, 2015.
2. Joseph Bonneau. The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. In *IEEE Symposium on Security and Privacy*. IEEE, 2012.
3. Joseph Bonneau, Elie Bursztein, Ilan Caron, Rob Jackson, and Mike Williamson. Secrets, Lies, and Account Recovery: Lessons from the Use of Personal Knowledge Questions at Google. In *International World Wide Web Conference*. IW3C2, 2015.
4. Joseph Bonneau, Mike Just, and Greg Matthews. What's in a Name? Evaluating Statistical Attacks on Personal Knowledge Questions. In *Financial Cryptography and Data Security*, volume 6052 of *LNCS*, pages 98–113. Springer, 2010.
5. John Brainard, Ari Juels, Ronald L. Rivest, Michael Szydlo, and Moti Yung. Fourth Factor Authentication: Somebody You Know. In *ACM Conference on Computer and Communications Security*, pages 168–178. ACM Press, 2006.
6. Simson L. Garfinkel. Email-Based Identification and Authentication: An Alternative to PKI? *IEEE Security and Privacy*, 1(6):20–26, 2003.
7. Virgil Griffith and Markus Jakobsson. Messin with Texas: Deriving Mothers Maiden Names Using Public Records. In *Applied Cryptography and Network Security*, volume 3531 of *LNCS*, pages 91–103. Springer, 2005.
8. Markus Jakobsson, Erik Stolterman, Susanne Wetzel, and Liu Yang. Love and Authentication. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 197–200. ACM Press, 2008.
9. Mike Just. Designing and Evaluating Challenge-Question Systems. *IEEE Security and Privacy*, 2(5):32–39, 2004.
10. Hyoungshick Kim, John Tang, and Ross Anderson. Social Authentication: Harder Than It Looks. In *Financial Cryptography and Data Security*, volume 7397 of *LNCS*, pages 1–15. Springer, 2012.
11. Kim Zetter – Wired. Hackers Finally Post Stolen Ashley Madison Data, August 2015. Online at http://www.wired.com/2015/08/happened-hackers-posted-stolen-ashley-madison-data/, as of December 5, 2015.
12. Kevin D. Mitnick and William L. Simon. *The Art of Deception: Controlling the Human Element of Security*. John Wiley & Sons, 2002.
13. Ariel Rabkin. Personal Knowledge Questions for Fallback Authentication: Security Questions in the Era of Facebook. In *USENIX Symposium on Usable Privacy and Security*, pages 13–23. USENIX Association, 2008.
14. D. Rosenblum. What Anyone Can Know: The Privacy Risks of Social Networking Sites. *IEEE Security and Privacy*, 5(3):40–49, 2007.
15. Stuart Schechter, A. J. Bernheim Brush, and Serge Egelman. It's No Secret. Measuring the Security and Reliability of Authentication via "Secret" Questions. In *IEEE Symposium on Security and Privacy*, pages 375–390. IEEE Computer Society, 2009.
16. Stuart Schechter, Serge Egelman, and Robert W. Reeder. It's Not What You Know, But Who You Know: A Social Approach to Last-Resort Authentication. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 1983–1992. ACM Press, 2009.
17. M. Zviran and W. J. Haga. A Comparison of Password Techniques for Multilevel Authentication Mechanisms. *The Computer Journal*, 36(3):227–237, 1993.

What Lies Beneath? Analyzing Automated SSH Bruteforce Attacks

AbdelRahman Abdou[1], David Barrera[2], and Paul C. van Oorschot[1]

[1] Carleton University, Canada
[2] ETH Zürich, Switzerland

Abstract. We report on what we believe to be the largest dataset (to date) of automated secure shell (SSH) bruteforce attacks. The dataset includes plaintext password guesses in addition to timing, source, and username details, which allows us to analyze attacker behaviour and dynamics (*e.g.,* coordinated attacks and password dictionary sharing). Our methodology involves hosting six instrumented SSH servers in six cities. Over the course of a year, we recorded a total of ∼17M login attempts originating from 112 different countries and over 6K distinct source IP addresses. We shed light on attacker behaviour, and based on our findings provide recommendations for SSH users and administrators.

1 Introduction

Internet accessible secure shell (SSH [18]) servers are consistently flooded with credential guessing attempts originating from a wide range of globally-distributed hosts. Every login attempt generates an entry in a log file which conscientious system administrators painstakingly monitor. While some of these guesses target specific systems and user accounts, most aim to identify weak authentication configurations on the entire public IPv4 address space every minute of every day. This status quo of seemingly endless login attempts has led to a cottage industry of software tools, recommended service configuration, and a folklore for throttling the frequency (and corresponding log entries) of such attacks.

In principle, sensible configurations (*e.g.,* rate-limiting login attempts, disabling remote logins for privileged accounts) of remotely accessible services in conjunction with strong password selection by users should suffice in limiting the success of such attacks. However, the low cost of executing these attacks from geographically-diverse datacenters or from compromised end user systems (*e.g.,* as part of a botnet) has apparently made these guessing attacks viable for attackers. Administrators also appear hesitant to deploy restrictive network access control (*e.g.,* IP address range-blocking) configurations which may lead to increased service desk call volume or poor user experience.

Well known software tools exist (many are free and open source) to monitor and limit the number of login attempts that can be performed on a given server. These tools usually maintain state measuring login attempts per source IP address, or login attempts from any source within a defined period of time. Once

some pre-defined threshold is met, future login attempts are blocked or throttled, either for a specified time period. Popular SSH server implementations (*e.g.,* OpenSSH and Dropbear) ship with default configurations that limit the number of allowed authentication attempts per session (*e.g.,* disconnecting the remote system after three unsuccessful attempts) and disable remote logins for privileged accounts. These settings can be easily configured by administrators. Despite the widespread availability of tools and configurations, SSH bruteforce login attacks are on the rise [4,1], supporting the theory that some of these login attempts must be succeeding.

While academic work in the field has identified attacker use of common usernames and passwords [14], and also focused on detection of highly distributed or stealthy attacks [13], to our knowledge, there is little academic work in understanding the anatomy of run-of-the-mill automated SSH bruteforce attacks. Understanding these attacks can shed light on how attackers create and use password dictionaries, guessing strategies, and adaptive techniques. Insight on attacker behaviour can help design and improve defensive techniques, and demonstrate the ineffectiveness of others. This paper makes the following contributions:

- We describe a data collection methodology for SSH login attempts that allowed us to record over 17 million guesses including over 1.4 million plaintext passwords and nearly 28 thousand usernames. The vast majority of these login attempts were performed by automated software tools.
- We perform an in-depth analysis of these login attempts by analyzing two datasets; the first contains log entries collected on a single system over a one-year period; the second includes logs from five servers over a 10-week period. Among others, our analysis offers insights into guessing strategies, password dictionaries used, and data sharing amongst attackers.

The sequel is organized as follows. Section 2 presents related work. Section 3 describes our data collection methodology and initial dataset observations. In Section 4, we detail characteristics of attacking sources including network location and number of attacker IPs per network block. Section 5 analyzes password guesses, including composition, re-use, and password sharing among sources. Section 6 covers distribution of usernames and password guesses per username. Section 7 provides details on timing dynamics. We discuss recommendations for users and administrators in Section 8, and conclude in Section 9.

2 Related Work

Host-based detection of bruteforce attacks. By default, SSH server software generates log entries when authentication requests are received. An entry includes timestamp, source IP address, source port, username, and the result of the authentication request (*e.g.,* success, incorrect password, invalid user, *etc.*). These logs can be locally monitored by client software, such as Blockhosts (`www.aczoom.com/blockhosts`), Fail2Ban (`www.fail2ban.org`), or DenyHosts

(`www.denyhosts.sourceforge.net`), to detect attacks by observing recent login attempts and blocking sources that exceed a threshold of login attempts per time period. Source blocking can be enforced at the network level (*i.e.,* by adding network firewall rules) or at the application level (*e.g.,* by using `tcpwrappers`). Certain tools allow submitting failed attempts to centralized services for aggregation and analytics [1,17].

Analysis of Passwords used in SSH Attacks. Most closely related to our work, Owens and Matthews [14] collected around 103,000 login attempts on three honeypots over an 11 week period in 2007-2008. The authors report an overwhelming majority of attempts targeting the `root` account, and nearly 49% of all attempted guesses had the username equal to the password. Compared to the size of the dataset of Owens and Matthews, the one collected herein is an order of magnitude larger. In addition, our attacker labelling methodology is more comprehensive by considering usernames, passwords, and timing independently, which allows us to identify dictionary re-use regardless of dictionary sizes or the order of password guesses within.

Network-based detection of SSH bruteforce attacks. Javed *et al.* [13] propose a methodology to detect distributed and potentially stealthy SSH guessing activity. Hofstede *et al.* [11] propose a large-scale system to detect SSH compromises (*i.e.,* successful guessing attempts) using NetFlow data. Satoh *et al.* [15] similarly use network flow data to identify the source's authentication type (interactive, key-based, *etc.*) in an effort to remove non-automated logins from their set. We achieve the same server behaviour by configuring the server to exclusively accept password authentication (see Section 3). Sperotto *et al.* [16] built Hidden Markov Models from an SSH bruteforce attack recorded at their university, and used this model to generate synthetic SSH network traces. The authors found that the generated traces can be used to simulate a ground truth for the evaluation of defense systems. We believe our dataset to be a ground truth of SSH dictionary attacks due to its construction.

3 Data Collection Methodology

All logging was performed on Ubuntu 12.04 or 14.04 virtual machines (VM) running on a popular cloud server platform, which assigned public and persistent IPv4 addresses to each VM at creation. IPv6 connectivity was not enabled. No publicly facing network services were enabled on the VMs other than two SSH daemons (one for data collection and one for administration), and software firewalls were configured to allow all inbound and outbound traffic.

The pre-installed SSH daemon served as the management interface for the VMs. To prevent guessing attempts against this interface, we moved the daemon to a non-standard TCP port and we disabled password-based authentication (requiring key-based logins). In Appendix A, we describe a small-scale experiment to determine if attackers target SSH servers on non-standard ports.

We installed a second instance of the OpenSSH server (version 6.5p1) on all VMs to log guessing attempts. To additionally record the password guesses,

we modified the OpenSSH server by inserting a log function in the password authentication module (`auth-passwd.c`). This modified server was configured to start at boot and listen for incoming authentication requests on the default TCP port 22. The server configuration was also changed to allow up to 50 (from the default 10) concurrent unauthenticated connections to the daemon. Only password authentication was allowed on the modified daemon. All login attempts were logged to the `syslog` logging facility.

Preventing accidental human logins. Our paper focuses on the analysis of automated login attempts. Thus, we wish to minimize the probability of recording login information from non-automated sources. To prevent accidental non-automated logins (by either curious users or by accidentally typing the wrong IP address or other misconfiguration), we displayed the following SSH banner to all incoming authentication requests prior to password entry:

```
*WARNING*
This OpenSSH server has been modified to STORE USERNAMES AND PASSWORDS.
This server does not have any valid user accounts, so no attempted logins
will succeed. The sole purpose of this server is to collect (for research
purposes) login information used in automated SSH brute-force attacks. If
you are human, you should not attempt to log in to this server.
```

Data collection. We began collecting login attempts on a single VM running in Ottawa (OTT), Canada on Mar 1, 2014. On Jan 4, 2015, we instrumented and enabled five additional VMs, each in a separate geographical region: San Francisco (SFO), New York (NY), London (LON), Amsterdam (AMS), and Singapore (SGP). These five VMs collected data for only 66 days, but broadened our geographical scope allowing us to make location-specific observations. Since these additional VMs were on distinct networks, they allowed us to detect sources which target multiple destination IPs. All data collection was halted on Mar 8, 2015, giving a total of 373 days of aggregate data collection. Throughout the paper, we report on the aggregate set of login attempts performed on all VMs, clarifying VM-specific observations where necessary.

4 Characteristics of Attacking Systems

Table 1 summarizes the collected login attempts. Collectively, the VMs received ∼17M attempts from ∼6.2K IP addresses located in 112 countries.[3] A total of ∼27K distinct usernames and ∼1.4M distinct password guesses were observed.

A single /24 subnet (`103.41.124.0`/24) based in Hong Kong (HK) was observed guessing credentials aggressively on the OTT VM beginning Nov 15, 2014, and later on all five VMs of the short-term study within two days of the VMs coming online (Jan 4, 2015). This subnet alone was responsible for 8,757,604 (or ∼51%) of all recorded guessing attempts, from 65 observed host addresses.

[3] We used the `http://ipinfo.io` IP geolocation database [12] to obtain geographic location and Autonomous System (AS) information of these IP addresses.

Table 1. A summary of the collected data.

VM	Attempts	AS	IP addresses				Regions		Distinct	
			/8	/16	/24	/32	Country	City	Usernames	Passwords
OTT	9,925,706	934	164	1,735	2,934	4,233	100	580	25,551	1,209,851
LON	2,609,164	357	125	573	826	1,134	66	197	2,742	618,869
NY	1,661,628	245	123	426	508	707	61	137	1,637	464,456
SFO	1,491,949	250	125	444	548	760	58	152	2,160	492,340
AMS	1,107,247	353	126	547	803	1,065	64	199	2,505	314,384
SGP	421,982	333	132	525	834	1,114	62	185	4,520	135,015
*	17,217,676	1,235	171	2,338	4,187	6,297	112	744	27,855	1,449,146

$*$ = Combined statistics.

Because the OTT VM collected attempts for the longest time period, it received the largest number of login attempts. The corresponding source IP addresses belong to 934 ASes and 100 countries.

For the short-term study, the five VMs experienced different login attempt rates. The LON VM received the highest rate, logging ∼2.6M attempts in 10 weeks. The SGP VM received less than one-sixth of those attempts, albeit from almost the same number of IP addresses (1,114 vs. 1,134). The reason for this discrepancy is unclear; the aggressive HK subnet discovered both VMs on the same day, Jan 5 (*i.e.,* the day we started observing login attempts from that subnet). Of the 65 host addresses belonging to the HK subnet, 64 made attempts on the LON VM, and 62 similarly on SGP. The two hosts not showing interest in the SGP VM were among the least aggressive in the subnet, responsible only for 0.3% and 0.9% of attempts made by the entire subnet.

Attackers may seek to compromise systems in specific regions, and the discrepancy between attacks seen on LON and SGP VMs suggests that attackers may actually be distributing their resources unevenly, perhaps for higher benefits (or smaller risk of being shut down).

4.1 Number of IPs per /24

Recall from Table 1 that we logged ∼4.2K distinct /24 subnets. About 84% (3,528) of those had only one host IP address recorded in our logs. On the other hand, we observed 76 hosts in `116.10.191.0`/24, making it the subnet with the largest number of *malicious*[4] hosts. This subnet was responsible for 7.5% (1,298,912) of all guessing attempts. We observed 626 subnets with between 3 and 10 malicious hosts, and 33 subnets with 11 or more malicious hosts.

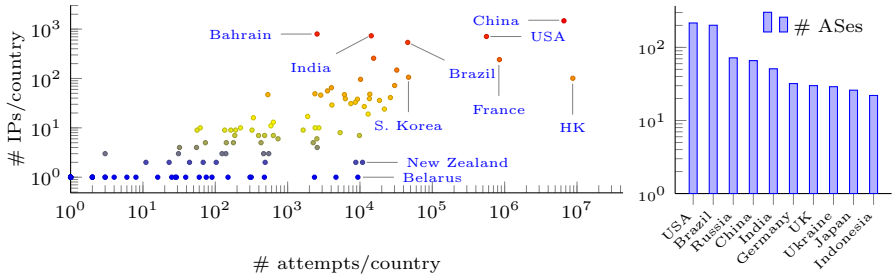
4.2 Countries with the most aggressive sources

Figure 1a shows the relationship between the number of IP addresses observed per country, and the number of attempts thereof. As expected, the chart reveals

[4] Although a logged IP address may not necessarily belong to a user with deliberate malicious intent (*e.g.,* it could be remotely exploited by a malicious third party) we refer to the IP as such for simplicity.

a positive correlation between these factors. Together, Hong Kong and China (two right most points) constitute ∼90% of all guessing attempts observed.

China also comes first in the list of countries by number of observed IP address (highest point), followed by Bahrain despite the country's relatively small IP address allocation—∼450K IPs [3] (see Section 4.3 below). Additionally, analyzing the collective number of attempts per /24 subnets, 7 of the 10 most aggressive (by number of attempts) subnets are Chinese; the remaining three are from Hong Kong, USA and France.



(a) (b)

Fig. 1. (a) The number of attempts, per country, with respect to IP addresses. For example, the point (2552, 796) indicates that a country (which is Bahrain) originated 2,552 attempts from 796 IP addresses. (b) The number of ASes per top 10 countries.

The distribution in the number of ASes observed per country is shown in Fig. 1b. UK, Ukraine, Japan and Indonesia show up in the top 10 countries with highest number of ASes, while absent from the list of top 10 by number of IP addresses. This suggests that malicious machines in those countries were more scattered amongst varying ASes, rather than concentrated in a few. Countries in which malicious hosts reside in a small number of ASes can more easily coordinate patching efforts. For example, although Bahrain comes second in the list of countries by number of IP addresses (796 addresses), all such addresses belonged to three ASes (six /8 subnets). On the other hand, we saw 538 Brazilian addresses belonging to 201 ASes, requiring involvement from more ISPs to fix.

4.3 IP addresses as a ratio of the total allocation per country

We analyze each country's proportion of malicious IP addresses relative to the total addresses allocated to the country. For example, Panama is allocated a total of 1,909,212 IP addresses [3] of which we observed only 6 in our logs, giving a ratio of 0.31 address per 100,000.

Bahrain tops the list of countries with the largest proportion of malicious addresses, at ∼177 IP per 100K allocated. Figure 2 shows a geo-chart with the number of IP addresses observed per 100K allocated for each country. Note that

Bahrain is not represented in this chart, as it overshadows the rest of the data. Additionally, we exclude from the chart data of 10 countries allocated less than 100K addresses in total.

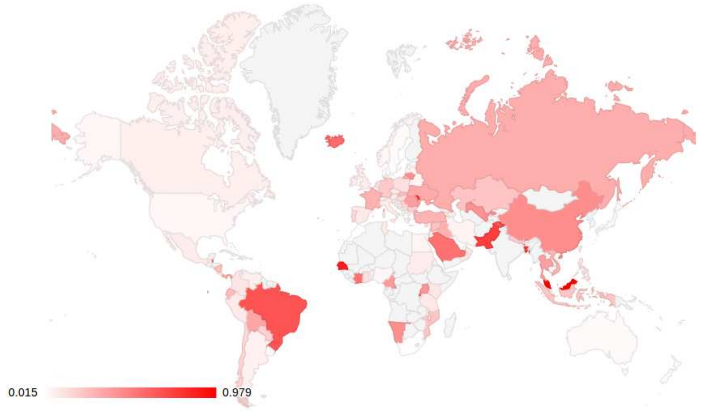


Fig. 2. Ratio of malicious addresses to 100K allocated. *Created using the Google Chart libraries, according to terms described in the Creative Commons 3.0 License.*

5 Password Analysis

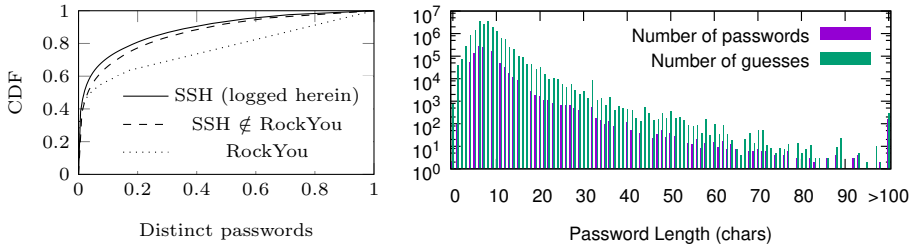
Figure 3a shows the Cumulative Distribution Function (CDF) of all observed password guesses, and the CDF of those not appearing in the RockYou (`www.rockyou.com`) dataset. Both distributions are highly skewed. Under the hypothesis that these distributions reflect previously successful SSH guesses, their skewness suggests the potential existence of accounts with poorly- (or carelessly-) chosen passwords. Passwords of such accounts would require less guess work [6] to crack than those in the RockYou dataset—CDF also plotted in Fig. 3a.

5.1 Password Length

Figure 3b plots the distribution of password lengths next to the number of passwords seen for each lengths. The distributions resemble each other, with more passwords of a given length being reflected as more login attempts for passwords of that length. Passwords varied in length from 0 (no password received) to 270. We observed several hosts in a single network attempting passwords that appear to be Unix shadow file entries (118 characters long), indicating that attackers may not always inspect the correctness of their dictionary entries.

5.2 Password composition compared to known dictionaries

In Table 2, we show the password composition of the 1.4M passwords in our set. Over 22% of password guesses observed were constructed as one or more



(a) (b)

Fig. 3. (a) CDF of distinct password guesses; a point (x, y) means the proportion x of distinct passwords accounted for the proportion y of all observed. (b) Password length distribution. For each password length, the bar on the left shows the number of unique passwords of that length in our set. The bar on the right shows the number of login attempts seen with a password of that length.

letters followed by one or more numbers (*e.g.,* `test123` or `admin6`). More than half of all passwords contained only lowercase characters. With the exception of only lowercase passwords,[5] the password composition of passwords in our set resembles that of the RockYou dataset, where the vast majority of passwords include only alphanumeric characters (*i.e.,* no special characters), and a large set of passwords was composed by lowercase characters followed by numbers.

Table 2. Password composition: observed per experiments herein versus RockYou.

Password Type	SSH•		RockYou dataset	
	Count	%	Count	%
Only lowercase	771,101	53.2	3,783,103	26.4
Only uppercase	5,883	0.406	234,913	1.64
Only numbers	140,074	9.67	2,348,128	16.4
Letters then numbers	325,547	22.5	5,340,129	37.2
Have no special characters	1,372,858	94.7	13,395,174	93.4
Have special characters	76,288	5.26	949,217	6.62
Total	1,449,146	100	14,344,391	100

•per experiments herein

We noticed more than 3.2K distinct passwords in the form of URLs (*e.g.,* ending in `.com`, or `.edu`), which were collectively tried ∼140K times. The most common of those ending in `.com` and `.net` respectively were `123.com` (attempted 7,014 times) and `nowtop.net` (971 times).

[5] Attackers may guess only lowercase passwords more frequently in expectation that system administrators pick these types of passwords more often.

Many of the attempted passwords have not been seen in the commonly studied leaked dictionaries (*e.g.,* RockYou and phpbb). For example, out of the ∼1.4M distinct passwords, 876,012 (60%) passwords were not present in the RockYou dataset. Those passwords were collectively attempted 7,136,356 times out of the ∼17M total attempts (or ∼41%). Examples of those passwords include `http://managers.at` and `CactiEZ`. See Appendix B for more details on the top ten passwords and their counts.

Note that the number of guesses of many passwords unique to our dataset was relatively high. For example, the password `idcidc` (likely short for "I don't care") was attempted 5,272 times (0.03% of all ∼17M attempts) from 533 IP addresses (of all 6,297) belonging to 27 ASes and 11 different countries. This information increases the likelihood of those ∼5K attempts originating from different attackers using shared lists. Note also that those attempts were (almost) evenly distributed over the course of 13 months. Table 3 shows five examples of passwords that did not appear in any of the following leaks: RockYou, GawkerPasswords, myspace, phpbb, SonyPasswords, and YahooVoicePasswords.

Table 3. Examples of passwords observed in our experiments but not appearing in common leaked dictionaries (see inline for details). The column to the right shows the number of days the password appeared in our logs.

Password	# of attempts	# of sources			# of days (months)
		IPs	ASes	Countries	
`o12nu27`	5,425	512	45	17	369 (12)
`idcidc`	5,272	533	27	11	368 (12)
`rkqldk`	4,457	381	9	5	349 (11.4)
`\\001`	3,837	26	3	1	341 (11.1)
`zxm10`	3,168	465	32	13	26 (0.85)

5.3 Dictionary Sharing and Splitting Among Sources

Owens and Matthews identified password sharing among bruteforcers [14]. Their work used a narrow definition of sharing that required distinct attackers to attempt the same username-password pairs in the same order to qualify as sharing. While we also observed same-order guesses as described by Owens and Matthews, we also noticed random guessing from shared dictionaries (described below).

One possible way to detect dictionary sharing is to identify overlap between dictionaries. Figure 4 shows a heatmap of the percentage overlap between pairs of dictionaries in our set. For this graph, dictionaries are built as the aggregate set of distinct passwords used by any source within a particular /24 network. We take the top 1000 largest (by password count) per-/24 dictionaries, and sort them from largest to smallest on the x and y axes.[6] For reference, the

[6] In Fig. 8, we show a similar heatmap for overlap between the largest 1000 per-IP dictionaries (*i.e.,* passwords seen used by each IP).

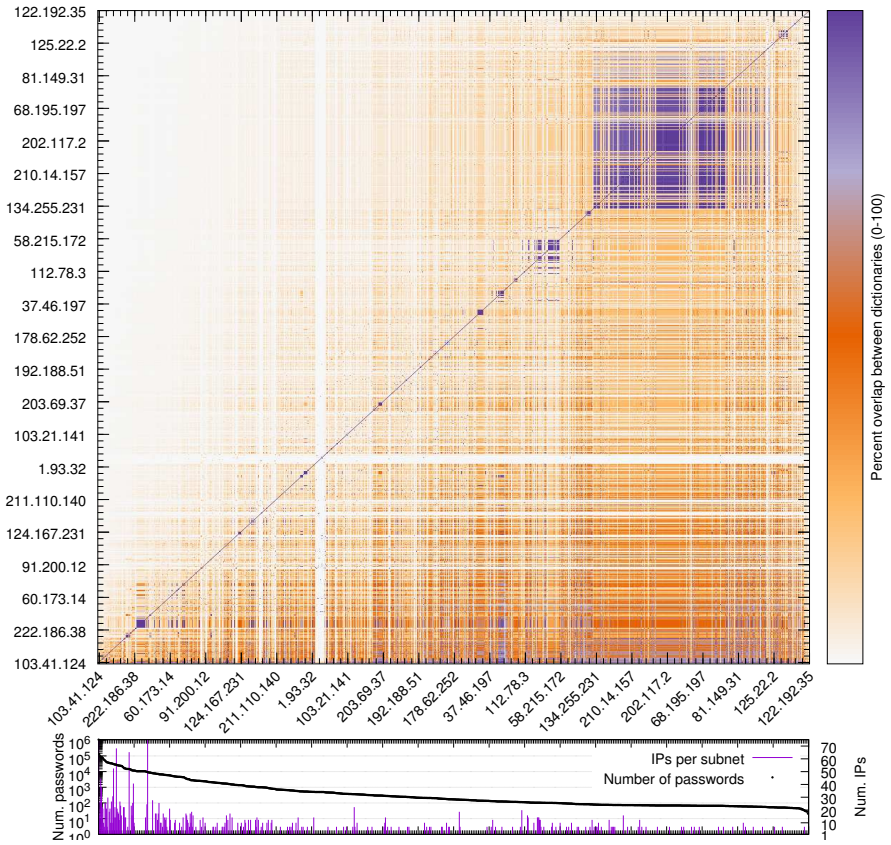


Fig. 4. Overlap between per-subnet dictionaries. See inline. Best viewed in colour.

largest subnet dictionary (*i.e.,* passwords used by `103.41.124.0`/24) contains ∼1.09M distinct passwords. The 1000^{th} largest dictionary (seen used by sources in the `122.192.35` subnet) contains 21 distinct passwords. As the colour gradient shows, white indicates no overlap (*i.e.,* no common entries) between a dictionary on the x-axis and that on the y-axis, while purple indicates 100% overlap.

Coordinated dictionary split. The horizontal and vertical white bands near the `1.93.32` subnet indicate that passwords guessed were mostly unique compared to all other passwords. Upon further investigation, we identified 13 sources each guessing from a set of 500 lowercase alphabetical passwords. 11 of the 13 sources showed no overlap between their 500 passwords and other sets in the group. This lack of overlap and equally sized dictionaries shows with high probability that a single attacker split a large dictionary into smaller sets of 500 entries, distributing these small sets to bots under his control.

Dictionary sharing. Purple squares near the $x = y$ line highlight (possibly non-contiguous) /24 subnets using the same set of passwords. We identified many

such instances, with the most notorious being the large purple region near the top right of the heatmap. Here, we identified 404 distinct sources from 54 countries all guessing from a dictionary of around 90 alphanumeric strings. These guesses also came from a set of around 90 distinct username-password combinations, indicating coordination among sources.

Another example is the largest purple region near the bottom left. Here 12 distinct networks were seen using the same dictionary of 10,546 passwords. Sources in this set also each performed the same number of guesses overall (10,852), and each guessed each password in their dictionary the same number of times (*e.g.,* `123456` 23 times, `password` 14 times, `123` 11 times, *etc*). While this behaviour may hint at coordination or centralized control, it could also reflect distinct attackers using software configured the same way (*e.g.,* by not modifying the default configuration or password list).

The predominant purple and orange shading suggests that dictionary sharing or distribution takes place among attackers.[7] Small sets of passwords, likely bundled with bruteforcing tools or found online, are used by many attackers. However, as we discuss in Section 5.2, the overlap of these passwords with common web-based leaked dictionaries (*e.g.,* RockYou) is not significant. One possible explanation is that attackers may be running SSH servers on their botnets to log guessing attempts made by other attackers. Such a strategy would allow attackers to effortlessly and quickly expand their dictionaries.

5.4 Reattempting username-password combination

Recall that none of our VMs had any valid accounts, so all login attempts failed for all guesses. Despite this, we noticed that some username-password guesses were made by the same source on the same destination VM more than once. For example, the username `nano` and password `bl3rand,.` were tried four times against the LON VM from a machine in Baotou, China between Jan and Feb 2015. About one-third of all source IP addresses (2,019 of 6,297) manifested that behavior (77 addresses were particularly responsible for ∼50% of all repeated attempts). Those addresses belong to 1,189 of all 4,187 /24 IP addresses. In total, such *repeated attempts* account for 25% (or ∼4.3M of ∼17M) of all guessing attempts, with time between pairs of repeated attempts ranging from less than one second to ∼11 months. Attempts repeated more than twice also occurred, as evidenced by an address that guessed username `root` and password `\\001` 1,220 times against the OTT VM in only 19 minutes.

One possible explanation for repeating attempts is that a machine could be controlled by multiple attackers simultaneously. This may happen when attackers, upon compromising a new system, fail to patch the vulnerability they

[7] We believe it is unlikely that all such highly overlapping dictionaries belong to a single attacker since many of their bruteforcing behaviors were different, *e.g.,* timing dynamics, rate of attempts, *etc.* Even dictionary pairs with extreme overlap had different guessing order.

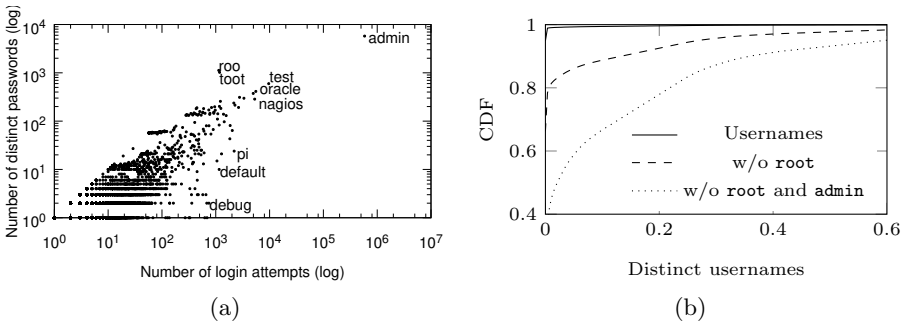


Fig. 5. (a) Number of login attempts and distinct passwords per username (excluding root). (b) A CDF of usernames; a point (x, y) means the proportion x of distinct usernames accounted for the proportion y of all ∼17M observed.

exploited.[8] Attackers may also be repeating attempts to account for password aging policies [7], expecting previously-failed attempts to succeed due to a password change. Although both hypotheses above could explain attempts repeated within months or weeks, they fail to explain repeated attempts within shorter intervals (*e.g.,* minutes or seconds). We found that the time between 59,514 (1.4%) pairs of repeated attempts was one second or less, and between 251,305 (5.7%) was one minute or less, hinting at attack software misconfiguration.

Another explanation is that attackers may not have enough resources to store which username-password pairs were attempted on which SSH server. Thus, they operate a herd of bots arbitrarily, where any bot guesses a combination (from the same fixed list of candidate guesses) on any server. Machines showing extremely large average time (*i.e.,* multiple months) between repeated attempts could have gone through three phases: compromised, patched, then re-compromised (either by the same attacker, or another one with overlapping dictionary). The source may also have completed a full cycle through a list of targets and passwords, arriving at the starting point once again.

6 Username Analysis

Root and Admin. The most commonly attempted usernames were `root` (∼95% of all attempts) and `admin` (∼3%). Interestingly, only 63% of source IP addresses guessed passwords for `root` or `admin`, meaning that the remaining 37% specifically targeted non-administrative accounts, though at a much slower rate. Figure 5a plots the number of login attempts observed for a specific username on the x-axis, and the number of distinct passwords for that username on the y-axis. Based on this behaviour, defensive strategies that block the IP addresses

[8] Attackers may be unwilling to change the password or patch the vulnerability used to compromise to avoid detection by the legitimate user of that system.

attempting `root` will fail to block a large portion of sources, but will succeed in blocking the vast majority of attempts (until attack behaviour changes).

Figure 5b shows a CDF of username distribution. We note that the skew in the CDF even when excluding `root` and `admin` indicates that attackers target a narrow set of usernames. We discuss the implications of this in Section 8.

Other Usernames. The remaining set of usernames were tried comparatively fewer times with fewer distinct passwords. Out of 27,855 total usernames, 7,612 (27.33%) saw only a single login attempt with a single password; 1,369 usernames (4.91%) saw 2 attempts with a single password; and 7,340 (26.35%) usernames recorded 2 attempts with 2 distinct passwords. This trend does not continue, since only 119 usernames received 3 attempts with 3 distinct passwords.

In 2008, Owens and Matthews [14] reported that attackers frequently use the username as the password (*e.g.,* user `david`, password `david`) in SSH guessing attacks. We recorded 50% of non-root and non-admin guesses attempting usernames as the password, confirming that attackers continue using this strategy.

7 Timing Analysis

Table 4 summarizes the rates of attempts received on each VM per day, and per hour. The last row shows combined statistics—minimum of the six minimums, median of medians, and maximum of maximums respectively.

Table 4. Summary of the rate of attempts.

VM	Rate of attempts					
	Per day			Per hour		
	Min	Median	Max	Min	Median	Max
OTT	349	17,805	273,120	0	754	85,770
LON	836	35,445	116,935	0	1,204	15,375
NY	354	15,036	192,351	0	53	69,770
SFO	180	15,993	119,981	0	47	60,026
AMS	281	9,655	91,901	0	101	7,742
SGP	516	3,755	38,290	0	55	10,451
*	180	15,515	273,120	0	55	85,770

On Jun 14, 2014, between 3pm and 4pm EST, the OTT VM received more than 85K attempts, averaging ∼24 per second. Collectively, those originated from 55 IP addresses in 13 /24s; however, about half of these attempts originated from a single /16. For the short term study (the other 5 VMs), we noticed that the two European-based VMs had relatively less skewed distribution of attempts over time. This can be inferred from the *Median Per Hour* (relatively large medians) and *Max Per Hour* columns (relatively small), suggesting the possible presence of IDS-like systems in common backbone European networks that enforce some form of rate-limiting.

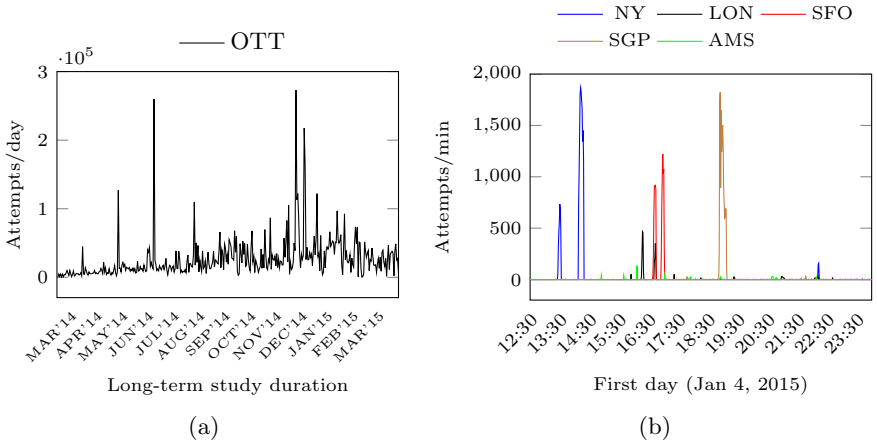


Fig. 6. (a) Daily attempts during the long-term study. (b) Attempts per minute during the first day of making the SSH servers public; all five VMs were started at the same time—the first point on the x-axis is 12:30pm UTC. Best viewed in colour.

At no single day did any of the six VMs receive zero attempts. The median attempts per day varied from 3.7K (SGP) to 17.8K (OTT). Statistical confidence from this data found a 90% chance the median number of (unthrottled) login attempts received per day by any SSH server is between 6.7K and 24.3K. The maximum number of attempts per day varied considerably across VMs. On Nov 16, 2014, the OTT VM experienced 273K attempts—the largest rate per day of all VMs. Similarly, the NY VM had the highest rate per day in the short term study, scoring 192K attempts on Jan 24, 2015.

Figure 6a shows the number of attempts received by the OTT VM in each day of the long term study (373 days). The spikes on May 6 and Jun 14 (126,917 and 260,046 attempts, respectively) occurred when an attacker first discovers the SSH server. A single source IP address was responsible for 94% of attempts on May 6, but that machine did not make further attempts on any VM after May 6. Likewise, a single IP was responsible for 98% of the attempts made on Nov 16; and `62.210.128.0`/18 was responsible for 79% of those on Nov 25.

Figure 6b shows the number of attempts in the first 12 hours (beginning at 12:30pm UTC) after starting the SSH servers of the five short-term study VMs. The NY VM was the first to receive guessing attempts only 51 minutes after it was started, versus 4 hours and 33 minutes for the SGP VM which was the latest. The number of sources attempting guesses on the first day varied as well, from 4 (SFO) to 15 (LON). Only a single source (located in Ukraine) made guesses on all five VMs on the first day within a two hour window (about four hours after starting the SSH service). This machine seemed not aggressive, making only one guess on each VM on that day. In fact, we found that this machine guesses a username-password combination on the five VMs then proceeds to the next combination; attempts on the VMs were in the same order (which is AMS,

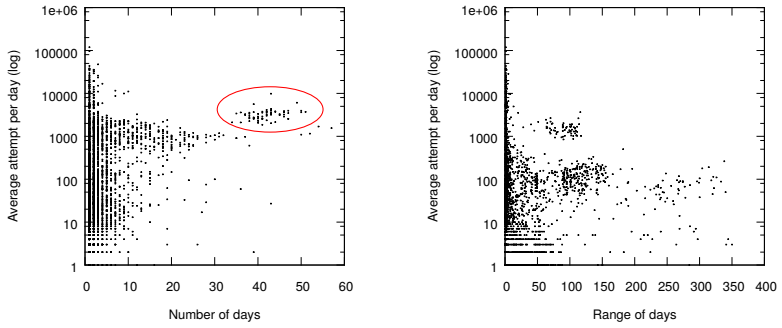


Fig. 7. A point (x, y) represents a machine (IP address) that made an average of y attempts per day and appeared in our logs (a) for x possibly non-consecutive days (b) for a range of x days, *i.e.,* difference between the first and the last day.

LON, SFO, NY, SGP), and the time between VMs was constant (*e.g.,* 7 minutes between AMS and LON, 25 between LON and SFO, *etc.*). It is thus likely that this machine performs reconnaissance concurrently to guessing passwords.

Next, we analyze the density of guessing attempts per source IP address over time. Figure 7 shows scatter plots where each point represents a source IP address. In Fig. 7a, the average number of attempts per day made by an IP address is plotted against the number of days the address appeared in our logs. Most IP addresses showed up for ten days or less. The highest point is a source machine that made 119,267 guessing attempts within a two hour period. Similar behaviour was noticed from five other hosts in the same /24, and there was a high degree of overlap between password guesses.

The point $(x, y) = (39, 2)$ is a source machine that collectively made 79 attempts in 39 non-consecutive days, averaging $\sim$2 attempts per day. All guesses were attempted on the SFO VM. The concentration of points between 35 and 50 (circled in Fig. 7a) belonged to the aggressive HK subnet `103.41.124.0`/24 (see Section 4), highlighting the persistence of this subnet.

Figure 7b similarly shows the average number of attempts but over a duration of consecutive days, from the first to the last day an IP address has appeared in the logs. The concentration of points is now closer to the bottom of the y-axis, compared to the previous case, since the average is taken over a longer period of time. Although the maximum number of (possibly non-consecutive) days an IP address have appeared in the logs did not exceed 60 (*i.e.,* from Fig. 7a), a range of up to 350 days was noticed. This could be due to an attacker not fully utilizing its computing resources, or perhaps machines being fixed then re-exploited. In conclusion, if IP-blacklisting was employed, system administrators should not haste into whitelisting addresses shortly after guessing attempts discontinue, since attacking machines may turn dormant for days before resuming activity.

8 Recommendations

Our data analysis allows us to propose concrete suggestions to administrators for reducing the probability of successful guesses of SSH credentials. We note that these recommendations do not preclude additional security mechanisms such as IDS, public key authentication, *etc.*

Network blocking. We observed that certain IP addresses, often working closely with other IP addresses in the same network block can exhibit aggressive behaviour when guessing credentials. Opportunistic detection and blocking of these sources could reduce the number of guesses by a significant fraction. For this purpose, tools such as DenyHosts and Fail2Ban (see Section 2), in addition to rate-limiting login attempts can prevent login attempts from polluting the system logs. We note, however, that any mechanism that blocks an IP address should periodically (*e.g.,* after three to six months) remove the block, as attacking systems tend to persist their attack for short time periods.

Username whitelisting/blacklisting. The overwhelming majority of login attempts target the `root` account. Thus, disabling remote logins for `root` could be an effective method of preventing successful guesses. A more proactive approach involves explicitly whitelisting usernames that are allowed to login via SSH, thus reducing the attack surface. Moreover, from our dataset, blacklisting the top 20% of observed usernames could prevent 80% of successful attacks, which aligns with suggestions by Florencio *et al.* [9]. This strategy may impose less cognitive load on the user (as compared to blacklisting passwords), since usernames must not be secret.

Proactive password checks. Proactive password checking is believed to improve the security of a system [5]. Florencio *et al.* further suggest that blacklists of as many as 10^6 passwords seems to be an effective operating point to prevent online guessing without causing substantial user inconvenience [10]. In addition to identifying frequently targeted accounts and passwords, we suggest proactive checking to disallow passwords equal to usernames.

9 Conclusion

We presented a methodology for collecting automated SSH login attempts, and analyzed the largest set of attack logs to date. By analyzing plaintext password guesses, we were able to identify broad dictionary sharing across all attackers, and note that many of the passwords used during SSH bruteforce campaigns are independent from those of common web-based password leaks. We also observed highly coordinated and highly distributed guessing attacks both within single subnets and distributed across multiple regions. Our analysis shows that attacks target all publicly accessible systems, but some systems may be of more interest to attackers due to their physical location or connectivity.

Acknowledgements. We thank Hala Assal, Elizabeth Stobert, Mohamed Aslan, Raphael Reischuk, and the anonymous referees for insightful comments which have improved this paper.

References

1. Internet Storm Center - SSH Scanning Activity. `https://isc.sans.org/ssh.html` Accessed September 13, 2015.
2. Nagios. `https://www.nagios.org` Accessed Sep. 13, 2015.
3. Country IP Blocks - Allocation of IP addresses by Country. `https://www.countryipblocks.net/allocation-of-ip-addresses-by-country.php`, Accessed September 13, 2015, 2015.
4. M. Alsaleh, Mohammad Mannan, and P. C. van Oorschot. Revisiting Defenses against Large-Scale Online Password Guessing Attacks. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 9(1):128–141, 2012.
5. F. Bergadano, B. Crispo, and G. Ruffo. High dictionary compression for proactive password checking. *ACM Transactions on Information and System Security (TISSEC)*, 1(1):3–25, 1998.
6. J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *IEEE Symposium on Security and Privacy*, 2012.
7. S. Chiasson and P. C. van Oorschot. Quantifying the security advantage of password expiration policies. *Designs, Codes and Cryptography*, pages 1–8, 2015.
8. Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide scanning and its security applications. In *USENIX Security*, Aug. 2013.
9. D. Florencio, C. Herley, and B. Coskun. Do Strong Web Passwords Accomplish Anything? In *USENIX HotSec*, pages 10:1–10:6, 2007.
10. D. Florencio, C. Herley, and P. C. van Oorschot. An Administrators Guide to Internet Password Research. In *USENIX LISA*, 2014.
11. R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras. SSH Compromise Detection using NetFlow/IPFIX. *ACM SIGCOMM CCR*, 44(5):20–26, 2014.
12. IPinfo. IP Address Details - ipinfo.io. `http://ipinfo.io` Accessed Sep. 13, 2015.
13. M. Javed and V. Paxson. Detecting Stealthy, Distributed SSH Brute-Forcing. In *ACM CCS*, 2013.
14. J. Owens and J. Matthews. A Study of Passwords and Methods Used in Brute-Force SSH Attacks. In *USENIX LEET*, 2008.
15. A. Satoh, Y. Nakamura, and T. Ikenaga. Identifying User Authentication Methods on Connections for SSH Dictionary Attack Detection. In *IEEE Annual Computer Software and Applications Conference Workshops (COMPSACW)*, 2013.
16. A. Sperotto, R. Sadre, P.-T. d. Boer, and A. Pras. Hidden Markov Model Modeling of SSH Brute-Force Attacks. In *Integrated Management of Systems, Services, Processes and People in IT*, pages 164–176. Springer Berlin Heidelberg, 2009.
17. J. L. Thames, R. Abler, and D. Keeling. A Distributed Active Response Architecture for Preventing SSH Dictionary Attacks. In *IEEE Southeastcon*, pages 84–89, 2008.
18. T. Ylonen. SSH - Secure Login Connections over the Internet. In *USENIX Security*, 1996.

Appendix A SSH servers on non-standard ports

A commonly suggested strategy to reduce the chances of guessing attacks succeeding is to change the default listening port (TCP 22) to a non-standard port. Using a port other than the default requires client-side changes, so easy to remember ports (such as 2222 or 2022) are sometimes suggested by administrators and users [14]. To investigate the validity of this suggestion, we created a network daemon that accepts incoming connections on all TCP ports (except port 22). When a new connection is received, the daemon looks for an initial SSH handshake and immediately closes the connection. We recorded the source IP, port number, and timestamp of all incoming SSH protocol connections. We ran this daemon on a separate VM over a 12 day period, and recorded 30,169 incoming connections to 522 distinct ports, out of which 77 were SSH connection. These incoming connections originated from 9 distinct source IP addresses. The full list of ports which received SSH connections are listed in Table 5. Notice that all incoming SSH connections have the digit 2 in the port number, many with two or more occurrences. Also of interest is that none of the 9 sources was seen making connections in the long-term or short-term study, hinting at the possibility that attackers dedicate some bots to non-standard scans/attacks. Despite the short duration of this study above (port numbers), the results show that attackers do not simply ignore non-standard ports. With availability of modern (and extremely fast) network scanning tools [8], attackers can quickly scan all open ports on sets of systems typically identified by an IPv4 address. Thus, moving an SSH daemon to a port other than 22 may not provide a comprehensive solution.

Table 5. List of non-standard ports on which incoming SSH connections were received.

Port	Count	Port	Count	Port	Count	Port	Count	Port	Count	Port	Count
2222	29	55022	3	22002	3	8822	1	2101	1	2233	1
1202	7	2312	3	21002	3	8022	1	2011	1	10022	1
2111	5	2266	3	7022	2	22203	1	2001	1		
22088	4	221	3	22201	2	2212	1	122	1		

Appendix B Top usernames and passwords (non-root)

Table 6 shows the top ten usernames and passwords, including the top ten passwords that did not appear in the RockYou dataset (RockYou's top ten are also included for reference). From the second column, we notice that the top ten passwords in our set are likely relevant to conventions among system administrators, due to the nature of SSH. The most common attempted password, `toor`, is the mirror of *root*. Nagios [2] is an open-source monitoring software.

For reference, in Table 7 we list the most frequent username-password combinations in our set. Note that most of these combinations follow the "username as password" strategy described in Section 6.

Table 6. Top ten passwords

SSH•			SSH• Not in RockYou			RockYou		
Password	Count	%	Password	Count	%	Password	Count	%
admin	20657	0.120	toor	7204	0.101	123456	290729	0.892
123456	17592	0.102	root@123	6771	0.095	12345	79076	0.243
password	14981	0.087	r00t	6593	0.092	123456789	76789	0.236
root	12122	0.070	data	6275	0.088	password	59462	0.182
1234	11515	0.067	root00	6269	0.088	iloveyou	49952	0.153
test	10091	0.059	p@ssw0rd1	5947	0.083	princess	33291	0.102
12345	9963	0.058	nagios	5908	0.083	1234567	21725	0.067
123	9371	0.054	admin@123	5806	0.081	rockyou	20901	0.064
abc123	9113	0.053	root123!@#	5581	0.078	12345678	20553	0.063
12345678	8747	0.051	shisp.com	5543	0.078	abc123	16648	0.051

•per experiments herein

Table 7. Top ten username-password combination, where the username is neither `root` nor `admin`. The length of this set is 294,694, of which 69,110 are unique.

Username	Password	Count	%
ubnt	ubnt	2098	0.712
guest	guest	1933	0.656
test	test	1932	0.656
oracle	oracle	1893	0.642
support	support	1516	0.514
info	info	1412	0.479
nagios	nagios	1203	0.408
pi	raspberry	1199	0.407
postgres	postgres	1103	0.374
ftp	ftp	1089	0.370

Appendix C Overlap of per-IP dictionaries

Figure 8 shows percentage overlap between per-IP dictionaries. For this graph, the per-IP dictionary contains the set of all passwords tried by that IP address during the full collection period. We sort the IP addresses numerically on the x and y axes, which allows us to identify large contiguous IP space exhibiting similar behaviour. For example, the subnet 103.41.124.0/24 contains many hosts with similarly sized dictionaries that have very little overlap (white vertical banding) with other dictionaries in the set.

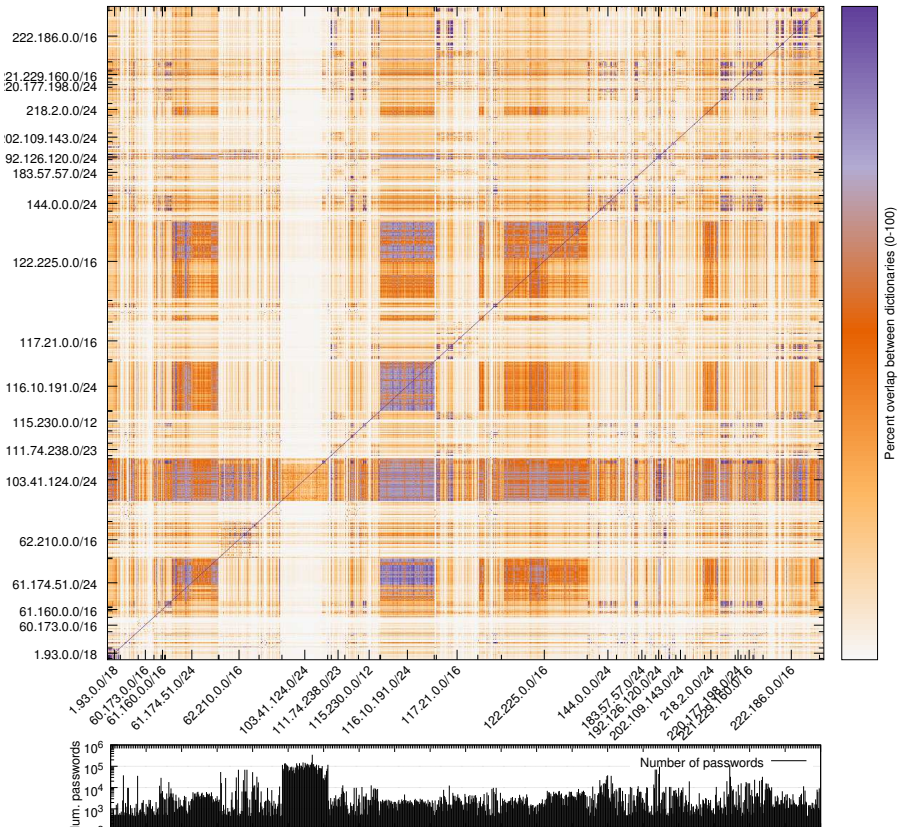


Fig. 8. Overlap between per-IP dictionaries. This figure plots overlap between the 1000 largest per-IP dictionaries. Dictionaries are sorted by IP address. The histogram below the heatmap shows the number of passwords in the per-IP dictionary for the IP immediately above.

Strengthening Public Key Authentication against Key Theft

Short Paper

Martin Kleppmann[1] and Conrad Irwin[2]

[1] University of Cambridge Computer Laboratory, mk428@cl.cam.ac.uk
[2] Superhuman Labs, San Francisco, conrad.irwin@gmail.com

Abstract. Authentication protocols based on an asymmetric keypair provide strong authentication as long as the private key remains secret, but may fail catastrophically if the private key is lost or stolen. Even when encrypted with a password, stolen key material is susceptible to offline brute-force attacks. In this paper we demonstrate a method for rate-limiting password guesses on stolen key material, without requiring special hardware or changes to servers. By slowing down offline attacks and enabling easy key revocation our algorithm reduces the risk of key compromise, even if a low-entropy password is used.

1 Introduction

Although passwords are the prevalent authentication mechanism on the internet today, there are some niches in which public key authentication systems have been successfully adopted. For example, SSH public key authentication [11] is widely used for remote login to servers, TLS client certificates [3] are used in some countries for access to public services [8], and FIDO U2F [10] provides 2-factor authentication for web applications.

In these protocols, a user account is associated with a public key, and a client authenticates itself to a server by computing a digital signature using the corresponding private key. The private key is stored on the client device (perhaps using a cryptographic hardware module), so the signature implements a machine-to-machine authentication protocol (a "something you have" factor). Since the device may be lost or stolen, an additional human-to-machine authentication step is employed to prevent an attacker using the key: for example, a password or biometric information can be used to unlock or decrypt the private key.

However, passwords and biometric identifiers are typically low in entropy, making them susceptible to offline attacks if a device is stolen. Our contribution in this paper is a scheme for storing an RSA private key in a way that makes it harder for an attacker to make use of stolen key material. We build upon the mRSA key-splitting scheme [2,6], which provides instantaneous key revocation, and extend it with a novel protocol for rate-limiting password guesses, which has the effect of slowing down offline attacks against stolen key material. In this work we limit our attention to RSA keys, but we hope to extend our approach to support other public-key cryptosystems such as ECC in future.

1.1 Threat Model

In our scenario, a client stores an RSA private key encrypted with a password. The client wishes to authenticate itself to a server as username r. We assume the server already knows which RSA public key belongs to which username. We require that all communication occurs over TLS, and that the client verifies the identity of the server using its existing PKI certificate or a pinned public key.

Our adversary is an active network attacker, but we assume that our use of TLS prevents the attacker from eavesdropping or tampering with messages. The attacker can steal encrypted private key material from a client device (e.g. by stealing the physical device or by compromising it remotely). We assume the attacker can trick the user into accessing fake services, but cannot trick the user into revealing the key encryption password to the attacker. We assume that the user is aware when a device has been lost or compromised, and that they are willing to take steps to revoke it.

In Sect. 2.2 we introduce a semi-trusted service called the *mediator*. We assume that data stored at the mediator is not accessible to the adversary who can steal private key material from clients. The mediator cannot authenticate on the client's behalf, and it need not be trusted as an authority.

2 Revocable Public Key Authentication

In this section we review an existing technique for instant revocation called *mediated RSA* (mRSA) [2,6]. We demonstrate it by example, using a simplified version of the FIDO protocols [10]. We build upon mRSA in Sect. 3 to explain our algorithm for rate-limiting password guesses.

2.1 Basic RSA Authentication

A client has a username r and an RSA private key (n, d), where n is the modulus and d the private exponent. The server knows the corresponding public key (n, e) for r, where e is the public exponent. To authenticate, the client first requests a fresh challenge c from the server. It then constructs an RSA signature s:

$$s = H(c \parallel cb \parallel u \parallel r)^d \mod n \ , \tag{1}$$

where u is the URL of the server, and cb is the TLS channel binding [1] or Origin-Bound Certificate [4] of the connection between server and client. The channel binding prevents MITM and replay attacks. H is shorthand for the EMSA-PSS-Encode operation (hashing and padding) defined in PKCS#1 [5].

The client then uses TLS to send the authentication request (s, c, u, r, n, e) to the server at URL u, which verifies that s is a valid PKCS#1 signature of $c \parallel cb \parallel u \parallel r$ using the public key (n, e), that c and u are valid for this server, that the channel binding matches, and that (n, e) is a public key for user r.

An adversary who steals the private exponent d can easily impersonate the client. A common solution is to encrypt d with a key derived from a password using a slow KDF such as scrypt [9]. However, password entropy is often low, so this is not sufficient to stop an attacker with significant computing resources.

2.2 The Mediator Service

To prevent theft of the private exponent d, we split it into key fragments using the mRSA method [2,6]. It is based on the identity:

$$s = m^d = m^{d_a + d_b} = m^{d_a} m^{d_b} \mod n \ . \tag{2}$$

The private exponent d is split into d_a, which is an integer drawn from the uniform random distribution $U(0, d)$, and $d_b = d - d_a$. Fragment d_a is encrypted with the user's password and stored on the client device a, while fragment d_b is stored on a remote server called the *mediator*. If the same user has multiple client devices, d can be split in a different way for each device, with the counterpart of each device's fragment stored on the mediator. It would be easy to split d into three or more summands, but we focus on the two-fragment case.

After the key has been split, a client device must work together with the mediator in order to construct a valid signature of the form in (1). When device a wants to generate a signature, it sends a message m to the mediator:

$$m = H(c \parallel cb \parallel u \parallel r) \ . \tag{3}$$

The request is sent over TLS and authenticated as described in Sect. 3.2. The mediator uses its key fragment d_b to calculate a response:

$$resp = m^{d_b} = H(c \parallel cb \parallel u \parallel r)^{d_b} \mod n \tag{4}$$

and returns *resp* to client device a. Now, a can calculate the RSA signature s:

$$s = H(c \parallel cb \parallel u \parallel r)^{d_a} \cdot resp = m^{d_a} m^{d_b} = m^d \mod n \ , \tag{5}$$

and thus authenticate with the server at URL u.

If a device's key fragment is stolen, it can instantly be revoked by deleting the counterpart fragment from the mediator, rendering the stolen fragment useless. This deletion request can be authenticated by another device owned by the same user, as discussed in Sect. 3.2. This implies that a user must enrol at least two physical devices with the mediator, so that the remaining device can revoke a lost device. A paper print-out of the key can serve as last resort in case all devices are lost or destroyed.

The mediator need only be partially trusted. It cannot authenticate as the user without the cooperation of one of the user's physical devices. The user only needs to trust the mediator to be always online, to keep key fragments safe from attackers who steal devices, and to correctly delete key fragments when the user requires key revocation. The user's privacy is protected by hashing the message $c \parallel cb \parallel u \parallel r$ before sending it to the mediator, so the mediator does not learn which services the user is logging in to, or which usernames they are using.

From the point of view of a server that uses public key authentication, the mediator does not even exist: a server simply verifies the RSA signature, and does not care how that signature was constructed. This is in contrast to federated login systems such as OpenID, where the relying party must trust the identity provider.

3 Rate Limiting Password Guesses

In the original proposal of mRSA [2], requests to the mediator are not authenticated. In this section we show that by adding authentication, we can strengthen mRSA to prevent offline attacks against stolen private key material.

Consider an attacker who has stolen a client device on which key fragment d_a is stored, encrypted with password *pass*. The attacker reads the encrypted fragment $E(d_a, pass)$ from the device, and mounts an offline attack by repeatedly trying a password guess $pass'$ (based on a dictionary or brute force) and computing $D(E(d_a, pass), pass')$ until the correct d_a is found.

However, an offline attack on the password requires the attacker to be able to determine whether a decryption attempt has indeed yielded the correct d_a. The following protocol ensures that an attacker must make a request to the mediator for every decryption attempt in order to determine whether it is correct. This allows the mediator to limit the rate of decryption attempts, giving the user more time to revoke the stolen device, even if the password is fairly weak.

3.1 Key Fragment Encryption

Let k be the RSA key length. A key fragment d_a can be encoded as a k-bit string, using zero padding for the most significant bits, since $d_a < d < n < 2^k$. This k-bit string can then be encrypted into a k-bit ciphertext *efrag*, using a stream cipher and a key derived from a password. For example, we can use the scrypt KDF [9] and AES-128 in CTR mode [7] as stream cipher:

$$efrag = \mathrm{AESCTR}(ctr, \mathrm{scrypt}(pass))_{\{0\ldots k-1\}} \oplus d_a \ , \tag{6}$$

where *ctr* is a random nonce that is stored in plaintext and incremented by AESCTR for each block of key stream. An attacker who has stolen *efrag* and *ctr* may guess a password $pass'$, and compute a guess d'_a of the key fragment:

$$d'_a = \mathrm{AESCTR}(ctr, \mathrm{scrypt}(pass'))_{\{0\ldots k-1\}} \oplus efrag \ . \tag{7}$$

If the password guess $pass'$ is incorrect, d'_a is a uniformly distributed pseudorandom number between 0 and 2^k. We deliberately choose *not* to use authenticated encryption, because the MAC would tell the attacker whether the password guess was correct, making an offline attack easy.

Note that d_a is drawn from a uniform distribution $U(0, d)$, whereas d'_a is drawn from $U(0, 2^k)$. Since $d < 2^k$, the distributions are different, which leaks some information: smaller values of d'_a are more likely to be correct than larger ones. Apart from this bias, there are no particular features that distinguish the correct d_a from a random bit string.

To quantify this assertion, we generated 50,000 RSA keys ($k = 2048$ bits) using OpenSSL, and drew a uniformly distributed random d_a with $0 \leq d_a < d$ for each private exponent d. Table 1 shows the bias in the most significant bits of d_a when encoded in k bits. The key fragments had an entropy of 2047.05 bits, implying that 0.95 bits of information are leaked by the bias. This can be used by an attacker to prioritize guesses that are more likely to be correct, but an attacker cannot rule out password guesses from examining d'_a alone.

Table 1. Probability that bit i of d_a is 1, when encoded in $k = 2048$ bits

i	2047	2046	2045	2044	2043	2042	2041	2040	2039
Probability	0.070	0.240	0.340	0.406	0.446	0.469	0.482	0.493	0.499

3.2 Authenticating Requests to the Mediator

Furthermore, to prevent offline attacks on encrypted key fragments, requests to the mediator must be authenticated. To see why this is the case, consider an unauthenticated mediator that accepts any message m and returns $m^{d_b} \mod n$ as in (4). An attacker could use this response to test whether a password guess $pass'$ is correct, by using (7) to compute d'_a and checking whether $m^{d'_a} m^{d_b} \mod n$ is a valid RSA signature.

To prevent this, a client must prove to the mediator that it knows the correct password *pass* without revealing the password or the decrypted key fragment d'_a. This is accomplished by the following protocol:

1. When the client requests the mediator to compute a partial signature on a message m, it must also include a partial signature s_m using d'_a:

$$m = H(c \parallel cb \parallel u \parallel r) \tag{8}$$

$$s_m = H(m \parallel cb_m)^{d'_a} \mod n \tag{9}$$

   where cb_m is a channel binding [1] of the TLS connection between the client and the mediator. Note that cb is between client and server, whereas cb_m is between client and mediator.
2. The mediator uses its own channel binding cb'_m of the connection from the client to compute:

$$s_m \cdot H(m \parallel cb'_m)^{d_b} = H(m \parallel cb_m)^{d'_a} \cdot H(m \parallel cb'_m)^{d_b} \mod n \tag{10}$$

   and checks whether the result is a valid signature of $m \parallel cb'_m$ for the user's public key (n, e). This check succeeds if $d'_a = d_a$ (i.e. the user's password was correct), and if $cb'_m = cb_m$ (preventing MITM and replay attacks).
3. If the signature is valid, the mediator computes

$$resp = m^{d_b} = H(c \parallel cb \parallel u \parallel r)^{d_b} \mod n \tag{11}$$

   as before, and returns it to the client. If the signature is not valid, the mediator returns "bad signature".

When a password-guessing attacker receives a "bad signature" response, it learns that the password guess $pass'$ was incorrect, but it does not gain any additional information that would help it determine whether any other password guess $pass''$ is correct or not. Thus, the attacker must make a request to the mediator for every guess. If the mediator receives too many requests for a signature with a particular fragment within a short time, it returns an error.

The same mechanism can be used to authenticate key revocation: the mediator only processes a revocation request for a device if it is authenticated by another device of the same user. This avoids relying on a central authority.

4 Conclusion

The security of key-based authentication is only as good as the protection of the private key material. In this paper we extend mRSA, an existing method for revocation of private keys, by authenticating requests to the mediator.

Our algorithm ensures that an attacker who has stolen a password-encrypted key, and wants to guess the password, must make a request to a mediator for every attempt. This gives the mediator the opportunity to limit the rate at which passwords can be tested, giving the user more time to revoke the lost device's key. No special hardware is required, and the server just performs standard RSA signature verification, making our approach compatible with existing systems.

Acknowledgements

We thank Alastair R. Beresford and the reviewers for their helpful feedback.

References

1. Altman, J., Williams, N., Zhu, L.: Channel bindings for TLS. IETF RFC 5929 (Jul 2010)
2. Boneh, D., Ding, X., Tsudik, G., Wong, C.M.: A method for fast revocation of public key certificates and security capabilities. Proceedings of the 10th USENIX Security Symposium pp. 297–308 (Aug 2001)
3. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) protocol version 1.2. Network Working Group RFC 5246 (Aug 2008)
4. Dietz, M., Czeskis, A., Balfanz, D., Wallach, D.S.: Origin-Bound Certificates: A fresh approach to strong client authentication for the web. In: 21st USENIX Security Symposium. pp. 317–332 (Aug 2012)
5. Jonsson, J., Kaliski, B.: Public-Key Cryptography Standards (PKCS) #1: RSA cryptography specifications version 2.1. Network Working Group RFC 3447 (Feb 2003)
6. Kutyłowski, M., Kubiak, P., Tabor, M., Wachnik, D.: Mediated RSA cryptography specification for additive private key splitting (mRSAA). IETF Internet Draft (Nov 2012)
7. Lipmaa, H., Rogaway, P., Wagner, D.: Comments to NIST concerning AES modes of operations: CTR-mode encryption (Sep 2000)
8. Parsovs, A.: Practical issues with TLS client certificate authentication. In: Network and Distributed System Security Symposium (NDSS) (Feb 2014)
9. Percival, C.: Stronger key derivation via sequential memory-hard functions. BSDCan '09 (May 2009)
10. Srinivas, S., Balfanz, D., Tiffany, E., Czeskis, A.: Universal 2nd factor (U2F) overview. FIDO Alliance Proposed Standard (May 2015)
11. Ylonen, T., Lonvick, C.: The Secure Shell (SSH) authentication protocol. Network Working Group RFC 4252 (Jan 2006)

On Password-Authenticated Key Exchange Security Modeling

Jean Lancrenon

Université du Luxembourg,
Interdisciplinary Centre for Security, Reliability and Trust,
6 rue Richard Coudenhove-Kalergi, L-1359 Luxembourg City,
LUXEMBOURG
jean.lancrenon@uni.lu

Abstract. Deciding which security model is the right one for Authenticated Key Exchange (AKE) is well-known to be a difficult problem. In this paper, we examine definitions of security for Password-AKE (PAKE) in the style proposed by Bellare et al. [5] at *Eurocrypt 2000*. Indeed, there does not seem to be any consensus, even when narrowing the study down to this particular authentication method and model style, on how to precisely define fundamental notions such as accepting, terminating, and partnering. The aim of this paper is to begin addressing this problem. We first show how definitions vary from paper to paper. We then propose and thoroughly motivate a definition of our own, and use the opportunity to correct a minor flaw in a more recent and more PAKE-appropriate model proposed by Abdalla et al. [3] at *Public Key Cryptography 2005*. Finally, we argue that the uniqueness of partners holding with overwhelming probability ought to be an explicitly required and proven property for AKE in general, but even more so in the password case, where the optimal security bound one aims to achieve is no longer a negligible value. To drive this last point, we exhibit a protocol that is provably secure following the Abdalla et al. definition, and at the same time fails to satisfy this property.

1 Introduction

Password-authenticated key exchange. *Key Exchange* (KE) is preoccupied with establishing a secure ephemeral *session key* between two remote parties over an insecure network. The well-known Diffie-Hellman protocol [16] solves this problem over communication lines that are perhaps eavesdropped on, but remain otherwise undisturbed. *Authenticated* KE (AKE) aims to realize KE in such a way that the communicating parties gain some guarantee that they have established a key with the right partner even in the face of malicious - i.e. more than just eavesdropping - attackers. This is only possible if all honest parties in play have some sort of trusted setup in place prior to the exchange, taking the form e.g. of a Public Key Infrastructure (PKI) correctly managing everybody's public keying information, or pairwise-shared long-term symmetric keys, etc. *Password* AKE (PAKE) is AKE in which the long-term keys are simple passwords.

Using ordinary passwords - by which we mean low-entropy, human-memorable pieces of data - as long-term authentication material cannot simply be securely viewed as a particular case of the shared-symmetric-key scenario. Indeed, the low entropy of passwords makes them vulnerable to dictionary attacks, both off-line and on-line. Hence, special care must be taken in designing a PAKE protocol. Note that it is always trivially possible to simply run the protocol with an honest participant using a guessed password as input, and observing whether the exchange fails or succeeds. Such an on-line attack is inherent to the service, and cannot be avoided. Intuitively, we would like to

limit a network adversary's effective attacks to this trivial strategy. This implies namely that untampered protocol network traffic must protect, at least computationally, every single bit of information on the underlying passwords. In a nutshell, on-line attacks are limited to successive login attempts, and off-line attacks are completely prohibited.

Security models. Complexity theoretic PAKE security models capturing these requirements made their first appearance at Eurocrypt 2000, in the form of the Bellare et al. [5] and Boyko et al. [9] models. This paper is concerned with proofs of security using the former, which we shall call BPR-style proofs. We also consider proofs using Abdalla et al.'s [3] model, which is a slight modification of the [5] model. BPR-style proofs are by far the most common in the literature, but unfortunately they are not very easy to understand. One reason for this is that often, small technicalities vary in the models' definitions from paper to paper, and it is not always clear why. This is especially the case when trying to define *when it is that two parties have had a correct protocol conversation*, a notion known as *partnering*.

Our aim is to draw attention to this issue, in order to at least start clarifying it. This is important to be able to more accurately interpret security claims in papers.

Our contributions. In Section 2, we focus on the current state of BPR-style modeling in the literature. We first show how definitions vary from paper to paper in terms of instances accepting, terminating, and partnering, citing precise examples. We then show how the subsequent security model of Abdalla et al. [3] is slightly bugged. These considerations lead into Section 3, in which we redefine the model(s) to fit all usual protocol scenarios and also fix the bug in [3]. The change has to do with the uniqueness requirement of the partner in the definition. As a follow-up, we examine this uniqueness property in detail in Section 4. We namely present evidence that it ought to be an explicitly stated and proven property of *any* AKE protocol, and even more so in the PAKE case, where the optimal security bounds are no longer negligible values (in order to account for the adversary's natural ability to simply guess a password and try it out). To this end, we exhibit a PAKE which is provably secure according to the Abdalla et al. definition, but fails to satisfy that partners be unique with overwhelming probability.

Related work. PAKE was first considered by Bellovin and Merritt [7] and Jablon [21], with informal security arguments. Lucks [30] and Halevi et al. [20] were the first to produce formal models involving passwords. They were followed by Bellare et al. [5] and Boyko et al. [9], building on the AKE models introduced in [6] and [34], respectively. Since, many protocols have been proposed and studied: Katz et al. [23] showed that PAKE can be practically realized without random oracles but with a *Common Reference String* (CRS), Goldreich et al. [18] showed that PAKE can be realized solely under general complexity assumptions, and Canetti et al. [13] introduced *universally composable* (see [12]) PAKE, to name a few. A more complete bibliography on PAKE can be found in [33].

Some research on comparing AKE security models in general has been conducted. Notably, Choo et al. [14] and Cremers [15] examine how differing definitions affect security in indistinguishability-based models, but none of these works focus on password protocols. To our knowledge, nothing of the sort for PAKE has been considered yet.

Organization of the paper. The rest of the paper is structured as follows. Section 2 overviews existing PAKE model definitions, and points to some differences and inconsistencies. Next, Section 3 revisits all of these definitions, trying to place them under one roof, and introduces the matter of partner uniqueness. Then, Section 4 shows why making uniqueness of partners a property is important, especially in the context of PAKEs. Finally, Section 5 concludes the paper.

2 Different BPR-style models

In what follows, we denote $\perp$ the special error symbol and ε the empty string. We use $\mathcal{U}$ to designate both the entity $\mathcal{U}$, and the bitstring identifying $\mathcal{U}$.

2.1 The models' main foundations

Here we describe the main aspects commonly shared by all BPR-style models. Some of the notions are *at first* left intentionally vague. We first detail the parties in play, then we list the adversary's abilities. The security definitions capturing confidentiality of SKs and authentication come at the end.

Principals and instances. An interactive game, indexed by the security parameter $\lambda \in \mathbb{N}$, is played between a challenger $\mathcal{CH}$ and an adversary $\mathcal{A}$. All of the algorithms considered are Probabilistic, Polynomial-Time (PPT) in λ.

At the beginning of the game, there is a fixed set of *principals* (or *users*), which is partitioned into non-empty sets of *clients* $\mathcal{C}$ and *servers* $\mathcal{S}$. Each client $\mathcal{C}$ is assigned a *password* $pw_{\mathcal{C}}$ drawn uniformly at random from some finite - possibly small - set PW of bitstrings. Each server $\mathcal{S}$ holds the full set of all clients' passwords $\{pw_{\mathcal{C}}\}_{\mathcal{C}}$.

The adversary $\mathcal{A}$ has oracle access - via the queries described below - to any number of *instances* $\mathcal{U}^i$ ($i \in \mathbb{N}$) of any principal $\mathcal{U}$. Intuitively, an instance of $\mathcal{U}^i$ represents an attempt $\mathcal{U}$ makes at running the AKE protocol over the network, which is fully controlled by $\mathcal{A}$. An instance's objective is to compute a *Session Key* (or SK) it believes it shares with an instance $\mathcal{V}^j$ of some other principal $\mathcal{V}$. This should happen only if $\mathcal{U}^i$ thinks it has *"had a correct exchange with"* $\mathcal{V}^j$.

At any point in time, an instance $\mathcal{U}^i$ may declare itself *"ready to use a SK"*. By this time, the instance should have computed **1)** a *Partner Identity* (or PID) $pid^i_{\mathcal{U}}$, a **2)** *Session Identity* (SID) $sid^i_{\mathcal{U}}$, and of course **3)** the SK $sk^i_{\mathcal{U}}$ itself. The PID is a bitstring indicating the identity of the instance with which $\mathcal{U}^i$ believes it has communicated with. The SID is a bitstring serving as an identifier for both the key exchange run that just occurred, and the session in which the computed SK will subsequently serve. Often in practice the SID is set to being the ordered concatenation of all exchanged protocol messages, except possibly the last message. At any point in time an instance may also *"refuse to participate any longer in the protocol"*, and halt altogether. This can happen if a message e.g has an incorrect format, or if authentication fails. Once an instance has finished - either declaring it has a SK, or having just stopped - it can no longer be reused.

We purposefully did not make the notions of "having a correct exchange with an instance", "declaring oneself ready to use a SK", and "refusing to participate any longer in the protocol" precise, because these are where consensus does not seem to hold. They will be made more precise **1)** in the different examples we give further below and **2)** when we give and motivate our own definition in Section 3. These notions are also correlated to when and how the PID, SID, and SK are computed.

We now describe the oracle queries the adversary has access to. These change according to whether we are in the *Find-then-Guess* (or FtG) model [5] or the *Real-or-Random* (or RoR) model [3].[1] The security definitions themselves will come after.

Find-then-Guess. This is the original model introduced in [5]. It can be viewed as a "password spin-off" of the classic Bellare/Rogaway model [6] for AKE security.

[1] To simplify our exposition, in this preliminary study we make no attempt at dealing with the *corruption* query - used to model the important property of *forward secrecy* - in this paper.

- $\mathsf{send}(\mathcal{U}, i, m)$: $\mathcal{A}$ has message m delivered to $\mathcal{U}^i$. $\mathcal{U}^i$ processes the message according to protocol specification. To instruct an instance $\mathcal{U}$ to send the first protocol message to entity $\mathcal{V}$, $\mathcal{A}$ makes the query with $m = \mathcal{V}$. This query is used to model arbitrary message delivery to an instance. In particular, it serves to count impersonation attacks.
- $\mathsf{execute}(\mathcal{U}, \mathcal{V}, i, j)$: The protocol is executed faithfully and completely between $\mathcal{U}^i$ and $\mathcal{V}^j$ and the resulting transcript is given to $\mathcal{A}$. $\mathcal{A}$ thus gets to see as many honest protocol runs as it wishes.
- $\mathsf{reveal}(\mathcal{U}, i)$: If $\mathcal{U}^i$ is not ready to use a SK, the query returns $\perp$. Otherwise, it returns $sk^i_{\mathcal{U}}$ to $\mathcal{A}$. This models leakage of session key information through use in the ensuing session (the quality of the algorithms of which we know *nothing* about).
- $\mathsf{test}(\mathcal{U}, i)$: If $\mathcal{U}^i$ is not ready to use a SK, this returns $\perp$. Otherwise, $\mathcal{CH}$ flips a coin b outside of $\mathcal{A}$'s view. If $b = 0$, a random string R is drawn from the session key space and $tk^i_{\mathcal{U}} \leftarrow R$. Otherwise, $tk^i_{\mathcal{U}} \leftarrow sk^i_{\mathcal{U}}$. The *test key* (or TK) $tk^i_{\mathcal{U}}$ is returned to $\mathcal{A}$. The test query may only be used once in the game. Its purpose is to measure the adversary's advantage in breaking session key security.

Eventually, $\mathcal{A}$ halts the overall game, at which point it outputs a bit b'. If the game was halted without $\mathcal{A}$ making any test query, then $\mathcal{CH}$ privately flips a coin b.

Freshness: To get a meaningful definition of SK security (see below), a restriction must be put in place on the test query. Namely, $\mathcal{A}$ cannot be considered victorious if it already trivially knows the SK that it tested. Therefore, the notion of *freshness* must be introduced: An instance $\mathcal{U}^i$ is *fresh* if neither it, nor any instance with which it has had a correct exchange, has been the subject of a reveal query. *Thus, we must slightly modify the* test *and* reveal *queries presented above:* we add that test returns $\perp$ if $\mathcal{U}^i$ is not fresh and that reveal returns $\perp$ if $\mathcal{U}^i$, or an instance with which it has had a correct exchange, has been tested.

Real-or-Random. This is the model introduced in [3]. In a nutshell, it differs from FtG in that it makes all of the keys revealed to the adversary either the truly computed keys or completely random strings. In [3] it is proven that for PAKEs, this makes a real security difference. Thus, it is recommended to use RoR rather than FtG when employing BPR-style models. (See [3] for details.)

The send and execute queries are identical to those in the FtG model. However, the reveal query is no longer available. Instead, it is replaced by as many test queries as $\mathcal{A}$ wants. How they are answered depends on the value of a bit b flipped by $\mathcal{CH}$ outside of $\mathcal{A}$'s view *at the beginning of the game.*

- $\mathsf{test}(\mathcal{U}, i)$: If $\mathcal{U}^i$ is not ready to use a session key, $\perp$ is returned. Otherwise, suppose first that $b = 0$. If $\mathcal{U}^i$ has not had a correct exchange with any instance, or no instance that it has had a correct exchange with was subjected to a test query, $\mathcal{CH}$ selects a random R from the session key space and sets $tk^i_{\mathcal{U}} \leftarrow R$. If $\mathcal{U}^i$ has had a correct exchange with an instance $\mathcal{V}^j$ that has been tested, $\mathcal{CH}$ sets $tk^i_{\mathcal{U}} \leftarrow tk^j_{\mathcal{V}}$. Suppose now that $b = 1$. In this case, $\mathcal{CH}$ sets $tk^i_{\mathcal{U}} \leftarrow sk^i_{\mathcal{U}}$. Then, $\mathcal{A}$ receives $tk^i_{\mathcal{U}}$.

The slight complication arising in case $b = 0$ is that even if the SKs assigned are random, they must at least remain consistent across instances that should hold the same keys. As in FtG, at any point in time $\mathcal{A}$ may halt the game and output a bit b'.

Technically defining security. In both FtG and RoR, one usually considers three security properties: SK security, Client-to-Server (C2S) authentication, and Server-to-Client (S2C) authentication.

SK security: Let S be the event that "$b' = b$ at the end of the game". Event S measures the *semantic security of the SK*, i.e. the adversary's ability to tell this key

apart from a random string. Since $\mathcal{A}$ can simply flip a coin to get the answer right with probability 1/2, $\mathcal{A}$'s natural advantage is defined to be $\mathsf{Adv}^s(\mathcal{A}) := 2\mathsf{Pr}[\mathsf{S}] - 1$. A PAKE protocol is said to have semantically secure SKs if there exists a non-zero constant $C \in \mathbb{N}$ such that for any PPT $\mathcal{A}$, there exists a negligible (in λ) function negl with the property that $\mathsf{Adv}^s(\mathcal{A}) \leq \frac{Cn_{se}}{|\mathsf{PW}|} + \mathsf{negl}(\lambda)$ where n_{se} is an upper bound on the number of send queries the adversary makes.

Authentication: Let $\mathsf{C2S}$ be the event that "there exists some server instance $\mathcal{S}^j$ that is ready to use a SK, but has not had a correct exchange with a client instance." This event measures the adversary's ability to cause client-to-server authentication to fail, i.e. a server thinks it is talking to a correct client when it is not. Here we simply set $\mathsf{Adv}^{c2s}(\mathcal{A}) := \mathsf{Pr}[\mathsf{C2S}]$, and we say that a PAKE achieves client-to-server authentication if there exists a non-zero C such that for any PPT $\mathcal{A}$, there exists a negligible function negl with the property that $\mathsf{Adv}^{c2s}(\mathcal{A}) \leq \frac{Cn_{se}}{|\mathsf{PW}|} + \mathsf{negl}(\lambda)$. Server-to-client authentication is defined similarly.

Intuitively, the constant C represents the number of passwords that can be ruled out per login attempt. Obviously, $C = 1$ is the optimal bound.

We will return to these definitions with precise terminology in Section 3.

2.2 Differences in accepting, terminating, and partnering

In this section, we illustrate how the notions of "having had a correct exchange", "declaring oneself ready to use a SK", and "refusing to further participate in the protocol", among other things, are formalized in various examples of the literature. These formalizations often vary from paper to paper, usually without much justification.

We begin with some technical terminology which is *mostly* common to all papers, but the interpretation and use of which are what always seem to vary:

- Two instances which "have had a correct exchange" are said to be *partnered*;
- An instance that is "ready to use a session key" is said to have *accepted* or *terminated*;
- The "refusal to further participate in the protocol", oddly, does not seem to have a technical term commonly attached to it.

In the examples that follow, we recapitulate precisely how the afore-mentioned points are dealt with and interpreted. We picked these three examples as they can be considered "landmark" papers: The first [5] introduced BPR-style reasoning to PAKE analysis, the second [3] brought in the RoR model, and the (conference version [23]) of the third [24] showed that PAKE is realizable without random oracles.[2] After each example, we also compile a list of additional papers that emulate (or claim to emulate) the example in question. Recall that ε designates the empty string.

In the original model from [5]. This model - the first of its kind - is FtG. ***Accepting:*** At any point in time, an instance $\mathcal{U}^i$ may *accept*. This means that it holds a non-ε SK and has computed a non-ε PID and non-ε SID. ***Terminating:*** At any point in time after having accepted, an instance may *terminate*. This means that it will no longer send, nor expect to receive, any more messages. ***Partnering:*** Two instances $\mathcal{U}^i$ and $\mathcal{V}^j$ are partnered if **1)** one is a client and one is a server, **2)** both instances have accepted, **3)** $pid^i_{\mathcal{U}} = \mathcal{V}$ and $pid^j_{\mathcal{V}} = \mathcal{U}$, **4)** $sid := sid^i_{\mathcal{U}} = sid^j_{\mathcal{V}}$, **5)** $sk^i_{\mathcal{U}} = sk^j_{\mathcal{V}}$, and **6)** no other instance accepts with a SID sid.

[2] Of course, these are not the only beacons in the field; they are just the most relevant to our work.

Apparent intended interpretation: Accepting in this model appears to mean being ready to use the session key, since it is under the condition of having accepted that reveal and test queries can be made. However, accepting is different from terminating in that an instance may wish to accept at one point in time, and yet terminate later. This is to model key confirmation: an instance accepts first - it believes it holds a good session key - but waits for its purported partner to send it a confirmation code to terminate.

Some observations: That the session key can be used before a confirmation code is received is puzzling to us. We believe it may be more logical to have the SK be formally accepted as such at termination time, using this terminology.

There does not appear to be any special term for when an instance refuses to continue in the protocol.

Also, notice that formally, for an instance to be partnered, its partner must be unique. More on this uniqueness is in Paragraph 2.3 and Section 4.

No notion of protocol correctness is defined.

Papers following this approach include [31, 10, 11, 29, 32, 8].

The RoR model paper [3]. This paper's primary topic of investigation is PAKE in the three-party setting[3] (or 3-PAKE, as opposed to the two-party setting, or 2-PAKE), but it contains contributions (namely, the RoR method) relevant to both 3-PAKE and 2-PAKE. (In this paper, we consider only 2-PAKE, which we shall also continue calling just "PAKE".) We focus on the 2-PAKE definitions, referring to the 3-PAKE ones when appropriate. ***Accepting:*** For 2-PAKE, accepting is not formally defined, so it is unclear whether a session key exists at this point or not. For 3-PAKE, accepting only happens "after receiving the last expected protocol message", so this actually corresponds more to termination in [5]. It is unclear whether the 2-PAKE definition is assumed to be the same. ***Terminating:*** This is not formally defined. However, it is stated that "in practice, the SID can be taken to be the partial transcript of the conversation between the client and the server instances before the acceptance." This implies that accepting comes earlier than something else, possibly termination. ***Partnering:*** For 2-PAKE, two instances $\mathcal{U}^i$ and $\mathcal{V}^j$ are partnered if **1)** both instances have accepted, **2)** $sid := sid^i_{\mathcal{U}} = sid^j_{\mathcal{V}}$, **3)** "the partner identifier for $\mathcal{U}^i$ is $\mathcal{V}^j$ and vice-versa", and **4)** "no instance other than $\mathcal{U}^i$ and $\mathcal{V}^j$ accepts with a partner identifier equal to $\mathcal{U}^i$ or $\mathcal{V}^j$".

Observations: Accepting and terminating are unclear. Accepting here may be the same as terminating in [5].

Nothing is in place to indicate if an instance refuses to continue in the protocol.

In terms of partnering, for 2-PAKE there is no need for one instance to be a client and the other a server, unlike in [5]. Points **3)** and **4)** are unclear, because the PID *of an instance* is never defined anywhere. However, these points should be probably be understood respectively as "$pid^i_{\mathcal{U}} = \mathcal{V}$ and $pid^j_{\mathcal{V}} = \mathcal{U}$" and "no other instance accepts with a SID sid", as in [5]. Also, in contrast to [5], there is no condition on the SK anymore.

The uniqueness condition is still required to formally satisfy partnering.

There is no mention of correctness.

Papers following this approach include [27, 2, 1].

The journal version [24] of [23]. Paper [23] presented the first practical PAKE secure under standard assumptions (but with a CRS). We discuss the journal version [24] here, which is FtG. ***Accepting:*** For a given instance $\mathcal{U}^i$, once $\mathcal{U}^i$ accepts, $sid^i_{\mathcal{U}}$, $pid^i_{\mathcal{U}}$,

[3] A server aids two clients that wish to exchange a key between themselves; each client shares a private password with the server.

and $sk^i_{\mathcal{U}}$ are no longer ε. Also, acceptance implies termination. ***Terminating:*** Terminating means the instance will no longer send nor receive messages. ***Partnering:*** Two instances $\mathcal{U}^i$ and $\mathcal{V}^j$ are partnered if **1)** one is a client and the other a server, **2)** $sid^i_{\mathcal{U}} = sid^j_{\mathcal{V}} \neq \varepsilon$, and **3)** $pid^i_{\mathcal{U}} = \mathcal{V}$ and $pid^j_{\mathcal{V}} = \mathcal{U}$.

Observations: Accepting and terminating are distinct, but unlike in [5], accepting implies termination. This suggests that termination without acceptance is used, in this paper, to designate when an instance refuses to continue.

Partnering differs here from [5] in that acceptance is no longer required, there is no condition on the SK, and there is no requirement of partner uniqueness.

There is a correctness notion: If $\mathcal{U}^i$ and $\mathcal{V}^j$ are partnered, then they must have both accepted and have equal session keys.

The authors state that their definition only covers implicitly authenticated protocols, and that partnering would have to be redefined in order to account for explicit authentication.

Papers following this approach include [17, 22, 19, 25, 26][4].

Having such a variety of definitions to pin down formally fundamental notions is problematic. First, while we have not attempted to show so in this work, it is more than likely that protocols deemed secure according to one definition become trivially insecure according to another definition. Situations like these have been documented in the past in non-password-based cases, see [14, 15]. Secondly, it is confusing for anybody trying to decide whether a protocol's formal security can be trusted or not.

In what follows, we focus on the "uniqueness of partners" aspect in these definitions. We begin by taking another look at the RoR model as defined in [3].

2.3 A bug in the RoR model

In [3], it seems the RoR definition is slightly ill-defined.

Consider the following scenario in the RoR security game. Suppose that the bit flipped at the beginning of the game ends up being 0, making $\mathcal{CH}$ output all-random keys. Suppose also that at some point in time, the adversary $\mathcal{A}$ gets a pair of instances $\mathcal{U}^i$ and $\mathcal{V}^j$ to accept and be partners according to the definition in Section 2.2. In particular, at this point in the game, $\mathcal{U}^i$ and $\mathcal{V}^j$ are the only instances to have accepted with a SID equal to $sid := sid^i_{\mathcal{U}}$. Next, $\mathcal{A}$ performs a test query on each instance. The result is that $\mathcal{A}$ receives the same random string R from both instances. Now, suppose that somehow $\mathcal{A}$ gets a third instance $\mathcal{V}^k$ with $pid^k_{\mathcal{V}} = \mathcal{U}$ to accept with SID equal to sid. This removes the uniqueness part of the partnering definition, thus $\mathcal{U}^i$ and $\mathcal{V}^j$ are not partnered anymore. Hence, the result of the test query is formally inconsistent with the definition, because $\mathcal{U}^i$ and $\mathcal{V}^j$ should have independent random keys, now that they are not partnered. As for $\mathcal{V}^k$, it is unclear really which key should be assigned in the event a test occurs. Should it be random in order to remain consistent with the definition, or should it be set to R in order to be consistent with $\mathcal{A}$'s view?

This definitional problem can be solved by changing the requirements of partnering with respect to uniqueness. Note however that ultimately, it is reasonable to expect that double partnering should occur with negligible probability only anyway. However, we cannot make any security statements to enforce this at this point in the model. It must be a proven property, so should not be integrated into the definition.

[4] [19] contains a notion of *semi*-partnering in order to have a definition for instances that have had a correct exchange even if the last message has not been delivered. We adopt this further in this work.

3 A well-motivated definition

In this section, we build on - and complete - all of the definitions above by supplying a model of our own. We then show how it could fit several protocol formats, covering all typical scenarios of implicit and explicit, unilateral and mutual authentication.

In order to obtain a full description of the formal model, one could simply replace the phrases "ready to use a session key", "refuses to pursue the protocol", and "has had a correct exchange with $\mathcal{U}^i$" with "has accepted", "has aborted", "is partnered to $\mathcal{U}^i$" respectively.

In order to stay with the habits of the literature, we continue to use the random variables for SID, PID, and SK, and the terms "accepting", "terminating", and "partnering". We add explicit terms to indicate when and instance simply stops, and when it refuses to continue mid-protocol.

The notions are compatible with both the FtG and RoR methodologies.

A full recap of the model we propose (taking into account our discussion on uniqueness of partners in Section 4) can be found in Appendix A.

3.1 The definition itself

Status of instances and partnering.

Halting. An instance halts if it stops sending and receiving messages, and ceases to compute anything.

Halting can either be "good" (i.e. with a SK) or "bad" (i.e. without a SK).

Accepting. An instance $\mathcal{U}^i$ accepts if and only if $sid^i_{\mathcal{U}}$ is set to a non-ε value. Accepting means that $\mathcal{U}^i$ believes it is holding enough information to compute a SK. The instance has not necessarily halted.

This formally includes the possibility that it may have actually computed the SK value, but is not yet willing to use it.

Terminating. An instance $\mathcal{U}^i$ terminates if and only if $sk^i_{\mathcal{U}}$ is set to a non-ε value. If an instance terminates, it halts. Terminating means $\mathcal{U}$ believes it holds a good SK, and is now willing to use it in higher-level applications. $\mathcal{U}^i$ will no longer send nor receive PAKE protocol messages.

If an instance terminates, it accepts, or has previously accepted. In both cases, $sid^i_{\mathcal{U}}$ is set to a non-ε value and remains so. If an instance accepts, it has not necessarily terminated.[5]

Aborting. An instance $\mathcal{U}^i$ aborts if it halts without having terminated.

In other words, it has stopped participating in the protocol exchange, and is unwilling to assign a value to $sk^i_{\mathcal{U}}$. Aborting can very well happen after accepting: Just imagine an instance holding a SK, but waiting for the last confirmation message to finally start using this SK.

Semi-partnering. $\mathcal{U}^i$ and $\mathcal{V}^j$ are semi-partnered if **1)** one is a client and one is a server, **2)** $pid^i_{\mathcal{U}} = \mathcal{V}$ and $pid^j_{\mathcal{V}} = \mathcal{U}$, **3)** $sid^i_{\mathcal{U}} \neq \varepsilon$, $sid^j_{\mathcal{V}} \neq \varepsilon$, and $sid^i_{\mathcal{U}} = sid^j_{\mathcal{V}}$.

Partnering. $\mathcal{U}^i$ and $\mathcal{V}^j$ are partnered if **1)** they are semi-partnered and **2)** $sk^i_{\mathcal{U}} \neq \varepsilon$, $sk^j_{\mathcal{V}} \neq \varepsilon$, and $sk^i_{\mathcal{U}} = sk^j_{\mathcal{V}}$.

By definition, if an instance is semi-partnered to another, it has accepted, and holds a SID. If it is partnered to another, it has terminated and holds a SK.

[5] We stuck to the idea in [5] that accepting may happen before terminating, even though the term "accepting" seems better suited to designate "successful termination". We did this because the original BPR model is still the most used, so it is probable that this is how the terminology is commonly understood.

Explicitly defining semi-partnering and partnering in this way seems to be the only way to have a unique formal treatment of security regardless of whether instances accept and terminate at the same time. In case an instance terminates after accepting, we still want to be able to express at acceptance time that it may have another unique instance to which it is bound, hence the semi-partnering.

To properly define security in the RoR model, and in particular to eliminate the bug identified in Section 2.3, it will be convenient to have the following notion:

Partnering graph. A partnering graph is a graph with instances for nodes. Two nodes have an edge if and only if the corresponding instances are partners.

Correctness. Let $\mathcal{U}$ be a client and $\mathcal{V}$ be a server. If $\mathcal{U}^i$ with $pid^i_{\mathcal{U}} = \mathcal{V}$ and $\mathcal{V}^j$ with $pid^j_{\mathcal{V}} = \mathcal{U}$ run the protocol fully and correctly, $\mathcal{U}^i$ and $\mathcal{V}^j$ are partnered.

Remark: Correctness can be formulated formally using *matching conversations* [6]. Note that correctness says that matching conversations lead to partnering. However, just because two instances are partnered does not mean that they have had a matching conversation. Intuitively, the protocol should provide a session key that is secure for use once the instances are partnered.

Queries in the FtG and RoR models. The list of queries in Section 2.1 remains exactly the same. We only re-write the test and reveal queries with the technical terms.

We begin by revisiting ***freshness*** for the FtG model: an instance $\mathcal{U}^i$ is said to be *fresh* if it is in a partnering graph in which no instance has been the target of a reveal query.

- (FtG) $\mathsf{test}(\mathcal{U}, i)$: If $\mathcal{U}^i$ has not terminated or is not fresh, this returns $\perp$. Otherwise, $\mathcal{CH}$ flips a coin b outside of $\mathcal{A}$'s view. If $b = 0$, a random string R is drawn from the session key space and $tk^i_{\mathcal{U}} \leftarrow R$. Otherwise, $tk^i_{\mathcal{U}} \leftarrow sk^i_{\mathcal{U}}$. $tk^i_{\mathcal{U}}$ is then returned to $\mathcal{A}$. The test query may only be used once in the game.
- (FtG) $\mathsf{reveal}(\mathcal{U}, i)$: If $\mathcal{U}^i$ has not terminated or if it is part of a partnering graph in which an instance has been tested, the query returns $\perp$. Otherwise, it returns $sk^i_{\mathcal{U}}$ to $\mathcal{A}$.
- (RoR) $\mathsf{test}(\mathcal{U}, i)$: If $\mathcal{U}^i$ has not terminated, $\perp$ is returned. Otherwise, suppose first that $b = 0$. If $\mathcal{U}^i$ is not partnered to any instance, or no instance in the partnering graph it is a part of was subjected to a test query, $\mathcal{CH}$ selects a random R from the session key space and sets $tk^i_{\mathcal{U}} \leftarrow R$. If $\mathcal{U}^i$ is within a partnering graph where some instance $\mathcal{V}^j$ that has been tested, $\mathcal{CH}$ sets $tk^i_{\mathcal{U}} \leftarrow tk^j_{\mathcal{V}}$. Suppose now that $b = 1$. In this case, $\mathcal{CH}$ sets $tk^i_{\mathcal{U}} \leftarrow sk^i_{\mathcal{U}}$. Then, $\mathcal{A}$ receives $tk^i_{\mathcal{U}}$.

It should be clear that the introduction of the partnering graph eliminates the bug in the RoR model. Of course, it is merely a tool to keep the definitions consistent; as we have pointed out before, one would want at most one partner to exist. The problem of partner uniqueness is studied in Section 4.

3.2 Examples of how it functions

We give here a few examples of protocol structures that fit our definition in various ways. In what follows, m and μ basically represent the main protocol flows, i.e. the messages that a shared secret is usually computed with. The symbols *sid* and *sk* designate the SID and SK. The k and κ values are *confirmation codes*; their role is to prove to the other party that the same shared secret was computed at both ends of the protocol run. As such, they must be computed from, or at the same time as, the shared secret.

We look at two-pass, three-pass, and four-pass protocols, but one can easily construct similar examples with protocols having more messages.

In practice, the SID is usually taken to be the concatenation of $\mathcal{C}$, $\mathcal{S}$, m, and μ. This makes the randomness of both parties an input to SID. We shall return to this point later.

Acceptance and termination occur in one step for both the client and the server. Examples of protocols like this are OMDHKE in [11] and EKE2 in [5]. The former achieves *explicit authentication of the server to the client*, as in Figure 1 and the latter achieves *implicit authentication*. (Take Figure 1 and remove all mention of κ and k.) Obviously, in two-pass protocols the parties involved have no other choice but to accept and terminate at the same time. Also, it is clear that the receiver of the first protocol message cannot be assured that it is talking to a live instance.

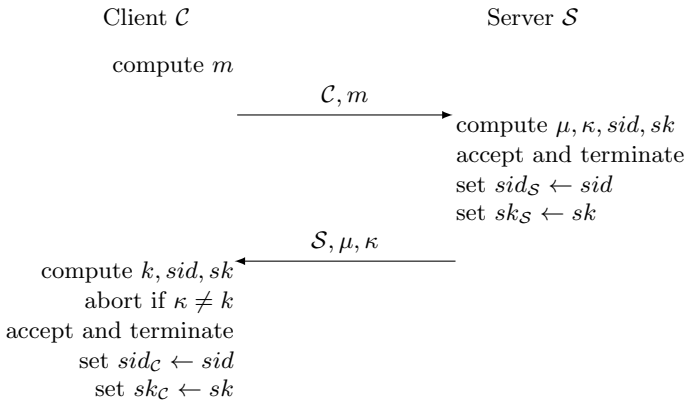


Fig. 1.

One party accepts and terminates in one step, the other accepts first and terminates later. In this class of protocols, we find e.g. the protocol of Groce and Katz [19], the OEKE protocol from [10], and the F + PaKE protocol from [17]. The Groce-Katz protocol achieves mutual authentication, as in Figure 2. OEKE and F + PaKE only achieve explicit authentication of the client to the server. (Remove all mention of κ_1 and k_1 in Figure 2.) In both cases, acceptance by the server occurs right before sending the second message, and termination occurs at the very end.

After $\mathcal{C}$ sends the last message, but before this message is received by $\mathcal{S}$, $\mathcal{C}$ and $\mathcal{S}$ are semi-partnered.

In this class of protocols, the code κ_2 and SK sk need not be computed by the server once it receives the first message (as shown in Figure 2); this can be postponed to the end.[6]

Both parties accept and terminate in two steps. This happens for instance in the AMP protocol from [28], see Figure 3. Both client and server accept at different stages. Note that sometimes protocols of this form can regroup certain messages in order to

[6] This may even be desirable for efficiency reasons.

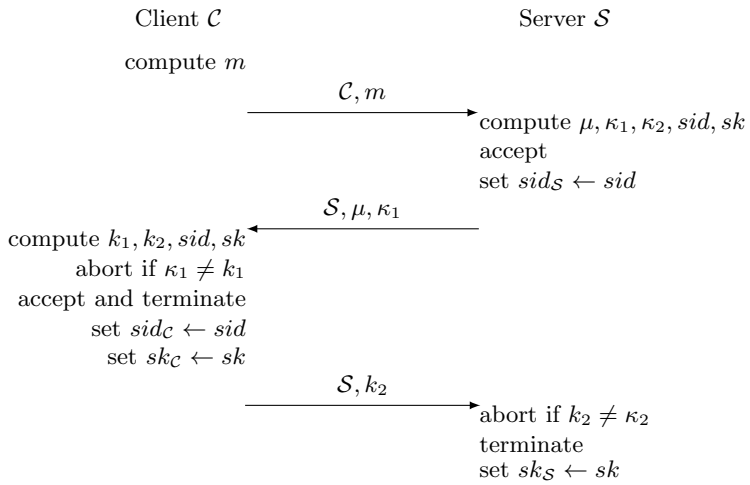


Fig. 2.

yield n-pass rather than $(n+1)$-pass protocols. (However, four-pass protocols have some advantages over three-pass ones too, e.g. in [29] Kwon reports on a practical *multiple-password* online guessing attack against three-pass protocols, that four-pass protocols do not suffer from. This attack depends on the server's *actual response time* in processing an authentication request, so falls out of the scope of currently used security models.)

As in the previous case, the actual computation of the SK (which is different from it being formally accepted as usable) can be moved around somewhat, this time both at the server and client end.

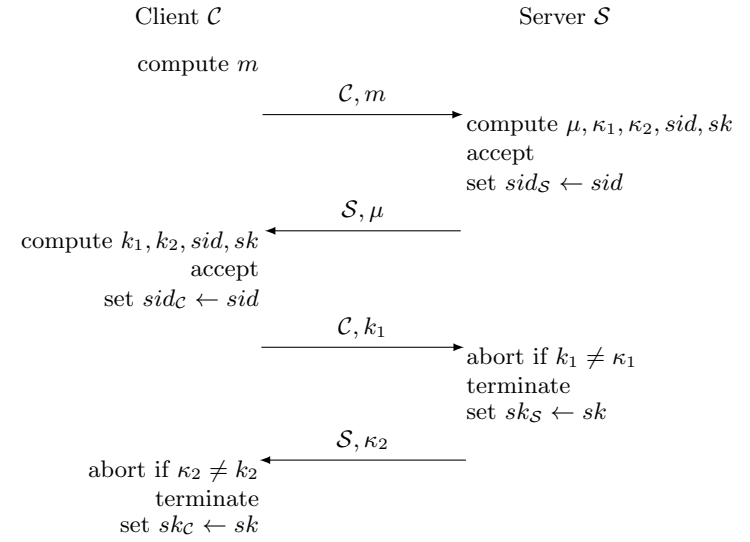


Fig. 3.

4 The quality of partner uniqueness

In this section, we examine just exactly "how unique" partners can be in AKEs, with a special treatment of PAKEs, ***assuming we stick to the definitions of partnering and testing that do not take into account our modifications, in particular with no notion of partnering graph. We also assume a partnering definition that does not mention uniqueness, e.g. that of [17, 22, 19, 25, 26].*** This serves as further evidence that partnering and partner uniqueness must be carefully treated, and probably separately.

In general, we note that it seems uniqueness of partners has completely disappeared from AKE requirements in a very large proportion of AKE research papers. Also, when uniqueness is mentioned, it is not even clear how probable this uniqueness should be (e.g. [5, 3, 26]). This is odd, since partner uniqueness was actually considered in perhaps the very first AKE modeling paper [6]. Of course, it is quite natural to expect that for *all* AKEs, if an instance runs the protocol it should have at most *one* partner with *overwhelming* probability, but this is rarely explicitly mentioned. One reason for this may be that usually an attempt is made at funneling all of the desirable security properties into the definition of session key security, but there are two problems with this, which we show below.

First, even in the general case not all security properties can be treated through this mechanism, and it so happens that uniqueness of partners is one of them. Secondly, in the PAKE case - where the optimal bound for semantic security is not even a negligible value - the property may formally fail altogether.

4.1 An obstacle caused by the test query

We begin with some flawed reasoning that applies to all AKE settings, and concerning the link between multiple partnering of instances and SK security. Again, we stress that this is specific to ***a partnering definition without partnering graphs, and without built-in uniqueness.*** Also, we use ***the RoR setting as an example.***

What we cannot do... Let AKE be an authenticated key exchange protocol. In the game played in Section 2.1, we can always consider the event "there exist distinct users $\mathcal{U}$ and $\mathcal{V}$, and distinct instances $\mathcal{U}^i$, $\mathcal{V}^j$, and $\mathcal{V}^k$ such that $\mathcal{U}^i$ and $\mathcal{V}^j$ are partnered and $\mathcal{U}^i$ and $\mathcal{V}^k$ are partnered", which we denote MP (for Multiple Partnering). For any adversary $\mathcal{B}$, let $\mathsf{Adv}^{mp}(\mathcal{B}) := \mathsf{Pr}[\mathsf{MP}]$. Our objective is to relate $\mathsf{Adv}^{mp}(\mathcal{B})$ to $\mathsf{Adv}^{s}(\mathcal{A})$ for a suitably constructed $\mathcal{A}$. Ultimately, it would be nice if the negligibility of the latter implied that of the former.

• ***Construction $\mathfrak{C}$:*** *Fix some adversary $\mathcal{B}$, and consider adversary $\mathcal{A}$, trying to break the semantic security of* AKE*, designed as follows. $\mathcal{A}$ runs exactly like $\mathcal{B}$, but examining whether or not $\mathcal{B}$ can cause* MP *to occur. Whenever an instance terminates, $\mathcal{A}$ checks to see if* MP *has happened. If $\mathcal{B}$ halts and* MP *never happened, $\mathcal{A}$ flips a coin b' and outputs b'. As soon as* MP *happens (if it does), $\mathcal{A}$ stops $\mathcal{B}$, and studies the involved instances. Let $\mathcal{U}^i$, $\mathcal{V}^j$, and $\mathcal{V}^k$ be these instances. $\mathcal{A}$ performs one* test *query on $\mathcal{V}^j$ and another on $\mathcal{V}^k$, receiving $tk^j_{\mathcal{V}}$ and $tk^k_{\mathcal{V}}$. If $tk^j_{\mathcal{V}} = tk^k_{\mathcal{V}}$, $\mathcal{A}$ sets $b' \leftarrow 1$, and otherwise $b' \leftarrow 0$. $\mathcal{A}$ then outputs b' and halts.*

At this point, in the event that MP does indeed occur, it is tempting to directly conclude that since the instances $\mathcal{V}^j$ and $\mathcal{V}^k$ are clearly not partners (since their PIDs do not correspond) but do hold identical session keys (since they are each partnered to $\mathcal{U}^i$), the test queries performed should give either independent random keys if $b = 0$

or identical keys if $b = 1$. From this, one immediately sees that if semantic security of the session key holds, then MP may only occur with negligible probabilility as well. However, we cannot make this assertion, because it may be that $\mathcal{U}^i$ has also had a test query performed on it. It the RoR model, this would make all keys identical no matter the value of b.[7]

Unfortunately, there does not seem to be any way around this obstruction. It looks as though one would have to somehow need the probability that $\mathcal{A}$ makes a test query on $\mathcal{U}^i$ to be itself a negligible value, but this is not justifiable. Thus, in order to prove anything in this way, one has to consider a more restricted version of MP.

...what we can do... We consider event MP^* defined as "there exist distinct users $\mathcal{U}$ and $\mathcal{V}$, and distinct instances $\mathcal{U}^i$, $\mathcal{V}^j$, and $\mathcal{V}^k$, such that **1)** $\mathcal{U}^i$ and $\mathcal{V}^j$ are partnered, **2)** $\mathcal{U}^i$ and $\mathcal{V}^k$ are partnered, and **3)** no test query was performed on $\mathcal{U}^i$."

We also consider construction $\mathfrak{C}^*$ which is basically identical to construction $\mathfrak{C}$, except that $\mathcal{A}$ looks out for event MP^* rather than MP. The next lemma precisely relates $\mathsf{Adv}^{mp^*}(\mathcal{B})$ to $\mathsf{Adv}^s(\mathcal{A})$.

Lemma 1 *It holds that* $\mathsf{Adv}^s(\mathcal{A}) = \big(1 - \frac{1}{2^{\ell-1}}\big)\mathsf{Adv}^{mp^*}(\mathcal{B})$.

Proof: We have

$$\mathsf{Pr}[S] = \mathsf{P}[\mathsf{S}|\mathsf{MP}^*]\mathsf{Pr}[\mathsf{MP}^*] + \mathsf{P}[\mathsf{S}|\neg\mathsf{MP}^*]\mathsf{Pr}[\neg\mathsf{MP}^*] \tag{1}$$

Conditioned on MP^* not having occurred, by definition of $\mathcal{A}$ we have $\mathsf{P}[\mathsf{S}|\neg\mathsf{MP}^*] = \frac{1}{2}$. If MP^* has occurred, all three instances should hold the same SK sk. However, since $pid^j_{\mathcal{V}} = pid^k_{\mathcal{V}} = \mathcal{U}$, $\mathcal{V}^j$ and $\mathcal{V}^k$ are not partnered. Finally, no test query was performed on $\mathcal{U}^i$. Therefore, if $b = 0$ the $\mathsf{test}(\mathcal{V}, j)$ and $\mathsf{test}(\mathcal{V}, k)$ queries output $tk^j_{\mathcal{V}}$ and $tk^k_{\mathcal{V}}$ independently and randomly. Thus, $\mathcal{A}$ will output the correct bit value unless these keys collide, which may happen with probability $\frac{1}{2^\ell}$. Plugging these values into Equation 1 gives $\mathsf{Pr}[S] = \frac{1}{2}\big(1 - \frac{1}{2^{\ell-1}}\big)\mathsf{Pr}[\mathsf{MP}] + \frac{1}{2}$. Taking the advantage function formulas finishes the proof. ■

On one hand, this lemma shows that for AKE protocols for which Adv^s is required to be negligible (such as PKI-AKEs, or SSK-AKEs), the probability that MP^* occurs is negligible as well. On the other hand, for PAKEs, this no longer holds: All we can say is

$$\mathsf{Adv}^{mp^*}(\mathcal{B}) \leq \Big(\frac{2^{\ell-1}}{2^{\ell-1}-1}\Big)\frac{Cn_{se}}{|\mathsf{PW}|} \tag{2}$$

where C and n_{se} are as in Paragraph 2.1.

...and what we can assert. Hence, we can basically gather the following three points:

- **1)** Adv^s's negligibility does not immediately imply that event MP will occur with negligible probability. Notably, this concerns the PKI and SSK cases.
- **2)** Adv^s's negligibility only implies that event MP^*will occur with negligible probability. This is certainly a desirable guarantee, but remains weaker than MP.
- **3)** In the PAKE case, even the restricted event MP^* has no immediate reason to be negligible at all.

Point **3)** is further illustrated in Paragraph 4.2 below, where we propose a PAKE that is secure according the the model of Paragraph 2.1, but at the same time can cause MP^* to occur with a probability nearly optimally respecting the bound in Equation 2.

[7] Similar reasoning shows that the FtG model suffers from the phenomenon as well, basically because if $\mathcal{U}^i$ is tested, the freshness condition prohibits testing of the two other instances. Thus, our observation is valid "beyond RoR".

4.2 A "secure" PAKE protocol where non-negligible multiple partnering may occur

We now describe our flawed protocol, dubbed P. It actually is not a "pure" PAKE, since the client authenticates the server using a public key, while the server authenticates the client using a password. We were unable to construct a suitable example in the "pure" setting, but we believe our point is still valid. (However, a "pure" example would be very interesting.)

Setup. From here on, $(\mathsf{SKeyGen}, \mathsf{Sig}, \mathsf{Ver})$ is a strongly-EU-CMA-secure (see [4])[8] signature scheme and $(\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a CCA-2-secure public key encryption scheme. We assume that there can be many different client identities, but only one server identity $\mathcal{S}$. Thus, there are many clients, and many client instances, but only one server, and many server instances. The KeyGen algorithms are run once to obtain public key/secret key pairs (pk_E, sk_E) and (pk_S, sk_S) for encryption and signing respectively. Each client $\mathcal{C}$ has its password $pw_{\mathcal{C}}$ registered at server $\mathcal{S}$, and has (pk_E, pk_S), say, hardcoded into its software specification. Server $\mathcal{S}$ holds the full password file $\{pw_{\mathcal{C}}\}_{\mathcal{C}}$, as well as (sk_E, sk_S).[9]

Running P. The protocol flows are shown in Figure 4. First, the server $\mathcal{S}$ pings the client $\mathcal{C}$ with a signed nonce. $\mathcal{C}$ then verifies the signature, accepts without terminating, and returns an encryption of the nonce it just received, a fresh nonce of its own, a session key it selects itself randomly, and its password. Upon receiving this encryption, $\mathcal{S}$ decrypts and checks if the password matches $\mathcal{C}$'s identity and if it recognizes its own nonce. If so, it accepts and terminates, validates the session key, and returns to $\mathcal{C}$ a new signature on both nonces. Finally, if the signature verifies $\mathcal{C}$ terminates and validates the session key.

Upon acceptance, for our protocol the SID is set as being $(M, \mathcal{C}, \mathcal{S})$, where M is $\mathcal{C}$'s chosen nonce. As the reader may expect, the fact that the SID is a function of *only the client's nonce* is what causes trouble.

Discussion. On one hand, P definitely looks like a bad protocol, given that it is absolutely littered with red flags:

- **1)** The SID depends only on the random bits of one of the two parties involved;
- **2)** The session key is completely determined by one of the two parties involved;
- **3)** The session key and identifier are completely decoupled.

These points certainly go against well-established design principles that PAKEs commonly follow. As for item **4)**, it illustrates the need to study uniqueness of partners as a security property in its own right in the PAKE case.

On the other hand, P actually satisfies an appropriately modified BPR-style definition of security for PAKEs:

Theorem 1 *Protocol P is a secure PAKE in that for any efficient adversary $\mathcal{A}$, when $\mathcal{A}$ plays the RoR game against challenger $\mathcal{CH}$ as described in Section 2.1, we have $\mathsf{Adv}^s(\mathcal{A}) \leq \frac{n_{se}}{|\mathsf{PW}|} + \mathsf{negl}(\lambda)$, where n_{se} is an upper bound on the number of send queries made by $\mathcal{A}$.*

[8] The fact that signatures are *strongly* secure is used to make the security proof simpler, but is not strictly necessary.

[9] One may think of a setup of this sort as being implemented e.g. for a large group of employees in a company.

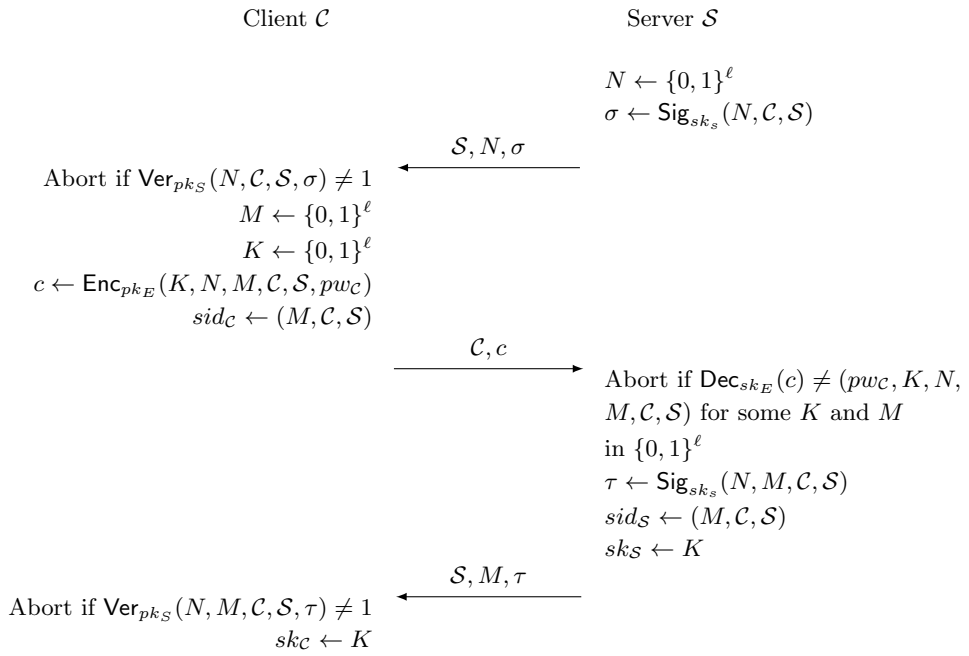


Fig. 4. The P protocol.

Proof: The proof of this theorem (using the standard game-hopping technique) will appear in the full version of the paper. ■

Furthermore, P also suffers from point **4)** below:

- **4)** Event MP^* may occur with non-negligible probability.

A demonstration of this is in the next paragraph.

Partnering a client instance to two server instances. We construct a specific attacker $\mathcal{B}$ in the security model described in Section 2.1. $\mathcal{B}$ is trying to cause the event MP^*.

$\mathcal{B}$ first initializes a client instance $\mathcal{C}^1$ and server instance $\mathcal{S}^1$ with $pid_\mathcal{C}^1 = \mathcal{S}$ and $pid_\mathcal{S}^1 = \mathcal{C}$. Next, it performs an execute query on these instances. They now share a SK sk and a SID $(M, \mathcal{C}, \mathcal{S})$. Then, it performs a test query on $\mathcal{S}^1$ to get sk. Now, for $j = 2, ...$ it repeats the following steps until some server instance $\mathcal{S}^j$ accepts:

- **1)** It initializes $\mathcal{S}^j$ with $pid_\mathcal{S}^j = \mathcal{C}$, and instructs it to send the first protocol message $\mathcal{S}, N^j, \sigma$;
- **2)** It chooses $pw^j \leftarrow \mathsf{PW}$ randomly and computes $c \leftarrow \mathsf{Enc}_{pk_E}(sk, N^j, M, \mathcal{C}, \mathcal{S}, pw^j)$;
- **3)** It sends c^j to $\mathcal{S}^j$ and observes whether $\mathcal{S}^j$ accepts or not.

When some $\mathcal{S}^j$ finally accepts, $\mathcal{B}$ has guessed the right password $pw_\mathcal{C}$, and the instances $\mathcal{C}^1$ and $\mathcal{S}^j$ are *partnered*: $sid_\mathcal{S}^j = sid_\mathcal{C}^1 = (M, \mathcal{C}, \mathcal{S})$ and $sk_\mathcal{S}^j = sk_\mathcal{C}^1 = sk$.

In the above scenario, it should be clear that if t is the number of $\mathcal{S}^j$ instances (for $j \geq 2$) used in order to succeed, we have

$$\mathsf{Adv}^{mp^*}(\mathcal{B}) = \mathsf{Pr}[\mathsf{MP}^*] = \frac{t}{|\mathsf{PW}|}$$

which is certainly not negligible. Also, since we have $t \leq n_{se}-2$, this is well in accordance with Equation 2 (taking $C = 1$, the best possible constant), and thus with the required bound on semantic security.

4.3 Lessons learned on requirements

We should conclude that MP has to be considered as an event to render negligible in its own right for both AKEs, where the semantic security of SK only provides a safety net in that it renders a more restricted event negligible, and PAKEs, where there is no safety net at all. Fortunately, most AKEs in the literature do not have these problems because of the way partnering is instantiated by concrete protocols. In particular, as previously stated most PAKEs build the SIDs from concatenations of almost all messages, so all parties' random values are involved, and uniqueness of SIDs, and therefore of partners, is a trivially verified matter. However, as we have shown, it can formally fail in "non-concatenation" cases, and so it is better off being a stated and formally proven requirement, no matter how trivial. Thus it seems worthwhile to add to the security model the property that MP should be negligible *all the time*. In fact, using our language from Section 3, multiple semi-partnering should be negligible all the time.

In the particular case where SIDs are used to establish partnering - this is almost always the case in BPR-style models - The simplest way to do this would be to prove that the event SID is negligible, where we define SID to be "there exist more than two instances that share the same non$-\varepsilon$ SID". In other words, we add the following point to our model:

Partner uniqueness: Let SID be the event that "there exist more than two instances that have the same non-ε SID". Let $\mathsf{Adv}^{sid}(\mathcal{A}) := \mathsf{Pr}[\mathsf{SID}]$. We say that PAKE achieves unique partnering if for any PPT $\mathcal{A}$ the function Adv^{sid} is negligible. Note that the quality of the long-term keying material here no longer should have an influence on the security bound we require.

For a complete model description, we refer to Appendix A.

5 Conclusion and future work

In this paper, we have shown that there are multiple BPR-style definitions of security for PAKEs in the literature, and have attempted to unify them. In the process, we found a way to solve a bug in the model of [3]. Finally, we showed that uniqueness of partners is a property worth establishing explicitly, similarly to explicit authentication and semantic security of the session key. Specifically, it should hold with overwhelming probability even in the PAKE case, where the other main security properties can only be ensured with non-negligible probability.

As far as this study goes, it should be extended to include long-term key corruptions. Also, it would be extremely interesting to find a counter-example similar to protocol P in the "pure PAKE" setting. Finally, it may be worthwhile to further refine BPR-style models by adding a specific variable to designate the shared secret computed by protocol participants. It seems reasonable that this variable would be non-ε when semi-partnering occurs.

Acknowledgments. We would like to thank the reviewers for their comments. The author is supported by the *Fonds National de la Recherche, Luxembourg*, via the CORE project AToMS and the INTER project SEQUOIA.

References

1. Abdalla, M., Benhamouda, F., MacKenzie, P.: Security of the J-PAKE Password-Authenticated Key Exchange Protocol. In: 2015 IEEE Symposium on Security and Privacy (2015)
2. Abdalla, M., Benhamouda, F., Pointcheval, D.: Public-key encryption indistinguishable under plaintext-checkable attacks. In: Katz, J. (ed.) Public-Key Cryptography – PKC 2015, Lecture Notes in Computer Science, vol. 9020, pp. 332–352. Springer Berlin Heidelberg (2015), http://dx.doi.org/10.1007/978-3-662-46447-2_15
3. Abdalla, M., Fouque, P.A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) Public Key Cryptography - PKC 2005, Lecture Notes in Computer Science, vol. 3386, pp. 65–84. Springer Berlin Heidelberg (2005), http://dx.doi.org/10.1007/978-3-540-30580-4_6
4. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology. pp. 83–107. EUROCRYPT '02, Springer-Verlag, London, UK, UK (2002), http://dl.acm.org/citation.cfm?id=647087.715701
5. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure Against Dictionary Attacks. In: Preneel, B. (ed.) Advances in Cryptology – EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer (2000)
6. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) Advances in Cryptology – CRYPTO '93. LNCS, vol. 773, pp. 232–249. Springer (1993)
7. Bellovin, S.M., Merritt, M.: Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In: 1992 IEEE Computer Society Symposium on Research in Security and Privacy, May 4-6, 1992. pp. 72–84 (1992)
8. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for sphfs and efficient one-round pake protocols. In: Canetti, R., Garay, J. (eds.) Advances in Cryptology CRYPTO 2013, Lecture Notes in Computer Science, vol. 8042, pp. 449–475. Springer Berlin Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-40041-4_25
9. Boyko, V., MacKenzie, P.D., Patel, S.: Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In: Preneel, B. (ed.) Advances in Cryptology – EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer (2000)
10. Bresson, E., Chevassut, O., Pointcheval, D.: Security Proofs for an Efficient Password-based Key Exchange. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) ACM Conference on Computer and Communications Security. pp. 241–250. ACM (2003)
11. Bresson, E., Chevassut, O., Pointcheval, D.: New Security Results on Encrypted Key Exchange. In: Bao, F., Deng, R.H., Zhou, J. (eds.) Public Key Cryptography. LNCS, vol. 2947, pp. 145–158. Springer (2004)
12. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science. pp. 136–. FOCS '01, IEEE Computer Society, Washington, DC, USA (2001), http://dl.acm.org/citation.cfm?id=874063.875553
13. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally Composable Password-Based Key Exchange. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer (2005)
14. Choo, K.K., Boyd, C., Hitchcock, Y.: Examining indistinguishability-based proof models for key establishment protocols. In: Roy, B. (ed.) Advances in Cryptology - ASIACRYPT 2005, Lecture Notes in Computer Science, vol. 3788, pp. 585–604. Springer Berlin Heidelberg (2005), http://dx.doi.org/10.1007/11593447_32
15. Cremers, C.: Examining indistinguishability-based security models for key exchange protocols: The case of ck, ck-hmqv, and eck. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. pp. 80–91. ASIACCS '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/1966913.1966925
16. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theor. 22(6), 644–654 (Sep 2006), http://dx.doi.org/10.1109/TIT.1976.1055638

17. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) Advances in Cryptology EUROCRYPT 2003, Lecture Notes in Computer Science, vol. 2656, pp. 524–543. Springer Berlin Heidelberg (2003), http://dx.doi.org/10.1007/3-540-39200-9_33
18. Goldreich, O., Lindell, Y.: Session-key generation using human passwords only. In: Kilian, J. (ed.) Advances in Cryptology CRYPTO 2001, Lecture Notes in Computer Science, vol. 2139, pp. 408–432. Springer Berlin Heidelberg (2001), http://dx.doi.org/10.1007/3-540-44647-8_24
19. Groce, A., Katz, J.: A new framework for efficient password-based authenticated key exchange. In: Proceedings of the 17th ACM Conference on Computer and Communications Security. pp. 516–525. CCS '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1866307.1866365
20. Halevi, S., Krawczyk, H.: Public-key cryptography and password protocols. ACM Trans. Inf. Syst. Secur. 2(3), 230–268 (Aug 1999), http://doi.acm.org/10.1145/322510.322514
21. Jablon, D.P.: Strong Password-Only Authenticated Key Exchange. ACM SIGCOMM Computer Communication Review 26(5), 5–26 (1996)
22. Jiang, S., Gong, G.: Password based key exchange with mutual authentication. In: Handschuh, H., Hasan, M. (eds.) Selected Areas in Cryptography, Lecture Notes in Computer Science, vol. 3357, pp. 267–279. Springer Berlin Heidelberg (2005), http://dx.doi.org/10.1007/978-3-540-30564-4_19
23. Katz, J., Ostrovsky, R., Yung, M.: Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In: Pfitzmann, B. (ed.) Advances in Cryptology – EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer (2001)
24. Katz, J., Ostrovsky, R., Yung, M.: Efficient and secure authenticated key exchange using weak passwords. JOURNAL OF THE ACM 57(1) (2009)
25. Katz, J., Vaikuntanathan, V.: Smooth projective hashing and password-based authenticated key exchange from lattices. In: Matsui, M. (ed.) Advances in Cryptology ASIACRYPT 2009, Lecture Notes in Computer Science, vol. 5912, pp. 636–652. Springer Berlin Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-10366-7_37
26. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) Theory of Cryptography, Lecture Notes in Computer Science, vol. 6597, pp. 293–310. Springer Berlin Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-19571-6_18
27. Kiefer, F., Manulis, M.: Oblivious pake: Efficient handling of password trials. Cryptology ePrint Archive, Report 2013/127 (2013), http://eprint.iacr.org/
28. Kwon, T.: Authentication and key agreement via memorable password. In: ISOC Network and Distributed System Security Symposium (2001)
29. Kwon, T.: Practical Authenticated Key Agreement Using Passwords. In: Zhang, K., Zheng, Y. (eds.) Information Security, LNCS, vol. 3225, pp. 1–12. Springer Berlin Heidelberg (2004)
30. Lucks, S.: Open key exchange: How to defeat dictionary attacks without encrypting public keys. In: Proceedings of the 5th International Workshop on Security Protocols. pp. 79–90. Springer-Verlag, London, UK, UK (1998), http://dl.acm.org/citation.cfm?id=647215.720526
31. MacKenzie, P.: The PAK Suite: Protocols for Password-Authenticated Key Exchange. DIMACS Technical Report 2002-46 (2002), (Page 7)
32. MacKenzie, P., Patel, S., Swaminathan, R.: Password-authenticated key exchange based on rsa. Int. J. Inf. Secur. 9(6), 387–410 (Dec 2010), http://dx.doi.org/10.1007/s10207-010-0120-3
33. Pointcheval, D.: Password-Based Authenticated Key Exchange. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) Public Key Cryptography PKC 2012, LNCS, vol. 7293, pp. 390–397. Springer (2012)
34. Shoup, V.: On Formal Models for Secure Key Exchange. Cryptology ePrint Archive, Report 1999/012 (1999), http://eprint.iacr.org/1999/012

A BPR-style models revisited

This appendix is just a formal recap of our complete list of requirements.

Principals and instances. An interactive game, indexed by the security parameter $\lambda \in \mathbb{N}$, is played between a challenger $\mathcal{CH}$ and an adversary $\mathcal{A}$. All of the algorithms considered are Probabilistic, Polynomial-Time (PPT) in λ.

At the beginning of the game, there is a fixed set of *principals* (or *users*), partitioned into non-empty sets of *clients* $\mathcal{C}$ and *servers* $\mathcal{S}$. Each client $\mathcal{C}$ is assigned a *password* $pw_{\mathcal{C}}$ drawn uniformly at random from some finite set PW of bitstrings. Each server $\mathcal{S}$ holds the full set of all clients' passwords $\{pw_{\mathcal{C}}\}_{\mathcal{C}}$.

Adversary $\mathcal{A}$ has oracle access - via the queries described below - to any number of *instances* $\mathcal{U}^i$ ($i \in \mathbb{N}$) of any principal $\mathcal{U}$. An instance of $\mathcal{U}^i$ represents an attempt $\mathcal{U}$ makes at running the PAKE protocol over the network fully controlled by $\mathcal{A}$. An instance's objective is to compute a *Session Key* (or SK) it believes it shares with an instance $\mathcal{V}^j$ of some other principal $\mathcal{V}$. This should happen only if $\mathcal{U}^i$ thinks it is partnered (see below) to $\mathcal{V}^j$.

At any point in time, an instance $\mathcal{U}^i$ may *terminate* (see below). By this time, $\mathcal{U}^i$ should have computed **1)** a *Partner Identity* (or PID) $pid^i_{\mathcal{U}}$, a **2)** *Session Identity* (SID) $sid^i_{\mathcal{U}}$, and **3)** a SK $sk^i_{\mathcal{U}}$. The PID is a bitstring indicating the identity of the instance with which $\mathcal{U}^i$ believes it has communicated with. The SID is a bitstring serving as an identifier for both the key exchange run that just occurred, and the session in which the computed SK will subsequently serve. Often the SID is always in practice set to being the ordered concatenation of all exchanged protocol messages, except possibly the last message. At any point in time an instance may also *abort* (see below), with no SK. Once an instance has *halted* (see below) it can no longer be reused.

Status of instances and partnering.

Halting. An instance halts if it stops sending and receiving messages, and ceases to compute anything. Halting can be "good" (i.e. with a SK) or "bad" (i.e. without a SK).

Accepting. An instance $\mathcal{U}^i$ accepts if and only if $sid^i_{\mathcal{U}}$ is set to a non-ε value. Accepting means that $\mathcal{U}^i$ believes it is holding enough information to compute a SK.

Terminating. An instance $\mathcal{U}^i$ terminates if and only if $sk^i_{\mathcal{U}}$ is set to a non-ε value. If an instance terminates, it halts. Terminating means $\mathcal{U}$ believes it holds a good SK, and is now willing to use it in higher-level applications. $\mathcal{U}^i$ will no longer send nor receive PAKE protocol messages. If an instance terminates, it accepts, or has previously accepted. In both cases, $sid^i_{\mathcal{U}}$ is set to a non-ε value and remains so.

Aborting. An instance $\mathcal{U}^i$ aborts if it halts without having terminated.

Semi-partnering. $\mathcal{U}^i$ and $\mathcal{V}^j$ are semi-partnered if **1)** one is a client and one is a server, **2)** $pid^i_{\mathcal{U}} = \mathcal{V}$ and $pid^j_{\mathcal{V}} = \mathcal{U}$, **3)** $sid^i_{\mathcal{U}} \neq \varepsilon$, $sid^j_{\mathcal{V}} \neq \varepsilon$, and $sid^i_{\mathcal{U}} = sid^j_{\mathcal{V}}$.

Partnering. $\mathcal{U}^i$ and $\mathcal{V}^j$ are partnered if **1)** they are semi-partnered and **2)** $sk^i_{\mathcal{U}} \neq \varepsilon$, $sk^j_{\mathcal{V}} \neq \varepsilon$, and $sk^i_{\mathcal{U}} = sk^j_{\mathcal{V}}$. So, if an instance is semi-partnered to another, it has accepted, and holds a SID. If it is partnered to another, it has terminated and holds a SK.

Partnering graph. A partnering graph is a graph with instances for nodes. Two nodes have an edge if and only if the corresponding instances are partners.

Correctness. If $\mathcal{C}^i$ with $pid^i_{\mathcal{C}} = \mathcal{S}$ and $\mathcal{S}^j$ with $pid^j_{\mathcal{S}} = \mathcal{C}$ run the protocol fully and correctly, $\mathcal{C}^i$ and $\mathcal{S}^j$ are partnered.

Find-then-Guess.

- $\mathsf{send}(\mathcal{U}, i, m)$: $\mathcal{A}$ has message m delivered to $\mathcal{U}^i$. $\mathcal{U}^i$ processes the message according to protocol specification. To instruct an instance $\mathcal{U}$ to send the first protocol message

to entity $\mathcal{V}$, $\mathcal{A}$ makes the query with $M = \mathcal{V}$. This query is used to model arbitrary message delivery to an instance. In particular, it serves to count impersonation attacks.
- $\mathsf{execute}(\mathcal{U}, \mathcal{V}, i, j)$: The protocol is executed faithfully and completely between $\mathcal{U}^i$ and $\mathcal{V}^j$ and the resulting transcript is given to $\mathcal{A}$. $\mathcal{A}$ thus gets to see as many honest protocol runs as it wishes.
- $\mathsf{reveal}(\mathcal{U}, i)$: If $\mathcal{U}^i$ has not terminated or if it is part of a partnering graph in which an instance has been tested, the query returns $\perp$. Otherwise, it returns $sk^i_{\mathcal{U}}$ to $\mathcal{A}$.
- $\mathsf{test}(\mathcal{U}, i)$: $\mathsf{test}(\mathcal{U}, i)$: If $\mathcal{U}^i$ has not terminated or is not fresh, this returns $\perp$. Otherwise, $\mathcal{CH}$ flips a coin b outside of $\mathcal{A}$'s view. If $b = 0$, a random string R is drawn from the session key space and $tk^i_{\mathcal{U}} \leftarrow R$. Otherwise, $tk^i_{\mathcal{U}} \leftarrow sk^i_{\mathcal{U}}$. $tk^i_{\mathcal{U}}$ is then returned to $\mathcal{A}$. The test query may only be used once in the game.

Eventually, $\mathcal{A}$ halts the overall game, at which point it outputs a bit b'. If the game was halted without $\mathcal{A}$ making any test query, then $\mathcal{CH}$ privately flips a coin b.

Freshness: An instance $\mathcal{U}^i$ is said to be *fresh* if it is in a partnering graph in which no instance has been the target of a reveal query.

Real-or-Random. The send and $\mathsf{execute}$ queries are identical to those in the FtG model. However, the reveal query is no longer available. Instead, it is replaced by as many test queries as $\mathcal{A}$ wants. How they are answered depends on the value of a bit b flipped by $\mathcal{CH}$ outside of $\mathcal{A}$'s view *at the beginning of the game.*

- $\mathsf{test}(\mathcal{U}, i)$: If $\mathcal{U}^i$ has not terminated, $\perp$ is returned. Otherwise, suppose first that $b = 0$. If $\mathcal{U}^i$ is not partnered to any instance, or no instance in the partnering graph it is a part of was subjected to a test query, $\mathcal{CH}$ selects a random R from the session key space and sets $tk^i_{\mathcal{U}} \leftarrow R$. If $\mathcal{U}^i$ is within a partnering graph where some instance $\mathcal{V}^j$ that has been tested, $\mathcal{CH}$ sets $tk^i_{\mathcal{U}} \leftarrow tk^j_{\mathcal{V}}$. Suppose now that $b = 1$. In this case, $\mathcal{CH}$ sets $tk^i_{\mathcal{U}} \leftarrow sk^i_{\mathcal{U}}$. Then, $\mathcal{A}$ receives $tk^i_{\mathcal{U}}$.

The slight complication arising in case $b = 0$ is that even if the SKs assigned are random, they must at least remain consistent across instances that should hold the same keys. As in FtG, at any point in time $\mathcal{A}$ may halt the game and output a bit b'

Technically defining security. In both FtG and RoR, one usually considers three security properties: SK security, Client-to-Server (C2S) authentication, and Server-to-Client (S2C) authentication. We explicitly add to this uniqueness of partners by requiring that SIDs are shared by at most two instances (SID).

SK security: Let S be the event that "$b' = b$ at the end of the game". $\mathcal{A}$'s natural advantage is defined to be $\mathsf{Adv}^s(\mathcal{A}) := 2\mathsf{Pr}[\mathsf{S}] - 1$. A PAKE protocol is said to have semantically secure SKs if there exists a non-zero constant $C \in \mathbb{N}$ such that for any PPT $\mathcal{A}$, there exists a negligible (in λ) function negl with the property that $\mathsf{Adv}^s(\mathcal{A}) \leq \frac{Cn_{se}}{|\mathsf{PW}|} + \mathsf{negl}(\lambda)$ where n_{se} is an upper bound on the number of send queries the adversary makes.

Authentication: Let $\mathsf{C2S}$ be the event that "there exists some server instance $\mathcal{S}^j$ that is ready to use a SK, but has not had a correct exchange with a client instance." Here we simply set $\mathsf{Adv}^{c2s}(\mathcal{A}) := \mathsf{Pr}[\mathsf{C2S}]$, and we say that a PAKE achieves client-to-server authentication if there exists a non-zero C such that for any PPT $\mathcal{A}$, there exists a negligible function negl with the property that $\mathsf{Adv}^{c2s}(\mathcal{A}) \leq \frac{Cn_{se}}{|\mathsf{PW}|} + \mathsf{negl}(\lambda)$. Server-to-client authentication is defined similarly.

Partner uniqueness: Let SID be the event that "there exists more than two instances that have the same non-ε SID". Let $\mathsf{Adv}^{sid}(\mathcal{A}) := \mathsf{Pr}[\mathsf{SID}]$. We say that PAKE achieves unique partnering if for any PPT $\mathcal{A}$ the function Adv^{sid} is negligible.

ITSME: Multi-modal and Unobtrusive Behavioural User Authentication for Smartphones

Attaullah Buriro[1], Bruno Crispo[2,1], Filippo Del Frari[1], Jeffrey Klardie[3], and Konrad Wrona[4]

[1] Department of Information Engineering and Computer Science, University of Trento, Italy,
attaullah.buriro@unitn.it, bruno.crispo@unitn.it, filippo.delFrari@unitn.it
[2] DistrNet, KULeuven, Belgium,
bruno.crispo@cs.kuleuven.be
[3] Vrije Universiteit Amsterdam, The Netherlands
[4] NATO Communications and Information Agency, The Hague, Netherlands,
konrad.wrona@ncia.nato.int

Abstract. In this paper, we propose a new multi-modal behavioural biometric that uses features collected while the user slide-unlocks the smartphone to answer a call. In particular, we use the slide swipe, the arm movement in bringing the phone close to the ear and voice recognition to implement our behaviour biometric. We implemented the method on a real phone and we present a controlled user study among 26 participants in multiple scenario's to evaluate our prototype. We show that for each tested modality the Bayesian network classifier outperforms other classifiers (Random Forest algorithm and Sequential Minimal Optimization). The multimodal system using slide and pickup features improved the unimodal result by a factor two, with a FAR of 11.01% and a FRR of 4.12%. The final HTER was 7.57%.

Keywords: Smartphone, Behavioral Biometrics, Sensors, Transparent Authentication

1 Introduction

The last decade mobile handheld devices have gone through a major evolution; smaller yet more powerful processors, better batteries and improved hardware such as gps, wifi and connectivity chips are all developments that enabled this progression. Most relevant examples are the Android platform launched by Google in 2008 [1] and the iPhone smartphone by Apple in 2007 [2]. Since their introduction these devices have overtaken most competitors and together captured a very dominant market share: in 2014 Android and iOS combined account for 96.3% of the smartphone operating system market [3].

Continuing hardware improvements combined with extensive user research have resulted in highly capable smartphones that provide users with rich communication capabilities. While this improves users' lives on one hand, it brings very serious security and privacy threats to the user on the other. A typical modern smartphone allows the user to do mobile banking, have full control over her email, continuously keep track of her location and many ways to indulge in social communications through popular apps such as Facebook, Whatsapp, Instagram and Twitter.

All these apps store privacy sensitive data about the user, which often become easily accessible once access to the phone is obtained. Unwanted access gained after a phone is lost, or even temporal access when not paying attention for a short period of time could have serious consequences.

Authentication techniques have traditionally been based on something a user knows (password, PIN), something she owns (keys, badges) or a combination of those two (ATM card + PIN). Certain properties make these insecure; passwords and pins are easily forgotten, but also easily guessed [4]. Keys and badges can be lost, or duplicated. Besides, requiring smartphone's users to carry an extra device for the sole purpose of authentication is not realistic. Recent updates in both Android and iOS include such biometric authentication; face-unlock on Android [5], and fingerprint unlock on iOS [6].

Biometric authentication is the process of verifying ones identity based on biometric features. The study and development of biometric authentication solutions have come a long way since it's first mention by Bertillon in 1870s [25]. Most popular features are physiological and behavioural features. Physiological characteristics are based on features of the body, e.g. fingerprints, hand geometry, iris or retina scans. Behavioural characteristics are based on behaviour, e.g. keystrokes, gait, signature placement and voice. Other biometrics use chemical features (based on events that happen in a persons body, measured by e.g. odour or temperature) and cognitive features (based on brain responses to specific stimuli, e.g. odour or sound).

Initial biometrics used information from a single source. These so-called unimodal systems had to deal with a range of problems like noisy data, spoof attacks and unacceptable high error-rates. Some of these issues can be addressed by combining multiple sources of information [7]. Due to the presence of multiple (mostly) independent features, the performance is expected to increase [8].

Using biometrics authentication for smartphone users faces two important challenges. First, users may use the phone in different situations and context (i.e. while walking, sit on a chair, standing up, in the dark, etc.). Thus any realistic solution should accomodate the possibility that data acquisition may fail or that a particular feature might be temporarily unavailable. Second, the solution must require as small effort as possible to users. Studies suggest that usability issues are a major driver of users' adoption decisions [9]. A recent study [33] reports that 70% users do not use any PIN/passwords to protect mobile phones because these are more annoying to users compared to other telephony related problems such as lack of coverage or low voice quality.

To partially address these challenges this paper presents a novel multimodal biometric system for smartphone users authentication. The system uses slide-unlock features, pickup movements and voice features while placing or answering a call. Being multi-modal the solution aim at robustness, such that users can still be authenticated even if some of the modalities fail.

To address the problem of usability, our authentication scheme requires zero effort to users. To the best of our knowledge this is the first authentication solution for smartphone that is completely unobtrusive. Users are not required to perform any action for the sole purpose of authentication. In fact, entering a password or PIN is more noticeable. Last but not least, our system can be implemented on most of the smartphones available on the market today.

The rest of this paper is organized as follows: Section 2 discusses related work, Section 3 describes the background knowledge. Section 4 presents the solution and the validation methology. Section 5 describes how we configured parameters in the models we used. Section 6 and Section 7 present and discuss the results of our approach. The paper is concluded in Section 8.

2 Related Work

This section reports related work that specifically take mobile devices into consideration. A wider survey of biometric authentication in general can be found in [10] and [11].

2.1 Unimodal Systems

In [12], Frank et al. consider touch operations for continuous authentication where a single type of operations are used (strokes or slides). An Equal Error Rate (EER) of 13% has been reported for one single stroke, and 2% to 3% for 11 subsequent strokes. In [13], a user is authenticated not only on the password pattern they input, but also the way they perform that input. A lab study and a long-term study provide evidence that it is possible to distinguish users and to improve the security of password patterns and even simple screen unlocks. The accuracy rate of the simple unlock is 57% at best (two-finger vertical unlock), while the accuracy of the password patterns is around 77%. In [14], Angulo et al. explored the same approach for improving password-patterns with biometrics. Using a Random Forest classifier an EER of approximately 10.4% is achieved.
Sae-Bae et al. [15] present a multi-touch gesture-based authentication technique. A classifier that uses pattern recognition techniques classifies movements characteristics of the center of the palm and fingertips. An average EER of 10% with single gestures was achieved, with improvements up to 5% EER when combining multiple gestures in a sequence.
In [16] Derawi et al. authenticate users based on gait recognition using accelerometers available in any modern mobile device. Using a low end phone (the Google G1 phone containing the AK8976A embedded accelerometer sensor) an EER of 20% is reached.

Tao et al. [17] implement a fast face detection and registration method based on a Viola-Jones detector [18]. A face-authentication method based on subspace metrics is developed. Experiments using a standard mobile camera showed that the method is effective with an EER of 1.2%.

2.2 Multimodal Systems

In [19], Saevanee et al. used SMS texting activities and messages in a multimodal authentication system. Keystroke dynamics and linguistic profiling was used to discriminate users with error rates of 20%, 20% and 22%, respectively. A fusion of these three led to an overall EER of 8%.
Buriro et al. [20] presented a sensor-enhanced touchstroke based smartphone authentication. Their study makes use of two human behaviors, i.e., how a person holds her phone and how she types her 4-digit *free text* PIN. Using Bayesian classifier and Random Rorest classifier, they achieved 1% EER.
Aronowitz et al. [21] introduced a new biometric modality called chirography which is based on user's writing on multi-touch screens using their fingers. By fusing this with face and voice features, an EER of 0.1% is reached in an office environment, and 0.5% in noisy environments.
In [22] Ferrer et al. introduced a multimodal biometric identification system that is based on the combination of geometrical, palm and fingerprint features of the users' hand.
In [23] a multimodal authentication approach is presented by Kim et al., using teeth and voice data acquired using mobile devices. The individual matching scores obtained from these biometric traits are combined using a weighted-summation operation. An EER of 2.13% was reported.
In [24], McCool et al. introduced a fully automatic bi-modal face and speaker system. A Nokia N900 was used during tests and EER results of 13.3% and 11.9% for female and male trials respectively have been reported for the fused score. This is a 25% performance improvement for the female trials, and 35% improvement for male trials.

3 Background

In this section we explain the technology and building blocks we used to build our solution.

3.1 Considered Sensors

We considered three built-in smartphone sensors, namely, accelerometer, orientation and gyroscope. The way in which each of these sensors work is explained below:

The *acceleration* (acc_n) is the acceleration applied to the device, including the force of gravity, measured on three axis' x, y and z. Android's sensor API uses a standard three-axis coordinate system. This system is defined relative to the

device's screen when it is held upright as shown in Figure 1 (a). The acceleration that is applied to a device A_d is calculated using the forces (including gravity g) that are applied to the sensor F_s itself using the following equation:

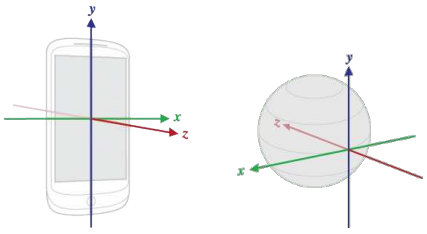
$$A_d = -g \sum \frac{F_s}{mass} \tag{1}$$

The *gyroscope* ($gyro_n$) measures the rate of rotation in radians per second (rad/s) around all axis'. The same coordinate system as described above is used.

The *orientation* (rot_n) is the rotation around the x- (pitch), y- (roll) and z-axis (azimuth) in radians (rad). Note that the orientation uses a different coordinate system than the accelerometer and the gyroscope[5]

- X is defined as the vector product $Y \cdot Z$ (it's tangential to the ground at the device's current location and roughly points West).
- Y is tangential to the ground at the device's current location and points towards the magnetic North Pole.
- Z points towards the center of the Earth and is perpendicular to the ground.

See Figure 1 for a graphical representation.



(a) (b)

Fig. 1: (a) Coordinate system relative to the device. [Source: Android SensorEvent] (b) Coordinate system used in Android's orientation sensor. [Source: Android SensorManager]

3.2 Considered Classifiers

Classification is a way of comparing an unknown query input sample with the stored templates. The Classifier/Matcher is the main component of any biometric system. The goal of the classifiers is to classify a variable $y = x_0$ called the class variable, given a set of attribute variables $\mathrm{X} = \{x_1 \ldots x_n\}$. The classifier

[5] http://developer.android.com/reference/android/hardware/SensorManager.html

$c : \mathrm{x} \rightarrow y$ is a function that maps a data instance x to a class value of y. The classifier itself it learned from a dataset D, consisting of samples over (x, y). Relating this to our scenario, the attribute variables x are the features that we extract from the touch events, motion sensors and microphone; they are the slide, pickup and voice samples. The class variable y is either *target* (meaning the instance belongs to the class learned during training) or *unknown* (meaning the instance does not appear to belong to the previously learned class).

We performed verification with three different classifiers, i.e., One-class BayesNET (BN) classifier, One-class Random Forest (RF) and One-class Sequential Minimal Optimization (SMO)-a Weka version of support vector machine (SVM). We chose these classifiers because they were shown to be very effecient in previous behavioral-based work [20,26]

We imported Weka library in our project and implemented our prototypes on smartphone.

During the training phase we only have training data available for a single instance class; the genuine user (the *target* class). At prediction time new instances with unknown class labels will have to be classified as either the *target* class or *unknown*. To handle this type of learning problem, typically called one-class classification, we wrap each classifier in a one-class classifier[6].

3.3 Performance Metric

Based on the binary outcome of this function (accept or reject), two types of errors can occur; false rejections and false acceptances. A false rejection occurs when a legitimate user is rejected access from the system and a false accept occurs when a imposter is granted access to the system. The errors are measured in the so-called False Rejection Rate (FRR) and False Acceptance Rate (FAR). These rates are calculated as follows:

$$FAR(\Delta) = \frac{FA(\Delta)}{nI} \tag{2}$$

$$FRR(\Delta) = \frac{FR(\Delta)}{nG} \tag{3}$$

Given a specific threshold Δ, the FAR is defined as the number of false acceptances (FA) divided by the number of imposters nI and the FRR is defined as the number of false rejections (FR) divided by the number of genuine users nG.

To evaluate the interaction of these error rates the Weighted Error Rate (WER) is used. The WER shows the combined error rate of both FAR and FRR with a weight α assigned to each. If the false accepts are considered worse than false rejects (focus on security), a weight > 0.5 should be used. If false rejects

[6] http://weka.sourceforge.net/packageMetaData/oneClassClassifier/

are worse than false accepts (focus on usability), than a weight < 0.5 is more appropriate. A special error rate is the EER where both errors have the same weight (i.e. $\alpha = 0.5$). The WER is defined [27] as follows:

$$WER(\alpha, \Delta) = \alpha FAR(\Delta) + (1 - \alpha) FRR(\Delta) \tag{4}$$

Given a specific weight α, the goal is to find the optimal threshold Δ^*_α for which the WER is as low as possible. This function can be defined as:

$$\Delta^*_\alpha = \underset{\Delta}{\operatorname{argmin}} \, |\alpha FAR(\Delta) + (1 - \alpha) FRR(\Delta)| \tag{5}$$

In our opinion the usability of an authentication system is of paramount importance for its adoptability. Therefore, in our system, we consider a false reject worse than a false accept, and we'll use $\alpha = 0.4$ in our evaluations.

As proposed by Poh et al. in [27], the final evaluation looks at the performance of the system after deciding on the weight α and the optimal threshold Δ^*_α. This is measured by the so called Half Total Error Rate (HTER), which is calculated as follows:

$$HTER(\Delta^*_\alpha) = \frac{FAR(\Delta^*_\alpha) + FRR(\Delta^*_\alpha)}{2} \tag{6}$$

The lower the HTER, the better the system performs given the chosen weight α.

4 Our Solution

In [28] Conti et al. introduce a new biometric measure to authenticate smartphone users; the movement a user performs when answering (or placing) a phone call. Several experiments with a prototype in a controlled environment have shown that the method is effective and that the performance is comparable to that of other transparent authentication methods, like face or voice recognition. These experiments also highlighted an issue with the data acquisition process, due to the variability in determining the end of the arm movement. To address this issue without compromising the unobtrusive nature of the initial idea we extended the solution as follows.

When placing or answering a phone call, three common steps have to be taken: 1) the user must unlock her phone, 2) bring it to her ear and 3) speak into the microphone. Our multimodal authentication solution uses features from all three steps to determine whether or not the current user is genuine, or if she is an imposter.

The complete system consists of four parts: slide movement recognition, pickup movement recognition, voice recognition and fusion. The data features are described in this section, while the next section describes the actual classification framework including fusion.

4.1 Setup

We conducted a controlled user study to test our mechanism in terms of performance and robustness. We recruited 26 participants of which 16 were male, and 3 operated their phone using their left hand. All of them were familiar with the slide-to-unlock pattern. Ages of our volunteers were ranging from 14 to 55. 2 participants were 14-19, 12 were 20-29, 7 were 30-39, 1 was 40-49 and 4 were 50 or older.

We created an Android application that targets SDK version 4.4 (*Kitkat*) and minimally requires version 4.0.3 (*Ice Cream Sandwich*). We implemented both the training phase and the classification phase using Weka 3.7 on android smartphone. The training module allows the user to anonymously record slide movements, pickup movements and voice samples which are sent to a central server. The classification module was implemented as a proof of concept and to analyze the performance on mobile phones.

A central server running on the Amazon cloud platform collected the training features in a database. A local running Java application (using Java 1.7) using the same classification module as implemented in Android was then used to test the robustness of the system. We used a Google Nexus 4 device by LG running Android 5.1 during the study. This device has a 4.7 inch screen, a Qualcomm APQ8064 Snapdragon 1.5GHz Quad-core processor, 2GB RAM, an accelerometer, gyroscope and proximity meter.

In each session, we first explained the purpose of the study to the participant and asked them if we could use their data anonymously, and noted their age and gender. After that we moved to the actual trials. Each user was required to collect at least 20 slide samples, 20 pickup samples and 10 voice samples. Samples that where distorted in any way could be removed by the user.

For the slide and pickup movements we instructed the participant to first do five movements while sitting or standing still and after that five while walking around. Then the user was asked to open a news app and read the fifteenth headline, which required the user to count while scrolling to the headlines. This usually confused users, and many had to recount from the top because they tried to wrap their head around the purpose of this task, and lost count. The goal of this distraction task was to minimize the learning effect that can occur when doing the same movement many times in quick succession. After the user read the article, she was again requested to record five movements while sitting, and five movements while walking.

4.2 Data Collection

We use the default Android slide lock as depicted in Figure 2. The center knob can be dragged towards any direction. When the user drags the knob and then release it at least as far as the circular boundary (slightly visible in the right image in Figure 2), the phone will be unlocked. If the knob is released before reaching the boundary, the phone stays locked.
During the training phase a pickup event starts when the user clicks the start

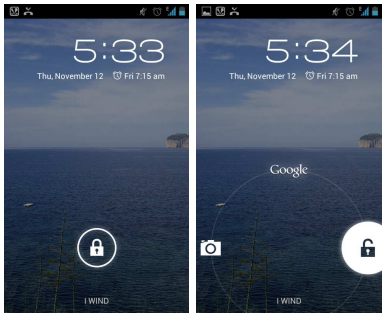


Fig. 2: Android slide lock. On the left the default state, on the right the state when a user drags the knob towards the circular boundary.

button, and ends automatically when the phone is at the user's ear (detected by the proximity detector). When used in combination with the other two modalities (e.g. during authentication), the sample starts when the slide unlock ends, and also finishes when the phone reaches the user's ear.
The Android system continuously delivers SensorEvents [7] to an event listener. As we use three sensor (accelerometer, gyroscope, orientation sensor), a delivered event can be produced by one of the sensors. Every time we receive a new event for any of the sensors, we extract the x, y and z values, and store them.

For the voice sample recording we requested the user to simply speak into the microphone as if they were answering a phone call, but to make sure to use a relatively lengthy sentence to fill the 2.5 seconds of recording. Most users used a sentence similar to *Hello, this is John Doe. Who am I speaking to?*. An audio sample is recorded for 2500 ms at a sample rate of 8 kHz using 16 bits per sample with one channel. The resulting pulse-code modulation (PCM) data is stored in a temporary WAV file on the device.

4.3 Feature Extraction

Slide A slide sample starts when the user touches the knob for the first time, and ends when the knob is released (e.g. the user stops the touch event). One slide is a path encoded as a sequence of vectors $(t_n, x_n, y_n, p_n, s_n)$. Only complete samples (samples that would unlock the phone in the original non-biometric implementation) are considered, others are simply discarded.

During the slide event the features in Table 1 are recorded at a average sampling rate of 150 Hz. From the given MotionEvent we extract multiple features. The *time offset* (t_n) indicates the offset since the start of the touch event in milliseconds.

The *x- and y-position* (x_n, y_n) are measured in pixels and indicate the exact position of the knob (controlled by the users touch) on the screen. Over time

[7] http://developer.android.com/reference/android/hardware/SensorEvent.html

Table 1: Slide features

Feature	Unit
Time offset	ms
X-position	px
Y-position	px
Pressure	Normalized value between 0 and 1
Size	Normalized value between 0 and 1

these coordinates create a path from the initial position of the knob towards the boundary of the circle, indicating exactly how the user moved the knob.

The *touch pressure* (p_n) of the touch event indicates the approximate pressure applied to the surface of the screen. The value is normalized to a range from 0 (no pressure at all) to 1 (normal pressure), but values higher than 1 may be generated depending on the calibration of the input device.

The *size* (s_n) is a scaled value of the approximate size of the area of the screen being touched. The actual value in pixels corresponding to the touch is normalized with the device specific range of values and scaled to a value between 0 and 1.

Pickup During the pickup event the features in Table 2 are extracted at a average sampling rate of 190 Hz. The *time offset* (t_n) indicates the offset since the start of the pickup event in milliseconds. One pickup movement is encoded as a sequence of vectors $(acc_n^x, acc_n^y, acc_n^z, gyro_n^x, gyro_n^y, gyro_n^z, rot_n^x, rot_n^y, rot_n^z, t_n)$.

Table 2: Pickup features

Features				Units
1-3	X-acceleration	Y-acceleration	Z-acceleration	$^m/_{s^2}$
4-6	X-gyroscope	Y-gyroscope	Z-gyroscope	$^{rad}/_s$
7-9	X-orientation	Y-Orientation	Z-Orientation	rad
10	Time offset			ms

Voice Using the recorded voice sample, we calculate the Mel-frequency cepstral coefficients (MFCCs) [29] and store them in a feature vector. MFCCs have been very popular in the realm of speech recognition due to its ability to represent the speech amplitude spectrum in a compact form [30]. Creating MFCCs is done by 1) converting the waveform to frames, 2) take the discrete Fourier transform, 3) take the Log of the amplitude spectrum, 4) Mel-scaling and smoothing and 5)

applying discrete cosine transform. The MFCC features are then used as data instances that we use to create models for our classifiers

4.4 Data Fusion

In our multimodal mechanism we use multiple biometric traits (slide movement, pickup movement and voice) which need to be fused to output one single decision: accept or reject. We fused these modalities at match-score level. However, because each modality performed differently, we give each modality a weight, based on it's unimodal performance.

Consider three modalities x, y and z, having an error rate (er) of 0.1, 0.2 and 0.3 respectively. Obviously, modality x is much better than y and z, and should therefore have a higher weight. For each classifier c we can calculate a success index. The success index indicates how much the classifier contributes to the sum $1 - er(c)$ for each classifier c.

$$index(c) = 1 - \frac{er(c)}{\sum_{i=1}^{n} er(i)} \tag{7}$$

The eventual weight can then be calculated using:

$$weight(c) = \frac{index(c)}{\sum_{i=1}^{n} index(i)} \tag{8}$$

Filling in the values for three modalities x, y and z, they would get weights of 0.42, 0.33 and 0.25 respectively. Better modalities get get higher weights.

4.5 Decision Making

To measure the performance of the classifiers we use the cross-validation method. The dataset is randomized and then split into k folds of equal size. In each iteration, one fold is used for testing, and the other $k - 1$ folds are used for training the classifier. We use $k = n$, meaning we apply leave-one-out cross-validation. The test results are averaged over all folds, which give the cross-validation estimate of the accuracy. This method is useful because we are dealing with small datasets and the idea is to test each sample. Using cross-validation we utilize the greatest amount of training data from the dataset.

When evaluating the performance of a biometric system, multiple criteria should be considered [27]. Biometric authentication systems make decisions based on the following decision function:

$$f(x) = \begin{cases} accept, & \text{if } c(I, \mathrm{x}) \geq \Delta \\ reject, & \text{otherwise} \end{cases} \tag{9}$$

where $c(I, f)$ is the output of the underlying classifier c that indicates how certain it is that the claimed identity I is correct based on the given dataset (features) x. The threshold Δ defines when an identity claim is accepted or rejected. Access to the system is accepted if the score is greater than or equal to the threshold, and rejected otherwise.

5 Parameters

Before we can show any results, we first need to identify the exact data and models under test. During the research we did many extensive tests to find the optimal setup. These tests led us to the best performing combination of parameters. The actual performance of the best classifier will be discussed and evaluated in the next section.

The tests have been carried out on a random subset (length: 10) of the participants in the user test. For each configuration considered we calculated the equal error rate (EER) based on all samples of the genuine user, and 10 random samples per other (non-genuine) user.

5.1 Parameters

For each modality we use all attributes, and do a grid search to find the best performing set of parameters. We also record the average model generation time so we can filter out configurations that would take too long on mobile phones.

Table 3: Best classifier per modality.

	Slide		Pickup		Voice	
Classifier	Comp. Time	EER	Comp. Time	EER	Comp. Time	EER
BayesNET	64	0.1242	762	0.2045	205	0.2681
Random Forest	4453	0.1434	13988	0.2083	4402	0.2452
SMO	8433	0.1864	~144000	-	548	0.2709

Table 3 gives for each modality an overview of the best performing classifiers. The parameter tests show that the Bayesian network classifiers yield the best results overall. Only with the voice modality the Random Forest classifier yields slightly better results. However, the Bayesian network is much faster.

From this point on when talking about the classifier, we mean the Bayesian classifier, with its parameters configured as shown in Table 4.

6 Results

The results we present here are based on the user data we collected during the controlled users tests, fed into the classifiers with their parameters configured as

Table 4: Parameter configuration per modality

Modality	Naive Bayes	Markov blanket	Max parents	Score type	Alpha
Slide	T	T	5	Entropy	0.25
Pickup	T	F	3	Bayes	0
Voice	F	F	5	Entropy	0

described in Section 5. For each user this gives us a certainty number (higher means more similar to the reference model) for both genuine and impostor samples.
It is important to note that when testing a classifier for user u, we use all samples from all other users to do our impostor tests. By doing so, we have much more impostor samples than genuine samples, leaving the FRR much more sensitive to deviations than the FAR.

Given the data from the user we can find the optimal threshold Δ_α^*. The optimal threshold is the threshold for which the Weighted Error Rate (WER) is at its minimum (see Equation 5).

6.1 Unimodal Systems

Slide Given $\alpha = 0.4$, we found that the optimal threshold $\Delta_\alpha^* = 49$. Re-running the tests with this threshold gives us a FAR of 22.28% and a FRR of 4.84%.

The HTER (defined in Equation 6) can now easily be computed:

$$HTER(49) = \frac{22.28\% + 4.84\%}{2} = \textit{13.56\%} \tag{10}$$

Pickup The optimal threshold $\Delta_\alpha^* = 42$. Running the tests with this threshold gives us a FAR of 26.69% and a FRR of 6.19%.

$$HTER(42) = \frac{26.69\% + 6.19\%}{2} = \textit{16.44\%} \tag{11}$$

Voice The optimal threshold $\Delta_\alpha^* = 85$. Running the tests with this threshold gives us a FAR of 63.92% and a FRR of 12.69%.

$$HTER(85) = \frac{63.92\% + 12.69\%}{2} = \textit{38.30\%} \tag{12}$$

It is evident that slide and pickup modalities are better than voice modality. Still, we are using it here to show how the use of multimodal biometric authentication can improve a unimodal authentication system.

6.2 Multimodal Systems

Slide+Pickup Modalities As described in Section 4.4 we use a match-score level fusion method, using weights for each classifier output. We calculate the weight using Equation 8. In the previous subsection we have seen that the slide and pickup classifiers have a HTER of 13.56% and 16.44% respectively. Filling in the equation this gives us a weight of 0.55 for slide and 0.45 for pickup.

The optimal threshold $\Delta_\alpha^* = 55$. Re-running the tests with this threshold gives us a FAR of 11.01% and a FRR of 4.12%. Calculating the HTER gives us:

$$HTER(55) = \frac{11.01\% + 4.12\%}{2} = \textit{7.57\%} \tag{13}$$

Comparing the slide and pickup modalities individually with this multimodal system, we can see that the latter performs almost twice as good.

Slide+Pickup+Voice Modalities We have seen that the slide, pickup and voice classifiers have HTERs of 13.56%, 16.44% and 38.30% respectively. Using Equation 8 this results in weights 0.40 (slide), 0.38 (pickup) and 0.22 (voice).

The optimal threshold $\Delta_\alpha^* = 62$. Running the tests with this threshold gives us a FAR of 10.72% and a FRR of 3.93%. Calculating the HTER gives us:

$$HTER(62) = \frac{10.28\% + 3.93\%}{2} = \textit{7.33\%} \tag{14}$$

A quick comparison shows that adding voice modality to the multimodal system using slide and pickup features does not improve the results significantly but still better (HTER 7.33% vs HTER 7.57%).

7 Discussion

The results show that the slide modality is better than the pickup modality. The main cause for this observation is that the pickup classifier is much more sensitive to the kind of activity the user performs while unlocking her phone. Because the rotation, gyroscope and acceleration of the device are the main features of the modality, the user's activity while unlocking has major influence on the classifier outcome: walking, running, standing in a crowded bus; they all have different impact on the motion sensors of the device.

The slide modality on the other hand does not use motion sensors but rather uses touchscreen. Touchscreen determines finger position, pressure and size on a screen which are much less influenced by external factors, making the modality more robust in a range of different scenario's.

When combining the slide and pickup modalities, we can see that the FAR improves significantly.

The voice modality is obviously not good enough (based on our experiments) and may not be deployed in real world because of higher error rates - FAR of 63.92% and FRR of 12.69%. The reason(s) for worst voice results may be due to the low quality of the open source library and by the fact we applied only basic clustering mechanisms. Still, the fusion of all three modalities yielded better results.

System like ours are suitable for risk-based authentication scenarios (e.g. mobile banking applications), where security may need to be traded for availability dynamically and adaptivelly.

8 Conclusion and Future Work

In this paper we proposed a new multimodal biometric system for smartphone user authentication that focusses on usability. The system uses features collected during a slide-unlock movement on a smartphone. We use finger position, pressure, size and time offset to generate a model and classify future slide movements. We shown how fusion of unimodal systems to multimodal ones using slide, pickup and voice modalities can significantly improve performance.

We have applied three different classifiers, i.e., BN, RF and SVM. BN classifier outperformed the other classifiers in terms of error rates and computation time.

From the three unimodal traits we tested (slide, pickup and voice); the slide modality performed best with a FAR of 22.28% and a FRR of 4.84%, resulting in a HTER of 13.56%. The pickup modality performed slightly worse, with an FAR and FRR of 26.69% and 6.19% respectively, and an HTER of 16.44%. However, with their fusion, we were able to achieve much improved performance (by a factor of two). A FAR of 11.01% and a FRR of 4.12% were reached, resulting in a HTER of 7.57%.

The voice based model performed much worse as the open source library we used was simply not good enough. However, we have shown the potential improvement of a multimodal system using slide, pickup and voice modalities.

This research can be extended in multiple directions. To validate the results presented here a larger user study should be conducted. The impact situations, context and environment may have on this type of biometrics need to be investigated further.

9 Acknowledgement

Authors would like to thank all the volunteers, who participated in this experiment for their valuable feedback and comments .

This work has been partially supported by the TENACE PRIN Project (n. 20103P34XC) funded by the Italian MIUR and EIT Digital MobileShield project.

References

1. Dan Morrill, Announcing the Android 1.0 SDK, release 1, Google, 2008, http://android-developers.blogspot.in/2008/09/announcing-android-10-sdk-release-1.html, accessed (20-04-2015)
2. Peter Cohen, Macworld Expo Keynote Live Update, PCWorld, 2007, http://www.macworld.com/article/1054764/liveupdate.html, accessed (20-04-2015)
3. , IDC: Android and iOS Squeeze the Competition, Swelling to 96.3% of the Smartphone Operating System Market for Both 4Q14 and CY14, According to IDC, IDC, http://www.idc.com/getdoc.jsp?containerId=prUS25450615, (2015), accessed (20-04-2015).
4. Wood, Helen M: The use of passwords for controlled access to computer resources, US Department of Commerce, National Bureau of Standards, Vol-500, number-9, (1977)
5. Google: Ice Cream Sandwich, http://developer.android.com/about/versions/android-4.0-highlights.html, (2011), accessed (20-04-2015).
6. Chris Velazco: Apples Touch ID Is A 500ppi Fingerprint Sensor Built Into The iPhone 5S Home Button, http://techcrunch.com/2013/09/10/apples-touch-id-a-500ppi-fingerprint-sensor, TechCrunch (2013), accessed (20-04-2015).
7. Ross, Arun and Jain, Anil: Information fusion in biometrics, Pattern recognition letters, Elsevier, Vol-24, number-13, 2115–2125 (2003)
8. Kuncheva, Ludmila I and Whitaker, Christopher J and Shipp, Catherine A and Duin, Robert PW: Is independence good for combining classifiers?, In proceedings of 15th International Conference on Pattern Recognition, IEEE, Vol-2, 168–171, (2000)
9. Bhagavatula, Chandrasekhar and Ur, Blase and Iacovino, Kevin and Kywe, Su Mon and Cranor, Lorrie Faith and Savvides, Marios: Biometric Authentication on iPhone and Android: Usability, Perceptions, and Influences on Adoption, (2015)
10. Jain, Anil K and Flynn, Patrick and Ross, Arun A: Handbook of biometrics, (2007)
11. Yampolskiy, Roman V and Govindaraju, Venu: Behavioural biometrics: a survey and classification, International Journal of Biometrics, Inderscience, Vol-1, number-1, 81–113, (2008)
12. Frank, Mario and Biedert, Ralf and Ma, Eugene and Martinovic, Ivan and Song, Dawn: Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication, IEEE Transactions on Information Forensics and Security, IEEE, Vol-8, number-1, 136–148, (2013)
13. De Luca, Alexander and Hang, Alina and Brudy, Frederik and Lindner, Christian and Hussmann, Heinrich: Touch me once and i know it's you!: implicit authentication based on touch screen patterns, In proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, Vol-8, number-1, 987–996, (2012)
14. Angulo, Julio and Wästlund, Erik: Exploring touch-screen biometrics for user identification on smart phones, Privacy and Identity Management for Life, Springer, 130–143, (2012)
15. Sae-Bae, Napa and Ahmed, Kowsar and Isbister, Katherine and Memon, Nasir: Biometric-rich gestures: a novel approach to authentication on multi-touch devices, In proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 977–986, (2012)
16. Derawi, Mohammad Omar and Nickel, Claudia and Bours, Patrick and Busch, Christoph: Unobtrusive user-authentication on mobile phones using biometric gait

recognition, Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), IEEE, 306–311, (2010)
17. Tao, Qian and Veldhuis, R: Biometric authentication for a mobile personal device, 3rd Annual International Conference on Mobile and Ubiquitous Systems-Workshops, IEEE, 1–3, (2006)
18. Viola, Paul and Jones, Michael: Robust real-time object detection, International Journal of Computer Vision, vol-4, 34–47, (2001)
19. Saevanee, Hataichanok and Clarke, Nathan L and Furnell, Steven M: Multi-modal behavioural biometric authentication for mobile devices, Information Security and Privacy Research, vol-4, 465–474, (2012)
20. Buriro, A., Crispo, B., Del Frari, F., & Wrona, K: Touchstroke: Smartphone User Authentication Based on Touch-Typing Biometrics, In New Trends in Image Analysis and Processing (ICIAP 15) Workshops, Springer International Publishing, 27-34, (2015)
21. Aronowitz, Hagai and Li, Min and Toledo-Ronen, Orith and Harary, Sivan and Geva, Amir and Ben-David, Shay and Rendel, Asaf and Hoory, Ron and Ratha, Nalini and Pankanti, Sharath and others: Multi-modal biometrics for mobile authentication, IEEE International Joint Conference on Biometrics (IJCB), IEEE, 1–8, (2014)
22. Ferrer, Miguel A and Morales, Aythami and Travieso, Carlos M and Alonso, Jesws B: Low cost multimodal biometric identification system based on hand geometry, palm and finger print texture, 41st Annual IEEE International Carnahan Conference on Security Technology, IEEE, 52–58, (2007)
23. Kim, Dong-Ju and Hong, Kwang-Seok: Multimodal biometric authentication using teeth image and voice in mobile environment, IEEE Transactions on Consumer Electronics, IEEE, vol-54, number-4, 1790–1797, (2008)
24. McCool, Christopher and Marcel, Sebastien and Hadid, Abdenour and Pietikainen, Matti and Matejka, Pavel and Cernocky, Jan and Poh, Norman and Kittler, Josef and Larcher, Anthony and Levy, Christophe and others: Bi-modal person recognition on a mobile phone: using mobile phone data, IEEE International Conference on Multimedia and Expo Workshops (ICMEW), IEEE, 635–640, (2012)
25. Bertillon, Alphonse: Signaletic instructions including the theory and practice of anthropometrical identification, Werner Company, (2008)
26. Sitova, Zdenka, Jaroslav Sedenka, Qing Yang, Ge Peng, Gang Zhou, Paolo Gasti, and Kiran Balagani: HMOG: A New Biometric Modality for Continuous Authentication of Smartphone Users, arXiv preprint arXiv:1501.01199 (2015).
27. Poh, Norman and Bengio, Samy: Database, protocols and tools for evaluating score-level fusion algorithms in biometric authentication, A review, Pattern Recognition, Elsevier, vol-39, number-2, 223–233, (2006)
28. Conti, Mauro and Zachia-Zlatea, Irina and Crispo, Bruno: Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call, In proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, Science and Technology, ACM, 249–259, (2011)
29. Davis, Steven and Mermelstein, Paul: Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, IEEE Transactions on Acoustics, Speech and Signal Processing, IEEE, vol-28, number-4, 357–366, (1980)
30. Logan, Beth and others: Mel Frequency Cepstral Coefficients for Music Modeling, ISMIR, (2000)
31. Hsu, Chih-Wei and Chang, Chih-Chung and Lin, Chih-Jen and others: A practical guide to support vector classification, 2003

32. Oshiro, Thais Mayumi and Perez, Pedro Santoro and Baranauskas, José Augusto: How many trees in a random forest?, Springer, 154-168, (2003)
33. Survey says 70% password dont protect mobiles: download free Mobile Toolkit, http://nakedsecurity.sophos.com/2011/08/09/free-sophos-mobilesecurity-toolkit, 2014