Dinosaurs are extinct - Passwords are not: A Usability Evaluation of the password manager software KeePass2

IPP

MSc Project Proposal

Harris Flourentzos - s1687849

THE UNIVERSITY *of* EDINBURGH
informatics

Master of Science
School of Informatics
University of Edinburgh
2018

Contents

1 Introduction

Although many of us don't yet realize and usually refer to as science fiction, our digital selves are already well established and alive in the world wide web. Scattered across several social media profiles, bank accounts, web browser histories and cookies, to name a few, they are undoubtedly an ever-increasing part of our identities and everyday lives. Yet, despite their numerous security and usability flaws, the only measure that ensures the privacy and security of our digital selves still remains the primitive tool we all know and hate, the password.

Password-Authentication successful usage, as is the case with many security systems, depends on two main aspects; A mathematical and a user-based aspect. The mathematical one is a problem long solved through the study of cryptography and assuming that it is implemented correctly by the software engineer of a security system, it will never fail. The user-based aspect unfortunately, has been shown to be the weakest link by several studies in the last few decades. The main reason of failure in this aspect is that it asks and depends up on the user to implement it correctly. To do so, the user must follow an increasingly difficult to manage list of rules, called password security policies, which try to guide them in creating and using passwords correctly. This is a process that burdens the user with mental effort for which he/she has limited reserves of [1][2], and so it is often ignored by a large portion of the user population. This is not surprising since users treat all security related tasks as secondary tasks that usually get in the way and interrupt the process of completing their primary objectives as shown in [3].

As a result, users usually tend to ignore security policies and security advice and choose passwords with considerably less than ideal strength, reuse passwords across different accounts, write passwords down [4].

Due to this fact, users are considered to be the weakest link in the security chain, and most of the attacks try to exploit this weakness in a variety of different approaches. Two main types of attacks are brute force attacks that try to exploit the way users choose passwords (for example choosing the name a favourite actor) and social engineering attacks that try to exploit the way that users manage their passwords (for example phishing, key-logging, etc).

To alleviate users of this effortful practise, two main approaches were suggested in the last few decades. One was education of users to use correct security policies and the other to design usable systems that depend on users' efforts as less as possible. The former approach is dismissed (shown to be problematic) by many [1] [5] [2] since it involves ever increasing lists of rules and policies and requires users to remember and update their knowledge constantly. The latter approach suggests replacing passwords with other authentication mechanisms or at least systems that manage passwords in a more user convenient and friendly manner. This has been proven to be an equally troublesome task, since simply none of the suggested alternatives, be it password managers, dual device authentication, biometrics, visual cryptography to name a few, provide complete superiority to password usage from a usability and security perspective [6].

One technique that seems to at least mediate the burden placed on users are password managers. Although not perfect, it is thought that with the correct usability focused design of a password manager, a reasonable security standard could be achieved while keeping the effort invested by users to the minimum. In a recent study [7] to outline the most important security advice to end-users, *"Use a password manager"* was one of the most common advice given by experts. Therefore, the focus of this study is to evaluate and re-design an existing password manager software called KeePass2 [https://keepass.info/].

In the following sections we expand on topics mentioned in the introduction and present an outline of the study we plan to curry out:

Section 2 will identify and clearly state the purpose of this project.

Section 3 will present background work that has been achieved so far. This will include a basic description of the KeePass2 software as well as previous studies conducted that either aimed to improve or build from scratch other password manager systems.

Section 4 outlines the methodologies we will use to evaluate the original KeePass2 and the early designs of the revised version.

Section 5 will present how we plan to evaluate the final revised version of KeePass2.

Section 6 will discuss expected outcomes of the study as well as back up plans in case of unexpected events.

Finally, Section 7 will present a timeline plan of the study.

2 Purpose

2.1 What is the problem

The main problem that this study is trying to address is that security software engineers take users' security knowledge for granted and design security systems with little usability that can only be effective if users apply increasingly long and complicated security policies.

2.2 Why does the problem exist

Research [5][1] suggests that the responsibility should be shifted from the users to the designers of such security systems. Software engineers should design systems that abstract as many security policies as possible from the user to remember and understand. Make sure that the mental model that their system imposes on the user is the correct one. And finally make the use of the system as effortless as possible.

The reason that this shift is necessary, is that security is not the primary task users are trying to accomplish. On the contrary security tasks are just secondary tasks that usually get in the way and disrupt the completion of the users' main objective. If the information they are asked to remember and the amount of effort and time they need to invest into the security task is well beyond their limited mental reservoirs they will ignore the advice. In the concept of passwords this involves creating and remembering difficult and numerous passwords, identifying potential social engineering threats etc.

To provide solution to the problem at hand, we will evaluate a password manager system, identify in this way key points that the engineers have missed and try to remedy those points.

2.3 Why is the study important

The importance of the study will be twofold:

- It will uncover any security risks the users are exposed to by using KeePass2 due to usability issues. Remedy those issues by designing a revised version of KeePass2 providing users with a free, usable and secure system that they can employ and continue to use in the intended way.

- Provide guidance to future security software engineers as to what to consider while creating new security software.

3 Background

3.1 Related Work

There has been a considerable volume of previous work that tries to identify usability design principles that intend to create usable, secure systems for end-users. As suggested in many of these studies, usability principles should be extended beyond the ones commonly used for generic user interface design and evaluation in order to incorporate the extra difficulties [8] that security introduces. These extra difficulties are encapsulated in a series of key points suggested in [8] (see appendix A.1).

More specifically some studies [9][10] have focused on password managers to provide solution to the problem. [9] have tried to design a password manager, called Tapas that uses dual device authentication instead of a master password to encrypt the database. The goal of the study was to alleviate users even with the task of creating and remembering a master password erasing completely the burden of memorability. This innovation came with no small price though since the user was burden with needing both a laptop and a mobile device. [10] conducted a usability evaluation in 2 password manager systems PwdHash and Password Multiplier. They discovered that the main reason for security exposures was that users had inaccurate or incomplete mental models of the security system. Furthermore they found that users were reluctant to trust a password manager with their passwords.

It should be noted that, to the knowledge of the writer, there has been no official usability study on KeePass2.

3.2 Passwords

3.2.1 Password Strength

Password strength is measured either in terms of the number of guesses an attacker will need to succeed in a brute force attack or in terms of the base-2 log of the number of guesses, called the entropy. Entropy is measured in bits and it is a concept from Information Theory. A password with, say, 42 bits of strength calculated in this way would be as strong as a string of 42 bits chosen randomly. Put another way, a password with 42 bits of strength would require 2^{42} (4,398,046,511,104) attempts to exhaust all possibilities during a brute force search. Thus, adding one bit of entropy to a password doubles the number of guesses required, which makes an attacker's task twice as difficult.

3.2.2 Password Storage

Hash Functions:
Storing passwords, for example in online databases or in password managers, should not be done without encryption. To do so, a **cryptographic hash function** is used that takes as input a message of arbitrary length and outputs a fixed-length string. Any changes in the input message will result in a different output string. A hash function has also the following properties:

- It is a one-way function, that is, a function which is easy to compute but hard to invert. This means that if an attacker has the hash value of a message, it should be very difficult to retrieve the original message from it.
- It is collision resistant, meaning that it is very difficult to find two different messages that produce the same hash value.

Since an attacker should not be able to reverse hash values to obtain the original messages, this is more secure than storing passwords in plaintext. Choosing a strong hash function should be of highest priority to the software engineer so as to avoid password leaks like the famous linkedin leak [11] with the use of SHA1 as a hash function.

Salting:
In cryptography, a salt is random data that is appended to a plain text password right before the hashing takes place. The primary function of salts is to defend against dictionary attacks or against its hashed equivalent, a pre-computed rainbow table attack.

A new salt is randomly generated for each password. In a typical setting, the salt and the password are concatenated and processed with a cryptographic hash function, and the resulting output is stored with the salt in a database. Hashing allows for later authentication without keeping and therefore risking the plaintext password in the event that the authentication data store is compromised.

Since salts do not have to be memorized by users they can make the size of the rainbow table required for a successful attack prohibitively large without placing a burden on the users. Since salts are different in each case, they also protect commonly used passwords, or those who use the same password on several sites, by making all salted hash instances for the same password different from each other.

3.2.3 Password User Habits

In a large study on end-users' password habits [4] involving more than half a million users, it was discovered that an average web user has more than 25 online accounts on average, with 6.5 passwords shared across 3.9 different websites. Furthermore, the average bit-strength of all user chosen passwords was 40.53 bits. The majority of users also prefer to select longer lower-case letter passwords and rarely use upper-case letters and special characters. It was also pointed out that users tend to forget their passwords very often (4.28% of Yahoo users forgot their passwords over a three month period.) and some fall victim to online phishing attacks (0.4% of the total user population per year).

As mentioned previously, a common policy adapted from many online organizations is to provide password meters and restrictive policies (for example not allowing users to pick passwords that are included in common dictionary attack lists) in an attempt to guide users to select stronger passwords. In a study [12] that focused on the effects of such visual meters and restrictive policies identified that, although there was significant difference in the behaviour of users in the presence and absence of visual meters (visual meters did encourage users to produce stronger passwords) the increase of password strength was not enough to make a difference against offline brute-force attacks. Furthermore, they discovered that stringent policies (for example use of all ASCII character types) employed by some organizations did have a significant effect on choosing stronger passwords with resistance to brute-forced attacks, but had the result in frustrating users.

3.2.4 Attacks on Passwords

We can group attacks on Password authentication into 2 categories under the context of this study, namely direct attacks and attacks through users.

Direct attacks:
The most common types of direct attacks are:

- **Brute-force attacks** - Brute-force attacks are one of the most popular password cracking methods. They use a trial-and-error approach by systematically submitting all possible passwords combinations until the correct one is found [12]. The amount of time needed for a brute-force attack increases exponentially in relation to the length of the password.It's the simplest way to crack a password, but also the most ineffective, since it wastes a lot of time making unlikely guesses. In general, this type of attack is eliminated by choosing higher bit strength passwords and by using rate limiting techniques, but of course this depends if the attack is offline or online. (see more details in offline and online attacks sections)
- **Dictionary Attacks** - A dictionary attack is performed by running through a list of potential passwords until the correct one is found. Dictionary attacks rely on the fact that users often choose common passwords (or variations thereof - by appending a special character or digit to the beginning or the end of the password), and therefore enable attackers to search through the most likely options. Dictionary attacks are relatively easy to defeat, e.g. by using a pass-phrase or otherwise choosing a password that is not a simple variant of a word found in any dictionary or listing of commonly used passwords.
- **Rainbow-Table Attacks** - A rainbow table is essentially a dictionary which stores a list of common passwords alongside their hashes for a given hashing algorithm. It is optimized for hashes and passwords so maintains a fast look-up speed, despite the space it takes.However, the table itself will be huge and require some serious computing power to run, and it's useless if the hash it is trying to find has been 'salted' by adding random characters to the password before applying the hashing algorithm.

Furthermore, direct attacks can be divided into two different categories depending on whether the attack is done online or offline:

- **Offline attacks** take place in the case of a leaked or stolen hashed database. These are brute force attacks where the attacker tries to guess the master password of the encrypted database through his own hardware and software resources. As technology progresses and GPUs become more powerful so is the success rates of these kind of attacks. The main defence against these attacks is the bit strength of the chosen master password. Other techniques that have appeared to increase resistance is hash iterations.
- **Online attacks** are also brute force guessing attacks but differ in the fact that the hashed database is not leaked. Attackers launch guessing attacks by trying different passwords the same way that a user tries to log in to his/her online account. In this case attackers are limited to how fast they can iterate through passwords and thus it is easier to guard against these. The determining factor again is the bit strength of the password, but since the attackers computational resources are limited weaker passwords can be acceptable. Typical defence mechanisms to further combat online attacks are password blacklisting and throttling.

Attacks through Users:
These are social engineering attacks that try to take advantage of users' unawareness of security systems. We list some of the main ones here and will provide a more comprehensive summary in the actual project.

1. **Phishing**
   Phishing is the attempt to obtain sensitive information such as usernames, passwords, and credit card details (and money), by disguising as a trustworthy entity in an electronic communication. According to the 2013 Microsoft Computing Safety Index, released in February 2014, the annual worldwide impact of phishing could be as high as US $5 billion. Phishing is typically carried out by email spoofing or instant messaging and it often directs users to enter personal information at a fake website, the look and feel of which are identical to the legitimate one and the only difference is the URL of the website in concern.

   In a study [13] to identify why phishing has been the number one most effective technique to disclose sensitive information from users, the researchers found the following main results:

   - Good phishing websites fooled 90% of participants.
   - Existing anti-phishing browsing cues are ineffective 23% of participants in the study did not look at the address bar, status bar, or the security indicators.
   - On average, participants made mistakes on the test set 40% of the time. Popup warnings about fraudulent certificates were ineffective: 15 out of 22 participants proceeded without hesitation when presented with warnings.
   - Participants proved vulnerable across the board to phishing attacks. In the study, neither education, age, sex, previous experience, nor hours of computer use showed a statistically significant correlation with vulnerability to phishing.

   Additional to the user-based results, they also summarized common phishing strategies from 200 phishing real life examples deriving 3 main dimensions:

   - Lack of knowledge
   - Visual deception
   - Bounded attention

   Due to space limitations, we include those in full in the appendix A.3 since they are crucial in designing systems that will eliminate common phishing attacks.

2. **Key-Logging**
   Keystroke logging, often referred to as keylogging or keyboard capturing, is the action of recording (logging) the keys struck on a keyboard, so that the person using the keyboard is unaware that their actions are being monitored. A keylogger can be either software or hardware. Software-based keyloggers vary from malware programs to legit software used for commercial purposes. Hardware-based keyloggers vary dramatically from firmware based, keyboard harware, wireless sniffers, acoustic, optical, electromagnetic etc.

3.3 Password Managers

Password Managers are software designed to provide some protection from the above attacks. They are implemented in different forms for example local software, browser plug-ins but they usually serve to offer users:

- Stronger Passwords – this is usually done by password meters to allow users to choose their own passwords or by automatic password creation.
- Eliminate password reuse across different profiles – This is establish by using a Master Password while hashing the stored passwords in an encrypted database.
- Alleviates users from the Memorability burden.
- Faster authentication – This is provided by auto-complete forms, copy&paste or drag&drop functions.
- Some protection against social engineering attacks – For example, phishing can be prevented by storing the correct URL of the webpage.

- Save time and effort in the long run – No forgetting passwords, No effort to come up with new passwords when asked to update an existing password

Numerous commercial solutions have been built and introduced to the public through the years, yet a few have been able to survive and somewhat become widely adopted by users. Some of them are, Dashlane, 1Password, LastPass, etc. The downside of those, is that they are not freely available to the public and their source code is not open source. We wanted to consider a software that is free, since asking a user to pay in order to accomplish a secondary task seems quite demotivating to begin with. Although it might be surprising to the reader, it has been suggested and consistently shown that revealing the implementation of a security system to the public has more chances of making the system more robust than the one that is kept in secrecy. This is simply because such a system can be validated en mass. These are the two main reasons that encouraged us to use KeePass2 in this study.

3.4 KeePass2

KeePass [https://keepass.info/] is a free open source password manager, which helps users to manage their passwords in a secure way. It allows the user to create a single database file that contains all his passwords, which is locked with one master password and/or a key file. The database can then be stored either locally or in the cloud. More specifically the main functionality of the software allows the user to:

- Create an encrypted database by choosing a Master Key, shown in figure 3.
- Populate the database with several password entries, shown in figure 2. This can be done either by migrating password credentials for an existing online profile or creating a new one. A single entry would look like | **Tittle** | **User Name** | **Password** | **URL** | **Notes** | and it will be fully encrypted; Not just the password field.
- Use the Master Key to unlock the database and access any of the stored password entries, shown in figure 4.
- Copy&Paste or Drag&Drop the appropriate field from the password entry to the appropriate website fields.

KeePass2 uses Advanced Encryption Standards (AES) and the Twofish algorithm to encrypt the database, which are considered very secured at the time of writing. SHA-256 is used to hash the master key components. SHA-256 is a 256-bit cryptographically secure one-way hash function. No attacks are known yet against SHA-256.

KeePass 1 and 2 was developed by Dominic Reichl, and the official project can run on Windows OS only. Since it is open sourced many unofficial ports exist for all main operating systems including OS X, Linux, Android and IOS. Its functionality can also be extended by the numerous plug-ins developed by 3rd party developers.

For this study we will only focus on the official software which provides the basic functionality, mainly because we want to avoid any trust related issues that users may experience due to the reference to the 3rd party software as “unofficial”.

It should be emphasised that being an open source software, is actually a positive and crucial security attribute and care should be taken through this study to inform users about this.

4 Methods

4.1 Overview of Usability Studies

In this section we provide a brief overview of what usability studies are and proceed to describe and explain particularly the methodologies that will be used to conduct this study as well as the reason for choosing each one.

In the field of Human Computer Interaction, there exist numerous methodologies in designing usable systems. More often than not, these methodologies are not strictly/formally defined, they can be combined to form new strategies and are usually grouped into families according to several different perspectives. For example, they can be grouped together according the time in the design phase they are used in – early to assess the requirements for the system we want to design and later in order to evaluate and refine it. They can also be grouped according to whether they involve users or experts and according to the data they try to capture quantitative/qualitative and behavioural/attitudinal.

In this study we will focus more on techniques and methodologies that:

- *are used in later design stages.* – The reason for this choice is simply because we already have performed (and will continue to do so through out this project) a considerable amount of research in existing literature in the specific field and have already identified basic requirements of the system. Furthermore, the system we want to evaluate has already been designed and developed.
- *use both expert [14] and user based (lab studies) techniques (though focusing more on user based).* – The reason for this choice is that both families of methodologies tend to reveal different usability flaws and complement each other in several aspects. In a nutshell, expert-based studies (heuristic evaluations, cognitive walkthroughs, etc) require a few HCI experts but no users and so they are fast and cheaper than user based. On the other hand, they are no substitute of user-based studies (think aloud, usability testing, task analysis, focus groups, etc) since usability of systems is always optimized with end-users in mind.
- *yield quantitative and behavioural data.* – We chose to elicit qualitative data because even though they can be difficult to analyze and introduce risks of subjective interpretation by experimenters, they maximizes the chances of directly revealing usability issues.

4.2 Usability Evaluation

In order to evaluate any security system in terms of usability, we firstly need to define what usable means. Defining usability usually results from the design requirements gathered during the early stages of the design phase. In our case these will come from research of previous literature rather than user interviews, contextual inquiries and focus groups to name a few.

We will use the definition proposed by a very famous paper [8] and potentially try to expand this definition further to meet the more specific requirements of a password manager software rather an email encryption one used in the paper. Security software is usable if the people who are expected to use it:

1. are reliably made aware of the security tasks they need to perform.
2. are able to figure out how to successfully perform those tasks.
3. don't make dangerous errors.
4. are sufficiently comfortable with the interface to continue using it.

Having defined usability, we proceed with the specific methodologies we propose to evaluate KeePass2.

Ideally, we plan to evaluate KeePass2 in 3 stages:

- *Stage 1 - Evaluation of Original KeePass2*: This will utilise a combination of a Cognitive Walkthrough [15] and a Heuristic Evaluation based on Nielsen's 10 Usability Heuristics [16]. Again, these methodologies will be refined to match any additional security requirements previously identified.
- *Stage 2 - Design and Early evaluation of Revised KeePass2*: During this stage, we will have designed a mockup solution based on the results of the evaluation above. The revised solution can be then evaluated further in a between-subject think aloud study that will compare the original version of KeePass2 against the mockup revised version. The users will be split randomly into 2 groups and asked to complete a set of predefined tasks on either of the 2 software. This method aims to identify strengths and weaknesses of the original and revised version and guarantee that the design is moving in the correct direction. The pre/post questionnaire will include questions about user demographics and general security questions about password authentication to access the level of user background knowledge before and after their exposure with the password managers.
- *Stage 3 - Evaluation of Revised KeePass2*: Using the insights gathered from the above 2 stages we will complete the Revised KeePass2 version (either using online mock up tools or directly through its source code) and perform another think aloud study to assess whether our proposed modifications improved users performance in the context of usable security.

The methodologies mentioned above will be briefly introduced here while more detail will be provided in the project.

4.2.1 Cognitive Walkthrough

These are usability inspection methods. Whereas the heuristic evaluation looks at a product or system holistically, the cognitive walkthrough is task-specific. It is based on the belief that people learn systems by trying to accomplish tasks with it, rather than first reading through instructions. It is ideal for products that are meant to be walk-up and use. State a clear goal that the user wants to achieve and make sure everyone understands that goal. This goal is the primary task. To achieve this task a typical user should go through a specific set of subtasks. The individual conducting the walkthrough presents each subtask to the HCI experts through a series of screens. When each screen is presented, everyone is asked to write down the answers to four questions:

1. Will users want to produce whatever effect the action has?
2. Will users see the control (button, menu, label, etc.) for the action?
3. Once users find the control, will they recognize that it will produce the effect they want?
4. After the action is taken, will users understand the feedback they get, so they can confidently continue to the next action?

Once this process is performed for each task, then results are discussed among the participants and feedback of each of the KCI experts is recorded along with the results.

4.2.2 Heuristic Evaluation

This method was firstly introduced by [16] and was later refined to a 10 Heuristics list in [17]. Three to five UX(user experience) experts (or novices trained on the heuristics) individually assess a product by walking through a core set of tasks and noting any places where heuristics are violated. The evaluators then come together to combine their findings into a single report of issues that should be addressed. Note that, products that adhere to all 10 heuristics are not guaranteed to meet users' needs, but it is significantly less likely that they will face the barriers of poor design. The list of Nielsen's 10 Heuristics can be found in the appendix.

4.2.3 Think-aloud / Usability testing

This methodology entails the observation of end users attempting to complete a predefined set of tasks using a prototype (or complete) version of the product under evaluation. Each participant interacts with the product during individual sessions by performing a series of predefined tasks and user performance on those tasks is recorded in order to identify usability issues. Each user is asked to go through the same tasks while verbalizing whatever comes to his/her mind. The sessions will be recorded, preferably with a video camera and analyzed later by going through the recorded footage.

The exact predefined set of tasks that the user would go through during the think aloud will be decided according to the results of the cognitive walkthrough and heuristic evaluation but none the less they will fall into the following general categories:

- Creation of a new encrypted database
- Populating the new database with password entries. These would be either new entries or existing online accounts that the user will have to migrate to KeePass2
- Logging into KeePass2 - Unlocking the encrypted database - Using saved entries to log in to online accounts
- Changing or Updating Password entries
- Accessing the database from a public PC (this is optional since not all users might choose to save their encrypted database into a cloud storage space.)

The participants will be trained on how to perform a think aloud experiment prior to the actual experiment using a predefined script to avoid any biases.

There is a lot of debate about the number of participants needed for usability evaluation (see [18] for an academic evaluation). [19] found that you get a better return on your investment if you conduct multiple rounds of testing; however, only five participants are needed per round. In other words, you will find more usability issues conducting three rounds of testing with five participants each if you iterate between rounds (i.e., make changes or add features/functionality to your prototype or product based on each round of feedback) than if you conduct a single study with 15 participants.

4.2.4 Pre/Post Questionnaires

The users participating to the think aloud studies will be asked to fill these before and after their interaction with KeePass2. The questionnaires will be used to obtain data about:

- User Demographics

- User general knowledge of Cybersecurity (this might be omitted to avoid unnecessary mental load to participants)
- User attitude towards Password Managers and Passwords
- User satisfaction with KeePass2

5 Evaluation

To evaluate the Usability of the Revised version of KeePass2 we will analyze the behavioural-qualitative data we receive from the last Think Aloud study that we perform, mentioned in the Methods section of this report.

It will however be interesting to compare our revised version of KeePass2 with 2 alternative Password Manager solutions that are currently widely adopted, namely 1Password [https://1password.com/] and LastPass [https://www.lastpass.com/business-password-manager] (DashLane could be an alternative to either of the two proposed). To accomplish this, we will use a list of evaluation criteria for security software solutions proposed by [6]. The list contains 25 key criteria that span 3 broad categories, namely **security**, **deployability** and **usability**. The list is suggested to be used not blindly as a highest score measure for the different solutions under study, but as key points for discussion about the strengths and weaknesses of each proposed system. Due to space constraints the full list is included in the appendix A.2.

6 Outputs

In this section we will briefly outline our expectations of the outcome of our study and consider alternative approaches to some of the methodologies and results we plan to produce.

6.1 Expected Outcomes

As previously mentioned, we expect the outcomes of the study to be twofold:

- It will produce a revised version of KeePass2 that will allow users to securely and effortlessly manage their passwords.
- Provide guidance to future security software engineers as to what to consider while creating new security software.

6.2 Back-Up Plan

6.2.1 User Education

We have proposed that we will try to limit as much as possible the amount of security knowledge and education the user will need in order to securely use KeePass2. This of course might not be possible and some basic advice will have to be given prior to using the system. During a recent study, [7] asked 231 security experts to list the top 3 pieces of advice they would give to a non-tech-savvy user to protect their security online. They received a vast array of advice – 152 unique pieces covering 15 categories. They found that, while individual experts provide plenty of thoughtful and considered advice, the security community as a whole has yet to form consensus

on a prioritized set of advice. We will use the results of [7] that aimed to both provide a set of the most important security advice as well as guidance in how advice should look like in order to be effective to construct any piece of advice that we will provide to the users of KeePass2.

6.2.2 Study Participants

For the current study plan, we would need:

- *Stage 1:* Cognitive Walkthrough and Heuristic Evaluation – 5 HCI experts; These can be MSc students that have previously taken the HCI class in the university of Edinburgh.
- *Stage 2:* Between-Subjects Think Aloud study – 5 users representative of the target population for each of the software evaluated. In case that further literature research and/or stage 1 results reveal no need to perform the Between-Subject lab study we could limit this to only 5 users that will be used to evaluate the revised version of KeePass2 only.
- *Stage 3:* Completed KeePass2 revised version – 5-10 users representative of the target population.

6.2.3 Software

As previously discussed, we plan to provide the revised version of KeePass2 in terms of a refactored version of the original source code. If however time does not allow, the solution will be given by the use of an online mock up application.

7 Workplan

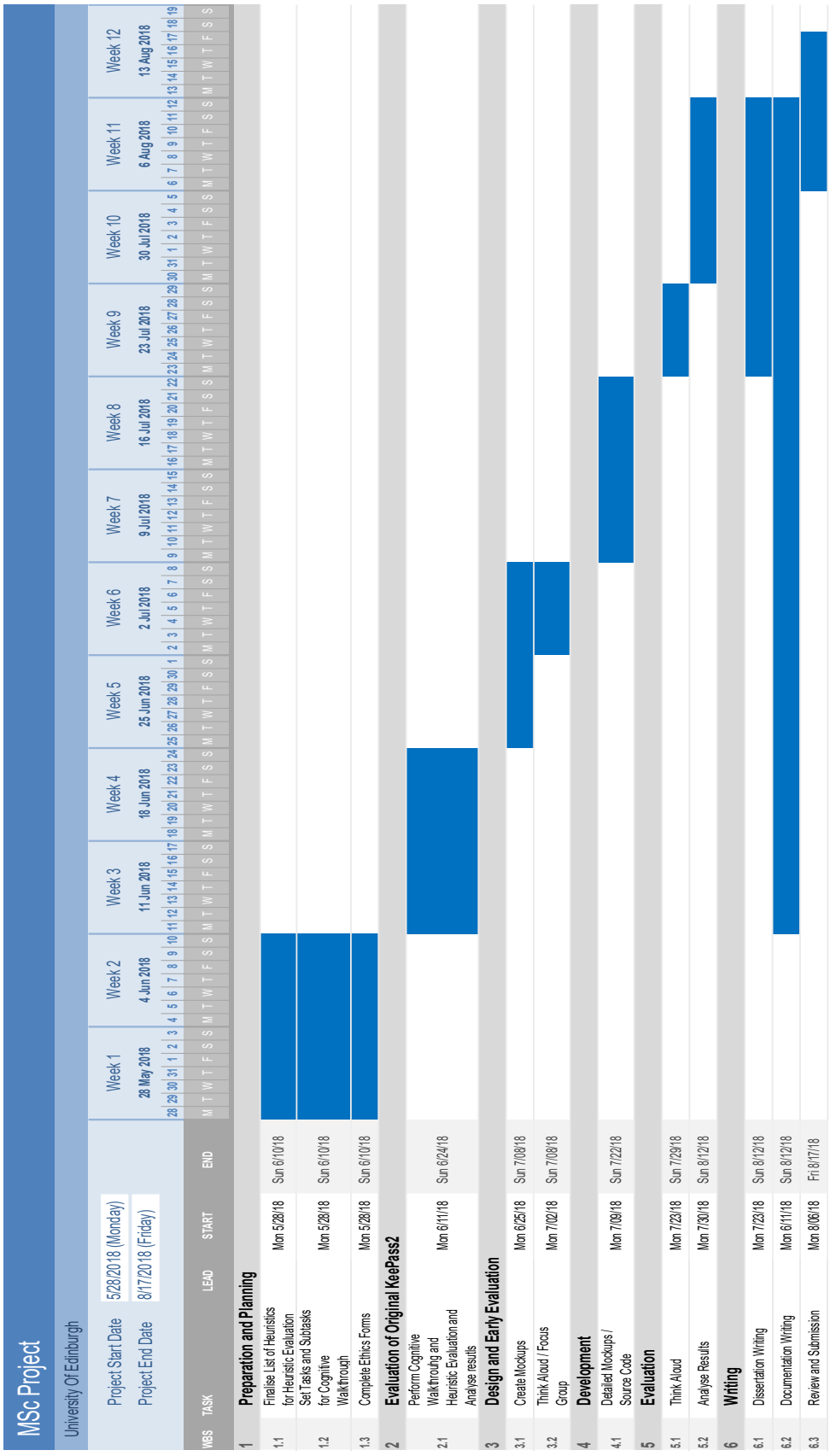

Figure 1: Gantt Chart depicting the planned timeline of the MSc project

References

[1] C. Herley, "So long, and no thanks for the externalities: the rational rejection of security advice by users," *Security*, pp. 133–144, 2009.

[2] B. Schneier, "The psychology of security," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5023 LNCS, no. 4, pp. 50–79, 2008.

[3] S. L. Pfleeger, M. A. Sasse, and A. Furnham, "From weakest link to security hero: Transforming staff security behavior," *Journal of Homeland Security and Emergency Management*, vol. 11, no. 4, pp. 489–510, 2014.

[4] D. Florencio and C. Herley, "A large-scale study of web password habits," *Proceedings of the 16th international conference on World Wide Web - WWW '07*, p. 657, 2007.

[5] A. Adams and M. A. Sasse, "Users are not the enemy," *Communications of the ACM*, vol. 42, no. 12, pp. 40–46, 1999.

[6] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," *Proceedings - IEEE Symposium on Security and Privacy*, pp. 553–567, 2012.

[7] R. W. Reeder, I. Ion, and S. Consolvo, "152 Simple Steps to Stay Safe Online: Security Advice for Non-Tech-Savvy Users," *IEEE Security and Privacy*, vol. 15, no. 5, pp. 55–64, 2017.

[8] A. Whitten and J. Tygar, "Why Johnny can't encrypt: A usability evaluation of PGP 5.0," *Proceedings of the 8th USENIX Security Symposium*, pp. 169–184, 1999.

[9] D. McCarney, D. Barrera, J. Clark, S. Chiasson, and P. C. van Oorschot, "Tapas: Design, Implementation, and Usability Evaluation of a Password Manager," *Proceedings of the 28th Annual Computer Security Applications Conference on - ACSAC '12*, pp. 89–98, 2012.

[10] S. Chiasson, P. V. Oorschot, and R. Biddle, "A usability study and critique of two password managers," *15th USENIX Security ...*, no. August, pp. 1–16, 2006.

[11] J. M. GOSNEY, "How linkedin's password sloppiness hurts us all."

[12] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. L. Mazurek, T. Passaro, R. Shay, T. Vidas, L. Bauer, N. Christin, and L. F. Cranor, "How Does Your Password Measure Up? The Effect of Strength Meters on Password Creation Blase," *Security'12 Proceedings of the 21st USENIX conference on Security symposium*, pp. 5–16, 2012.

[13] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," *Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI '06*, no. November 2005, p. 581, 2006.

[14] J. Nielsen, "Usability inspection methods," *Conference companion on Human factors in computing systems - CHI '94*, pp. 413–414, 1994.

[15] C. Lewis, P. G. Polson, C. Wharton, and J. Rieman, "Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces," *Proceedings of the SIGCHI conference on Human factors in computing systems Empowering people - CHI '90*, pp. 235–242, 1990.

[16] J. Nielsen and R. Molich, "Heuristic Evaluation of user interfaces," *CHI '90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, no. April, pp. 249–256, 1990.

[17] J. Nielsen, "10 usability heuristics for user interface design," *Nielsen Norman Group*, vol. 1, no. 1, 1995.

[18] S. Borsci, R. D. Macredie, J. Barnett, J. Martin, J. Kuljis, and T. Young, "Reviewing and Extending the Five-User Assumption," *ACM Transactions on Computer-Human Interaction*, vol. 20, no. 5, pp. 1–23, 2013.

[19] a. Mathematical, "Model of the Finding of Usability Problems," *Transport*, pp. 206–213, 1910.

Appendices

A HCI related

A.1 Extra difficulties of Usable Security

Extra difficulties of Usable Security Identified in [8]:

1. **The unmotivated user property:** Security is usually a secondary goal. People do not generally sit down at their computers wanting to manage their security; rather, they want to send email, browse web pages, or download software, and they want security in place to protect them while they do those things. It is easy for people to put off learning about security, or to optimistically assume that their security is working, while they focus on their primary goals. Designers of user interfaces for security should not assume that users will be motivated to read manuals or to go looking for security controls that are designed to be unobtrusive. Furthermore, if security is too difficult or annoying, users may give up on it altogether.
2. **The abstraction property:** Computer security management often involves security policies, which are systems of abstract rules for deciding whether to grant accesses to resources. The creation and management of such rules is an activity that programmers take for granted, but which may be alien and unintuitive to many members of the wider user population. User interface design for security will need to take this into account.
3. **The lack of feedback property:** The need to prevent dangerous errors makes it imperative to provide good feedback to the user, but providing good feedback for security management is a difficult problem. The state of a security configuration is usually complex and attempts to summarize it are not adequate. Furthermore, the correct security configuration is the one which does what the user "really wants", and since only the user knows what that is, it is hard for security software to perform much useful error checking.
4. **The barn door property:** The proverb about the futility of locking the barn door after the horse is gone is descriptive of an important property of computer security: once a secret has been left accidentally unprotected, even for a short time, there is no way to be sure that it has not already been read by an attacker. Because of this, user interface design for security needs to place a very high priority on making sure users understand their security well enough to keep from making potentially high-cost mistakes.
5. **The weakest link property:** It is well known that the security of a networked computer is only as strong as its weakest component. If a cracker can exploit a single error, the game is up. This means that users need to be guided to attend to all aspects of their security, not left to proceed through random exploration as they might with a word processor or a spreadsheet.

A.2 Evaluation framework for Security Systems

Evaluation framework for assessing security systems in terms of Security, Usability and Deployability as suggested by [6]:

Usability benefits:

- U1 Memorywise-Effortless: Users of the scheme do not have to remember any secrets at all. We grant a Quasi-Memorywise-Effortless if users have to remember one secret for everything (as opposed to one per verifier).
- U2 Scalable-for-Users: Using the scheme for hundreds of accounts does not increase the burden on the user. As the mnemonic suggests, we mean "scalable" only from the user's perspective, looking at the cognitive load, not from a system deployment perspective, looking at allocation of technical resources.
- U3 Nothing-to-Carry: Users do not need to carry an additional physical object (electronic device, mechanical key, piece of paper) to use the scheme. Quasi-Nothing-to-Carry is awarded if the object is one that they'd carry everywhere all the time anyway, such as their mobile phone, but not if it's their computer (including tablets).
- U4 Physically-Effortless: The authentication process does not require physical (as opposed to cognitive) user effort beyond, say, pressing a button. Schemes that don't offer this benefit include those that require typing, scribbling or performing a set of motions. We grant Quasi-Physically-Effortless if the user's effort is limited to speaking, on the basis that even illiterate people find that natural to do.
- U5 Easy-to-Learn: Users who don't know the scheme can figure it out and learn it without too much trouble, and then easily recall how to use it. U6 Efficient-to-Use: The time the user must spend for each authentication is acceptably short. The time required for setting up a new association with a verifier, although possibly longer than that for authentication, is also reasonable.
- U7 Infrequent-Errors: The task that users must perform to log in usually succeeds when performed by a legitimate and honest user. In other words, the scheme isn't so hard to use or unreliable that genuine users are routinely rejected.2 U8 Easy-Recovery-from-Loss: A user can conveniently regain the ability to authenticate if the token is lost or the credentials forgotten. This combines usability aspects such as: low latency before restored ability; low user inconvenience in recovery (e.g., no requirement for physically standing in line); and assurance that recovery will be possible, for example via built-in backups or secondary recovery schemes. If recovery requires some form of re-enrollment, this benefit rates its convenience.

Deployability benefits

- D1 Accessible: Users who can use passwords3 are not prevented from using the scheme by disabilities or other physical (not cognitive) conditions.
- D2 Negligible-Cost-per-User: The total cost per user of the scheme, adding up the costs at both the prover's end (any devices required) and the verifier's end (any share of the equipment and software required), is negligible. The scheme is plausible for startups with no per-user revenue.
- D3 Server-Compatible: At the verifier's end, the scheme is compatible with text-based passwords. Providers don't have to change their existing authentication setup to support the scheme. D4 Browser-Compatible: Users don't have to change their client to support the scheme and can expect the scheme to work when using other machines with an up-to-date, standards-compliant web browser and no additional software. In 2012, this would mean an HTML5-compliant browser with JavaScript enabled. Schemes fail to provide this

benefit if they require the installation of plugins or any kind of software whose installation requires administrative rights. Schemes offer Quasi-Browser-Compatible if they rely on non-standard but very common plugins, e.g., Flash.

- D5 Mature: The scheme has been implemented and deployed on a large scale for actual authentication purposes beyond research. Indicators to consider for granting the full benefit may also include whether the scheme has undergone user testing, whether the standards community has published related documents, whether open-source projects implementing the scheme exist, whether anyone other than the implementers has adopted the scheme, the amount of literature on the scheme and so forth.
- D6 Non-Proprietary: Anyone can implement or use the scheme for any purpose without having to pay royalties to anyone else. The relevant techniques are generally known, published openly and not protected by patents or trade secrets.

Security benefits:

- S1 Resilient-to-Physical-Observation: An attacker cannot impersonate a user after observing them authenticate one or more times. We grant Quasi- Resilient-to-Physical-Observation if the scheme could be broken only by repeating the observation more than, say, 10–20 times. Attacks include shoulder surfing, filming the keyboard, recording keystroke sounds, or thermal imaging of keypad.
- S2 Resilient-to-Targeted-Impersonation: It is not possible for an acquaintance (or skilled investigator) to impersonate a specific user by exploiting knowledge of personal details (birth date, names of relatives etc.). Personal knowledge questions are the canonical scheme that fails on this point.
- S3 Resilient-to-Throttled-Guessing: An attacker whose rate of guessing is constrained by the verifier cannot successfully guess the secrets of a significant fraction of users. The verifier-imposed constraint might be enforced by an online server, a tamper-resistant chip or any other mechanism capable of throttling repeated requests. To give a quantitative example, we might grant this benefit if an attacker constrained to, say, 10 guesses per account per day, could compromise at most 1accounts in a year. Lack of this benefit is meant to penalize schemes in which it is frequent for user-chosen secrets to be selected from a small and well-known subset (low min-entropy [14]).
- S4 Resilient-to-Unthrottled-Guessing: An attacker whose rate of guessing is constrained only by available computing resources cannot successfully guess the secrets of a significant fraction of users. We might for example grant this benefit if an attacker capable of attempting up to 240 or even 264 guesses per account could still only reach fewer than 1is meant to penalize schemes where the space of credentials is not large enough to withstand brute force search (including dictionary attacks, rainbow tables and related brute force methods smarter than raw exhaustive search, if credentials are user-chosen secrets).
- S5 Resilient-to-Internal-Observation: An attacker cannot impersonate a user by intercepting the user's input from inside the user's device (e.g., by keylogging malware) or eavesdropping on the cleartext communication between prover and verifier (we assume that the attacker can also defeat TLS if it is used, perhaps through the CA). As with Resilient-to-Physical-Observation above, we grant Quasi-Resilient-to-Internal-Observation if the scheme could be broken only by intercepting input or eavesdropping cleartext more than, say, 10–20 times. This penalizes schemes that are not replay-resistant, whether because they send a static response or because their dynamic response countermeasure can be cracked with a few observations. This benefit assumes that generalpurpose devices like software-updatable personal computers and mobile phones may contain malware, but that hardware devices dedicated exclusively to the scheme can be made malware-free. We grant Quasi-Resilient-to-Internal-Observation to two-factor schemes where both factors must be malware-infected for the attack to work. If infecting only one factor breaks the scheme,

we don't grant the benefit.

- S6 Resilient-to-Leaks-from-Other-Verifiers: Nothing that a verifier could possibly leak can help an attacker impersonate the user to another verifier. This penalizes schemes where insider fraud at one provider, or a successful attack on one back-end, endangers the user's accounts at other sites.
- S7 Resilient-to-Phishing: An attacker who simulates a valid verifier (including by DNS manipulation) cannot collect credentials that can later be used to impersonate the user to the actual verifier. This penalizes schemes allowing phishers to get victims to authenticate to lookalike sites and later use the harvested credentials against the genuine sites. It is not meant to penalize schemes vulnerable to more sophisticated real-time man-in-the-middle or relay attacks, in which the attackers have one connection to the victim prover (pretending to be the verifier) and simultaneously another connection to the victim verifier (pretending to be the prover).
- S8 Resilient-to-Theft: If the scheme uses a physical object for authentication, the object cannot be used for authentication by another person who gains possession of it. We still grant Quasi-Resilient-to- Theft if the protection is achieved with the modest strength of a PIN, even if attempts are not ratecontrolled, because the attack doesn't easily scale to many victims.
- S9 No-Trusted-Third-Party: The scheme does not rely on a trusted third party (other than the prover and the verifier) who could, upon being attacked or otherwise becoming untrustworthy, compromise the prover's security or privacy.
- S10 Requiring-Explicit-Consent: The authentication process cannot be started without the explicit consent of the user. This is both a security and a privacy feature (a rogue wireless RFID-based credit card reader embedded in a sofa might charge a card without user knowledge or consent).
- S11 Unlinkable: Colluding verifiers cannot determine, from the authenticator alone, whether the same user is authenticating to both. This is a privacy feature. To rate this benefit we disregard linkability introduced by other mechanisms (same user ID, same IP address, etc).

A.3 Why Phishing is effective

This is the complete list of results of the study [13] to identify the main dimensions of phishing strategies.

1. Lack of Knowledge

- **Lack of computer system knowledge.** Many users lack the underlying knowledge of how operating systems, applications, email and the web work and how to distinguish among these. Phishing sites exploit this lack of knowledge in several ways. For example, some users do not understand the meaning or the syntax of domain names and cannot distinguish legitimate versus fraudulent URLs (e.g., they may think www.ebay-members-security.com belongs to www.ebay.com). Another attack strategy forges the email header; many users do not have the skills to distinguish forged from legitimate headers.
- **Lack of knowledge of security and security indicators.** Many users do not understand security indicators. For example, many users do not know that a closed padlock icon in the browser indicates that the page they are viewing was delivered securely by SSL. Even if they understand the meaning of that icon, users can be fooled by its placement within the body of a web page (this confusion is not aided by the fact that competing browsers use different icons and place them in different parts of their display). More generally, users may not be aware that padlock icons appear in the browser "chrome" (the

interface constructed by the browser around a web page, e.g., toolbars, windows, address bar, status bar) only under specific conditions (i.e., when SSL is used), while icons in the content of the web page can be placed there arbitrarily by designers (or by phishers) to induce trust.

Attackers can also exploit users' lack of understanding of the verification process for SSL certificates. Most users do not know how to check SSL certificates in the browser or understand the information presented in a certificate. In one spoofing strategy, a rogue site displays a certificate authority's (CA) trust seal that links to a CA webpage. This webpage provides an English language description and verification of the legitimate site's certificate. Only the most informed and diligent users would know to check that the URL of the originating site and the legitimate site described by the CA match.

2. Visual Deception Phishers use visual deception tricks to mimic legitimate text, images and windows. Even users with the knowledge described in (1) above may be deceived by these.

- Visually deceptive text. Users may be fooled by the syntax of a domain name in "type-jacking" attacks, which substitute letters that may go unnoticed (e.g. www.paypai.com uses a lowercase "i" which looks similar to the letter "l", and www.paypa1.com substitutes the number "1" for the letter "l"). Phishers have also taken advantage of non-printing characters and non-ASCII Unicode characters in domain names.
- Images masking underlying text. One common technique used by phishers is to use an image of a legitimate hyperlink. The image itself serves as a hyperlink to a different, rogue site.
- Images mimicking windows. Phishers use images in the content of a web page that mimic browser windows oror dialog windows. Because the image looks exactly like a real window, a user can be fooled unless he tries to move or resize the image.
- Windows masking underlying windows. A common phishing technique is to place an illegitimate browser window on top of, or next to, a legitimate window. If they have the same look and feel, users may mistakenly believe that both windows are from the same source, regardless of variations in address or security indicators. In the worst case, a user may not even notice that a second window exists (browsers that allow borderless pop-up windows aggravate the problem).
- Deceptive look and feel. If images and logos are copied perfectly, sometimes the only cues that are available to the user are the tone of the language, misspellings or other signs of unprofessional design. If the phishing site closely mimics the target site, the only cue to the user might be the type and quantity of requested personal information.

3. Bounded Attention Even if users have the knowledge described in (1) above, and can detect visual deception described in (2) above they may still be deceived if they fail to notice security indicators (or their absence).

- Lack of attention to security indicators. Security is often a secondary goal. When users are focused on their primary tasks, they may not notice security indicators or read warning messages. The image-hyperlink spoof described in (2b) above would thwarted if user noticed the URL in the status bar did not match the hyperlink image, but this requires a high degree of attention. Users who know to look for an SSL closed-padlock icon may simply scan for the presence of a padlock icon regardless of position and thus be fooled by an icon appearing in the body of a web page.
- Lack of attention to the absence of security indicators. Users do not reliably notice the absence of a security indicator. The Firefox browser shows SSL protected pages with four indicators (see Figure 1). It shows none of these indicators for pages not protected by SSL. Many users do not notice the absence of an indicator, and it is sometimes possible to insert a spoofed image of an indicator where one does not exist.

B KeePass2 related

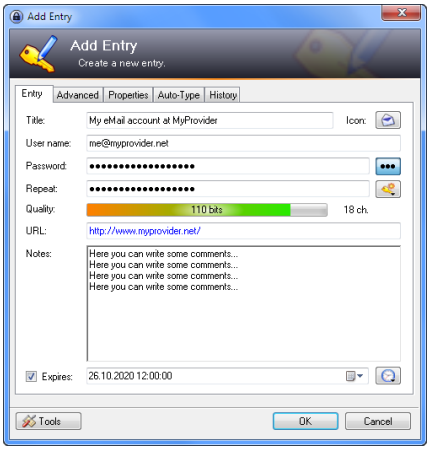

Figure 2: UI for adding a new entry to already created encrypted database

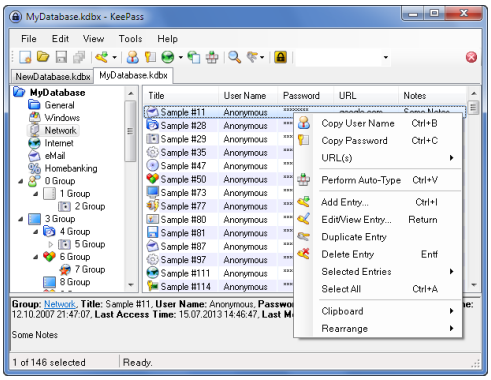

Figure 3: Main UI screen: On the left hand side compartment an outline of the database folders is presented - On the right hand side compartment the contents (individual password entries) of the specific folder is depicted

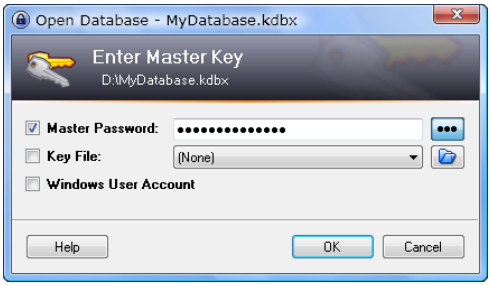

Figure 4: UI for providing the Master Password