The Usable Security of Passwords based on Digital Objects: From Design and Analysis to User Study*

Mohammad Mannan
Electrical and Computer Engineering Dept.
University of Toronto
Toronto, Canada
m.mannan@utoronto.ca

Tara Whalen, Robert Biddle
P.C. van Oorschot
School of Computer Science
Carleton University
Ottawa, Canada

Abstract

Despite all efforts, password schemes intended to deploy or encourage the use of strong passwords have largely failed. As an interesting alternative to enable users to create, maintain and use high quality passwords willingly, we propose Object-based Password (*ObPwd*), leveraging the universe of personal or personally meaningful digital content that many users now own or have access to. ObPwd converts user-selected digital objects to high-entropy text passwords. Memorization of exact passwords is replaced by remembering password objects. We present the design details, variants, and usability and security analysis of ObPwd; briefly discuss (publicly available) prototype implementations in various forms on several platforms; and as a major focus, report on the results of a hybrid in-lab/at-home user study on 32 participants. The results suggest the scheme has good usability, with excellent memorability, acceptable login times, and very positive user perception, achieved while providing strong security for the threat context explored. While we anticipate further experience with ObPwd will lead to improved security and usability, and best practice guidelines, we believe this work lays the foundations for a promising password selection paradigm.

1 Introduction and Motivation

Text passwords remain ubiquitous, despite endless criticism. Independent studies conducted decades apart reveal that people consistently choose 'weak' passwords [19, 6] for many reasons, including users trying to manage on average 25 password-protected accounts [7]. Losing strategies include blaming users, and imposing complex password rules. Some claim that choosing weak passwords (despite repeated advice otherwise) is a rational economic response [13]. As alternatives to passwords come with their own deployment barriers, passwords appear likely to continue to dominate user authentication for the foreseeable future, despite repeated rumors of their impending death.

Some argue [13, 7] that strong passwords are not essential for preventing automated online dictionary attacks; for example, password-protected sites can present challenge CAPTCHAs after (e.g., three) failed attempts, or lock out the targeted account temporarily. However, the latter can affect legitimate users, and CAPTCHA schemes are regularly defeated by improved attacks in the artificial intelligence arms-race, by human ("sweat-shop") solvers, or bypassed due to implementation flaws [34]. Bulk guessing attacks [7] may yield access to accounts when attackers know many valid userids, even if lock-out rules are used. Passwords used for applications other than remote authentication, including file/disk encryption and encrypting private keys, are also subject to offline dictionary attack. Patterns in user-chosen passwords allow *space-reduction* attacks [37] and efficient construction of password dictionaries [20, 33]. The recent rise in SSH password-guessing attacks[1] highlights that strong passwords remain important despite increasing password stealing attacks, phishing, pharming and keylogging.

*Version: February 16, 2010. Parts of this work appeared in preliminary form in a 6-page workshop paper [15].

[1]One experimental setup [26] reported on average 2,805 SSH login attempts per computer per day.

Even if, through education, persuasion and proactive checking [38], users choose strong passwords for everyday use, they typically remain usable (memorable) only if used often. Indeed, passwords for rarely-used services, or *secondary authentication* (e.g., when a user has lost the primary password) such as Personal Validation Questions (PVQs) pose even greater usability and security challenges; users choose weak passwords/secrets that are difficult to forget or obvious when given a hint [25, 28].

To address these issues, we introduce *ObPwd*, an object-based password scheme to generate passwords used infrequently, used in common web authentication, or used to access encryption keys. The basic idea is as follows. Many users currently possess a large collection of digital content such as photos, audio recordings, videos, documents and email messages. Much of this content is *mobile*: stored on personal devices (e.g., USB drives, cellphones, laptops), protected remote servers (e.g., personal email providers with HTTPS access), or uploaded to personal sites (some password-protected). Many users also have instant access to static content from the web, e.g., Internet Archive (`archive.org`), Project Gutenberg (`gutenberg.org`). ObPwd generates a password from such items by computing a hash from the user-selected object (e.g., photo file) then converting the hash bitstring, by known techniques [12, 11, 27], to an appropriate password format, e.g., a string of keyboard characters or optionally a human readable (even smartphone-friendly) word sequence.

In place of remembering exact passwords or passphrases, users only need a strategy to remember *which* password object they chose (e.g., hints for an image, video, entire/partial document, URL, or text passage from a web page). Recalling or browsing through such personal and emotionally meaningful content appears to be more satisfying and rewarding (and was found to be well-received in user tests) than complying with standard password guidelines and procedures. Users can use affective objects for authentication in existing password-protected sites. While being more satisfying does not itself increase security, the underlying entropy of digital objects (in the threat model assumed) together with the rejection of password advice in traditional schemes, provides ObPwd security and usability advantages. Users must keep a record (memorized or written) of a hint for their content used in generating each password. The generated password can be written down in a 'secure' place, or re-created from content when needed. ObPwd requires no modifications to password system interfaces, and the system side (remote or local) need not be aware of ObPwd, facilitating deployment. For other benefits and features, see Section 7.3. Our user study shows promising usability results, further supported by informal comments and popularity in a real-world deployment.

Our contributions include the basic design and variants of an object-based drop-in replacement for text password schemes; usability and security analysis; results and interpretation of a 32-participant hybrid user study, including exploration of performance, user acceptance and multi-password interference; and publicly available implementations (Firefox browser extension, and stand-alone applications in Microsoft Windows, Mac OS X, and Linux).

2 Object-based Password Scheme

This section presents details of ObPwd, the operational model, variants of the basic idea, and implementations.

2.1 Operational Assumptions, Model and Design

We assume that password-generating objects are selected mainly from (i) a user's personal, locally stored digital content; and (ii) a large and preferably stable public collection of files (e.g., pdf files and text strings therein), including from large academic digital archives (of millions of documents, each with thousands of words). The idea is that the inaccessibility of private content, and/or the large size of pools of source objects, precludes effective offline dictionaries. Ideally users would not choose objects from their (publicly accessible) personal website or public profiles in Facebook/MySpace sites. (However, salting such objects may adequately reduce risks; see Variant 1 below.) To enable *access-from-anywhere*, users carry password-generating objects with them, or have online access to the objects (e.g., email messages). Passwords generated by ObPwd, and hints (text reminders) to objects, can optionally be written down. Written passwords may be used as an alternative access means when a user does not carry content files with her.

ObPwd requires a malware-free user environment (as do text passwords), and may be at risk to network-based observation if public password objects are retrieved for use in web login. Attacks and countermeasures are discussed in Section 6.2; depending on the application, variants may be suitable to counter some attacks (see below). ObPwd is designed to increase usability while generating 'strong' passwords by leveraging distinctive object choices made by a user, and leveraging personal content. The following notation is used:

U	An ObPwd user.
M	A password object selected by U for a target task.
$h(\cdot)$	An appropriate cryptographic hash function.
pwd	A password as generated by Hash2Text$(\cdot)$.
Hash2Text$(\cdot)$	A function (e.g., based on [12, 27]) converting binary hash output to keyboard character strings.

Steps in ObPwd (see also Fig. 1):

1. U selects a memorable object M from personal media or the web. To preclude offline dictionary attacks and predictable object prefixes, M must exceed a minimum size (e.g., $m = 160$ bytes). To bound the time to hash very large objects (e.g., 4GB movie files), M is truncated to n bytes (e.g., $n = 100,000$).
2. U indicates the selected object to the ObPwd tool (e.g., through a file dialog, or drag and drop interface), which generates the hash $H = h(M)$.
3. Generate pwd = Hash2Text(H). H may be truncated depending on the required size of pwd. M and pwd should not be stored on the same media as the protected content. For web login, pwd may require special encoding (as in [27]; see also Section 6.1).
4. Then pwd may be copied to a desired site/application, written down or saved, or used to generate high-entropy encryption keys. Copy-pasting of pwd may be automated in some settings; see Section 2.2.

ObPwd variants can increase security, albeit decreasing usability; their suitability depends on target environments.

Variant 1: Salted ObPwd. Use $H = h(M, s)$, appending a user-selected *salt* s, e.g., a weak password, or PIN, as an input. The cost: memorizing s. If used only rarely, s may be written down in a safe place, rather than memorized. A user-selected n in place of the current limit ($n = 100,000$ bytes) may serve as an alternative form of salt, as would changing ObPwd to allow user-specified ranges (e.g., bytes n_1 through n_2 for suitable n_i).

Variant 2: Multi-stage/mixed-object ObPwd. Use $H = h(M_1, M_2, ...)$ where M_i are multiple user-selected objects (e.g., two photos, or a music file and text string). Public and private objects could also be mixed (e.g., a Wikimedia image and a text string from a private document file).

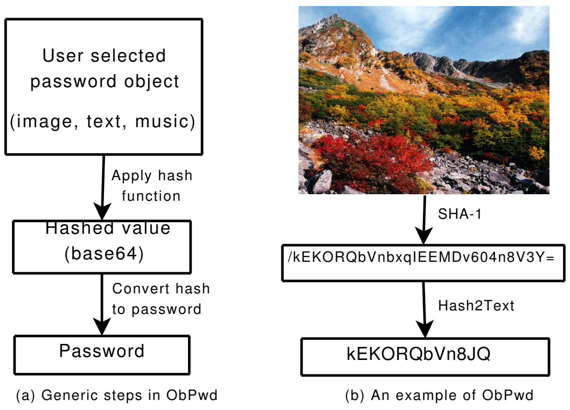


Figure 1: ObPwd steps with an example

Variant 3: Anti-phishing ObPwd. Use $H = h(M, url)$, appending the URL of a target site (cf. PwdHash [27]). This can be implemented in a browser extension without user involvement. However, for ObPwd stand-alone applications, users must drag-drop, copy-paste or type-in url.

Variant 4: ObPwd on untrusted platforms. In a compromised PC, ObPwd cannot protect pwd or M (pwd might resist simple keyloggers when copy-pasted to password boxes). ObPwd can enhance password mechanisms resistant to PC malware (e.g., Bumpy [18], MP-Auth [16]), e.g., replacing MP-Auth's user-chosen password by generating ObPwd passwords on malware-free personal mobile devices.

2.2 Publicly Available Implementations

ObPwd implementations are publicly available as a Firefox browser extension (improved from an original version), a stand-alone Microsoft Windows application, a Mac OS X application, and a Linux command-line application. Our initial Firefox extension and prototypes on other major platforms were well-received, and public feedback has resulted in modifications and upgrades.[2]

ObPwd Firefox extension. Our extension can be activated from the browser context menu (i.e., right-/secondary-click menu). Under the "Object-based Password (ObPwd)" menu, several sub-menus appear (depending on the right-click context, see Fig. 2): (i) "Get ObPwd from Local File" brings up a file dialog box for selecting a local file as a password object; (ii) "Get ObPwd from Selected Text" generates a password using the selected text block on the web page (if there is any selected text string); (iii) "Get ObPwd from Image" generates a password from the selected image (i.e., the one right-clicked on, if any); (iv) "Get ObPwd from Link" generates a password from the content as pointed by the URL right-clicked on, if any. Certain types of relatively stable HTTP and HTTPS links are supported by default (e.g., pdf, mp3, avi, txt, jpg, zip, wav), but not several common URL extensions (e.g., html, php, asp) which may host dynamic content — e.g., the content of a news page may change as user comments are added, precluding regeneration of the original password.

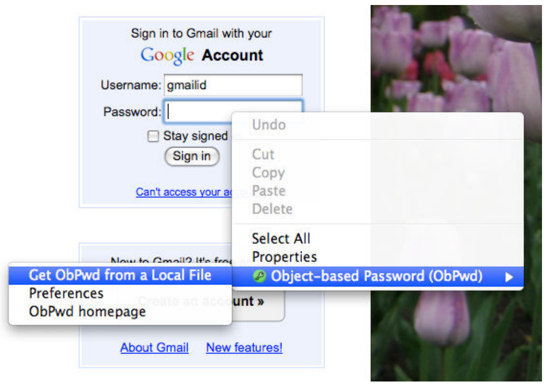
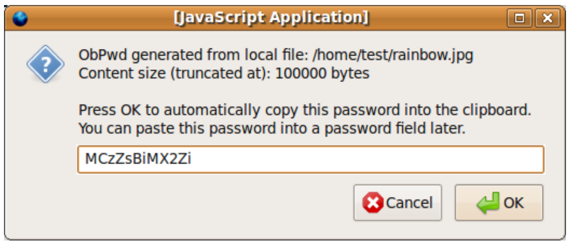


Figure 2: ObPwd extension menu in Firefox

The restriction on HTTP and HTTPS links can be overridden by changing preferences in the ObPwd "Preferences" dialog, at the risk of losing ObPwd passwords generated from dynamic pages. The generated passwords might thus be manually written down or saved a priori, but this brings usability costs. To comply with password rules in some sites, ObPwd allows changing the default pwd length (6 to 20 characters, default 12) and including special characters ('+' or '/') in Hash2Text. The implemented extension does not currently store site-specific preferences. Thus if a password is generated with certain preferences, the same preferences must be selected to re-create that password. More options can be added as preferences, e.g., other special characters, or restricting Hash2Text to digits. As such options may make ObPwd more appealing to advanced users, but complicate use by common users, we implemented a simple set of preferences and recommend using default options.


[JavaScript Application]

ObPwd generated from local file: /home/test/rainbow.jpg
Content size (truncated at): 100000 bytes

Press OK to automatically copy this password into the clipboard.
You can paste this password into a password field later.

MCzZsBiMX2Zi

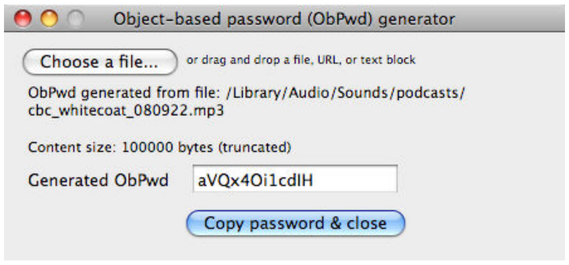
Figure 3: Password generated from a local image file

When a password object (an image, highlighted text, URL, or a local file) is selected, the extension generates a password from the underlying content and displays the password in a dialog box (Fig.3) in plaintext to enable users to write it down or save it in a secure way. If the OK button is clicked, the password is copied to the system clipboard for later pasting anywhere by the user. Note that by default, for security and privacy reasons, Firefox clipboard data is not accessible to JavaScript embedded on a site.[3] The password is inserted directly into a password input box on a login page, if the extension is activated from such a box (i.e., the context menu is brought up by right-clicking on the password box). This auto-filling automates the password copy-paste step, and hides the password from shoulder-surfing attackers.

[2]The ObPwd FAQ and download page is available at `http://www.ccsl.carleton.ca/~mmannan/obpwd/`.
[3]`http://kb.mozillazine.org/Granting_JavaScript_access_to_the_clipboard`.

Some web pages may disable right-click, and some sites are entirely implemented using Flash. The extension cannot be directly activated from such sites; users may generate the ObPwd password from any other page (i.e., in another browser tab or window) to paste onto those sites.

ObPwd stand-alone applications. In the Mac application (see Fig. 4), users can select a file from a file dialog by using the "Choose a file..." button. Users can also drag and drop the following objects on the application window: files from Finder, images from iPhoto, selected text blocks from any application (e.g., a browser or word processor), and URLs. Clicking the "Copy password & close" button copies the generated password to the clipboard which then can be pasted on any authenticating site/application. Preferences similar to the Firefox extension are also supported. In contrast, our current Windows and Linux implementations are primitive; only files can be used in the Windows application, and the Linux application has no GUI (but supports files and text strings). We focused on the Firefox extension as it is deployable on any OS where Firefox is available, but are also exploring implementations in other popular browsers (e.g., IE), and mobile platforms (e.g., iPhone, Symbian, Android).


Object-based password (ObPwd) generator
Choose a file...
or drag and drop a file, URL, or text block
Copy password & close

ObPwd generated from file: /Library/Audio/Sounds/podcasts/cbc_whitecoat_080922.mp3

Content size: 100000 bytes (truncated)

Generated ObPwd aVQx4Oi1cdlH

Figure 4: ObPwd application for Mac OS X

Additional notes on implementation. ObPwd uses SHA-1 to hash objects. Its output is mapped to a base-64 character set, then converted to an alphanumeric password (default 12 characters) by known techniques [27]. Up to the first $n = 100,000$ bytes are used from an object; 160 bytes are required. For mobility, if an ObPwd tool is not available on a given device, a website for generating passwords from user objects could be designed (cf. `pwdhash.com` [27]).

Various technical/usability questions received (informally via email and online comments) from users are clarified in an online FAQ page: e.g., ObPwd does not remember user passwords (it is not a password manager but can be used with one), and authenticating sites receive the output password but not the password object. Modifying password objects prevents re-generating old passwords, but users may rename, copy, or move file objects as ObPwd uses no file metadata (except when metadata is embedded in the file itself).

Section 3 reports on a user study on the Firefox extension, simplified to accept only local files as objects, and hard-coded to generate fixed-length 12-character alphanumeric passwords. Scientific study of the usability of other prototypes remains to be done.

3 ObPwd User Study: Methodology

Here we report on a formal user study, designed to explore usable security of ObPwd. We detail the methodology in order that this study can be replicated by others. Section 4 presents the results, which are interpreted in Section 5.

3.1 Study Goals and Setup

Our hybrid study combined two lab sessions with a phase conducted in participants' regular "at-home" environment to approximate an ecologically valid context of use. The lab sessions were designed to carefully measure usability factors, such as effectiveness (e.g., login success rate), efficiency (e.g., login times), and satisfaction (e.g., perceived ease-of-use). The at-home component allowed participants to use ObPwd with real-world websites of their choice, to examine naturalistic behavior outside of the controlled lab setting. We also wished to develop an understanding of the types of files users select to create their passwords, and the reasons for choosing specific files. For example, do users pick mostly music files, or the same type of file for all logins? When they consider which password file to choose, are they concerned mainly with convenience, or memorability?

Memory load was another important aspect explored. The lab study had two sessions, 7–10 days apart, to evaluate participants' ability to recall passwords over time. The design of our study also incorporated password interference (cf. [2, 5]), asking participants to use a different password for each of 8 study websites.

Test websites. In each lab session, participants logged into four websites created for study purposes, varied in appearance, content, and implied level of sensitivity. Participants followed a role-playing scenario, as a new purchasing department employee required to create new web accounts. The first lab session involved websites for a credit union, employee assistance (counseling), intramural sports, and the purchasing department's internal site. In the second session, the sites were for health insurance, human resources, corporate finance, and a news blog. Each site had a unique appearance (e.g., different color schemes and images) for visual differentiation. Participants registered on each site, logged in, and were tasked to look for a piece of information on the main page, such as a departmental phone number.

Participants. Participants were recruited within a university campus, using email lists and an institutional research study recruitment website. Participants were required to use Firefox regularly, and to provide a laptop that they used regularly, on which they consented to install the ObPwd extension.

Thirty-two participants (20 female, 12 male) completed all parts of the study. (One additional person failed to complete the study due to illness; that person's data was omitted from analysis.) Participants ranged in age from 16–59, with a mean age of 22. 75% of participants used the web more than 10 hours per week. On a 1-7 scale of concern about the security of passwords online (1: "not at all concerned," 7: "very concerned") the mean rating was 5.57. Participants indicated what types of computer tasks they had experience with: 16% had programming experience, 41% had created web pages, and all had installed software. Five participants used Mac and the rest used Windows (no Linux users). Participants familiar with Firefox were recruited, which might suggest a bias towards advanced users. However, per recent statistics,[4] Firefox is now used by numbers of Internet users well beyond solely experts.

3.2 Procedure

The user study consisted of three parts: two lab sessions and one at-home component. Each lab session took approximately one hour, and the at-home tasks took about 30 minutes in total. The lab sessions took place in an office; participants brought their own laptops, which were used to complete all experimental tasks. This allowed naturalistic use of ObPwd, within a user's regular computing environment, and allowed participants to access their own digital content.

a) Lab Session 1. After informed consent was obtained, participants were given a brief demonstration which introduced them to the ObPwd Firefox extension, using a computer we provided. While in real-life, users may not get any demonstration or training before their first use of every newly deployed security tool, we provided a brief demo in order to explore performance among users with a basic familiarity. We acknowledge, however, that the brief training fails to offset the countless years of training that all users have with text passwords. We did not mention to participants whether or not that those running the study had any involvement in the design of ObPwd, nor did we suggest any expectation of ObPwd performing well.

Once the participant had a chance to understand the features of ObPwd, they were asked for permission to install the extension on their own laptop. After installation, participants were given a tutorial about how to use ObPwd, and guidelines on choosing appropriate files for creating passwords. Guidelines included warnings about dynamic files (i.e., these would create different passwords), including files they changed themselves and through updates (including system updates). They were also reminded to ensure the file was accessible at the time of login, in order to re-create their password. They were also asked to consider other people's access to their password-object file: others with access to that file, or a copy of that file, could re-create that password and potentially break into user accounts. (The specific example cited was a profile

[4] According to StatCounter (`gs.statcounter.com`), Firefox's worldwide market-share is 32%, as of Feb. 2, 2010.

picture on Facebook.) Participants were also reminded that they could use many different kinds of files, including photos, music, documents, and movies.

Participants then completed two practice tasks to familiarize themselves with ObPwd and with the experimental tasks (data from these tasks was not included in our analysis). Then the principal experimental trials began. Each trial was conducted on all 4 websites in each lab session. Websites were logged into in random order. Mimicking the commonly-given advice to avoid re-using passwords, participants were asked to use a different password object file for each website, although this was not enforced. (Any password re-use was detected, however.) Participants were also informed that they were permitted to record password information if desired, including hints to help them remember the file, or the generated passwords, on paper or in a file on their laptop. However, they were requested to not use Firefox's password manager to record passwords, as this would preclude use of ObPwd itself for entering passwords. (During the practice session, participants disabled the Firefox option of remembering passwords for sites on our study domain.)
The trials were in two phases. The first had six parts:

1. *Create*: Participants registered on a website when visiting it for the first time. They were given a unique 8-character username, which they typed into the username field. They then created a password from their own files, using ObPwd, by right-clicking on the web page. The password was entered into the password field (automatically, or through copy-paste), then entered again for verification. The participant then clicked on "register" to complete the process.
2. *Confirm*: Participants entered their username and password on the "Login" page of the site, to confirm that they could successfully log in. If login was unsuccessful, they were allowed to attempt login as often as desired, and could reset their password if needed. After a successful confirmation, they logged out.
3. *Distraction*: Participants were asked to browse their file system seeking files fulfilling specific criteria (e.g., filenames starting with a specific letter, or time of last modification in a specific date range.) If not found, they could stop after one minute. This task was designed to flush working memory related to their password object and simulate a longer passage of time by focusing attention on a separate task (cf. [22]).
4. *Login*: Participants attempted to log in to the site a second time, again trying as often as desired with the option of resetting their password if needed.
5. *Website information task*: After a successful login, participants answered a question requiring locating a specific piece of information on the site, e.g., a phone number, which they wrote on a sheet of paper. As above, this simulated passage of time, and gave them a chance to familiarize themselves with the simulated site that they would be revisiting later in the session; once located, they logged out.
6. *Questionnaire*: Participants answered a series of questions about the password file they selected, the ease of creating passwords and using the ObPwd tool to login, as well as their perceived likelihood of being able to successfully login with this in one week's time. They answered questions on 7-point semantic differential scales (from "very easy" to "very difficult").

After the participant performed all the above steps on all four sites, the second phase of trials began. Participants revisited the sites a second time, again in a random order. This phase had three parts: login, website information task, and questionnaire (similar to those in phase one). At the end, each was given information about the at-home component.

b) At-home component. This portion was designed to simulate realistic use of ObPwd, by including tasks outside of the controlled lab environment. Participants were asked to complete two different sets of tasks, detailed on a printed handout which they took home. One set was to choose three real-world websites, try to use ObPwd to create passwords for these and log in. For each site, they answered a short questionnaire, including questions about the site, the file chosen, the ease of using ObPwd, and any difficulties encountered. The other set of tasks was to revisit the same four sites from the lab session, and log in exactly once to each. For each login, there was a short questionnaire, asking about the ease of use of ObPwd. The logins were carried out in a random order. Participants could complete the sets of tasks in any order, possibly interweaving the real-world logins with lab site logins. They could do the tasks whenever desired, including all at once, or spread across the week, but completed before returning for the second lab session.

c) Lab Session 2. Participants returned for a second lab session 7–10 days after their first. This session consisted of four parts. In Part 1, they re-visited the four sites from the first session (presented in a random order), tried to log in, and answered a short questionnaire. Part 2 identically repeated the steps from the first session, on four new websites (to allow us to compare attitudes and behaviors across two sessions). Part 3, as in the first session, consisted of a second round of logins on the new sites (introduced in this second session). In Part 4, participants answered a comprehensive questionnaire about their experiences with, and attitudes towards ObPwd. This included questions about managing multiple passwords, notable positive or negative aspects of ObPwd, and scales to measure several usability factors. Because ObPwd can also be used for secondary authentication, such as through Personal Verification Questions (PVQs), a series of questions on PVQs was also included.

d) Post-study questionnaire. Participants could opt-in to a post-study questionnaire, sent approximately four weeks after their first lab session. Those who opted in were asked, by email, about their continued use of ObPwd (if any) after the completion of the formal study.

4 User Study Results

As customary in HCI studies, we first present the results, followed by a separate interpretation of the results in Section 5. The results consist of descriptive statistics; inferential statistical analysis was not performed, as in this study, ObPwd was not formally evaluated against an alternative authentication scheme.

a) Password creation times. The time taken to create a password was calculated from the time that the Registration page request was received at the web server, to the time that the "register" request (from a button on that Registration page) was recorded in the website database. This time includes typing an 8-character username, selecting a file using ObPwd, and entering that password twice (in two password fields). Participants who used the copy-paste option of ObPwd could paste the password twice in succession, but those who used the auto-paste had to recreate the password a second time. (As passwords were masked with asterisks on the page, users could not copy them directly from that field.) The times are shown in Table 1.

	Mean	Median	Std. Dev.
Sites: Lab Session 1	52.3	46.0	32.1
Sites: Lab Session 2	35.6	29.1	21.6

Table 1: Time (in seconds) taken to create a password

b) Login times. The time taken to login was calculated from the time that the Login page request was received at the web server, to the time that the participant successfully logged into the site. This time is cumulative: it includes time taken for any failed attempts, until the point when a successful login occurs. It includes the typing of an 8-character username, entering the password (e.g., using ObPwd to locate the file and re-create the password) and clicking on the "Login" button on the web page. The sites in Week 1 were logged into on five occasions: when confirming the password (initial login); twice in Lab Session 1; at home; and revisiting during Lab Session 2. The sites in Week 2 were logged into on three occasions: confirming, and twice in Lab Session 2. The times are shown in Table 2.

Sites	Action	Mean	Median	Std. Dev.
Lab Sess. 1	Confirm	18.5	15.5	11.0
	Login #1	18.4	15.4	12.6
	Login #2	26.3	20.1	25.8
	At-home login	43.4	24.2	61.5
	Sess 2 login	33.3	25.1	39.9
Lab Sess. 2	Confirm	14.5	13.1	7.4
	Login #1	21.8	14.6	24.2
	Login #2	19.3	16.1	12.9

Table 2: Time (in seconds) taken to log in

c) Login success rate and errors. A login attempt was any instance when the user clicked on the "Login" button, whether or not that attempt led to a successful login. In Lab Session 1, with 32 participants and

4 websites, the optimal number of logins would be 128 (i.e., each person four times). The actual number of attempts differed from this for two reasons: (i) any failed attempt added to the total; and (ii) some participants neglected to log out between steps of the trial; e.g., they might log in and fail to log out before the distraction task, then complete the web information task without logging in a second time. For each such instance, one login attempt would be missing. Because the password hashes were stored, we could track when a participant tried using a password created for a different account (a "password mismatch" error). Participants could also attempt to log in as often as desired; we report here the percentage of login attempts successful on the first try, and those that were successful within three attempts (allowed on many sites). Results shown in Table 3 include the total number of password mismatch errors across all login attempts, and the total number of times participants reset their passwords during the entire study.

	% Success 1st attempt	% Success within 3 attempts	# Passwd mismatch errors	# Passwd resets
Sess. 1 sites	65	90	42	6
Sess. 2 sites	93	99	14	2

Table 3: Login success rate and errors

d) Files chosen for passwords. Participants identified the type of file they chose for each password (e.g., music file, document). These file types were aggregated into categories, as listed in Table 4. The total number of possible passwords generated in each of the two lab sessions was 128 (4 sites $\times$ 32 participants). A few people chose to use files that could be updated without user intervention, such as .dlls, despite being informed during the tutorial that such dynamic files could lead to unusable passwords. In Lab Session 1, 50% of participants chose the same file type (such as photos) for all four websites; in Lab Session 2, 69%. For the at-home component with real-world websites, the total number of possible passwords was 96 (3 sites $\times$ 32 participants); 41% of participants used the same file type for all three sites. A Chi-Squared test shows no evidence that the categorizations across all three conditions (two lab and one at-home) are different with statistical significance ($\chi^2(8) = 15.1617, p > 0.05$).

File type	Lab Session 1	Lab Session 2	At-home
Image	58	58	40
Document	30	36	16
Music/audio	23	30	26
Video/movie	12	3	9
Other	3 applications 1 app shortcut 1 Windows DLL	1 Windows DLL	3 DB files 2 unspecified

Table 4: Types of files used to create passwords

Presented with a list of reasons for choosing a specific file, (e.g., "It was easy for me to find the file on my hard drive"), participants chose all reasons that factored into their decision, and the most important reason influencing their choice. Results shown in Table 5 give the overall frequency each reason was selected, for each phase of the experiment, and the reasons most often chosen as most important.

e) Memorability and recording passwords. Participants were given the option of recording password information, including hints; they were asked whether they recorded such information and whether or not they used this on logins. In Session 1, during four trials, 16 participants (50%) recorded password infor-

	Lab Session 1		Lab Session 2		At-Home	
	% selected overall	% selected as most important	% selected overall	% selected as most important	% selected overall	% selected as most important
Easy to remember which file I picked	19.9	32.5	20.0	24.2	19.0	28.4
File is related to website (can associate)	6.4	16.3	8.6	14.8	10.9	23.2
If someone has access to laptop, can't easily guess file	14.9	15.4	14.8	14.8	15.2	9.5
File won't change	19.5	14.6	19.6	21.1	19.5	14.7
Easy to find on my hard drive	16.7	9.8	19.2	13.3	16.0	13.7
No one else has access to file	7.0	4.1	6.5	2.3	7.9	5.3
I will have access when need to log in	14.1	3.3	10.2	3.9	10.7	2.1
Other	1.4	4.1	1.2	5.5	0.8	3.2

Table 5: Reasons for selecting file for creating password

mation (primarily hints) at least once; the others did not record any memory aids. In Session 2, 18 (56%) did, at least once, while the remaining 14 did not record anything. For the at-home portion, half recorded password information, and half did not. The most common type of recording, in all sessions, was writing password hints on paper. In responses to the background questionnaire, which asked about normal password management behavior, half of the participants indicated that they did (at least occasionally) write passwords down; three used a password manager, and 75% used their browser to remember passwords for at least a few websites. At the end of each trial, participants rated how likely they thought it was that they would be able to log in successfully a week later, for each website and password, on a scale of 1–7 (with 7 "highly likely"). The ratings were 6.40 (Lab Session 1), 6.30 (Lab Session 2), and 6.24 for the real-world websites in the at-home component.

f) Password reuse. Participants were requested to refrain from reusing password objects, as we wanted to observe how users cope with multiple ObPwd passwords. While we did not block reuse, we detected any occurrences of reuse across sites. 15.6% of passwords were reused (40 out of 256). Participants also self-reported whether they had reused any passwords; 9 of the 13 who were detected as reusing passwords recalled having done so for study sites.

g) Password visibility. Because ObPwd could be used with either manual or automatic pasting of passwords, participants were asked whether they looked at the generated password (only possible with manual paste) or used ObPwd to paste the password into the field without the intermediate step. In Session 1, six participants (19%) looked at the password when it was first created in at least one trial; in Session 2, four (13%) did so. When the passwords were used later to log in, none looked at the password: all logged in using the automatic paste feature.

h) Perceived usability. After the password creation phase, a 7-point rating scale was provided for participants to indicate their perceived level of usability for four factors: ease of thinking of a file to select for the password; ease of locating the file chosen; ease of creating the password using ObPwd; and ease of logging into the website with the chosen password. 1 represented "not at all easy" and 7 "very easy." Table 6 reports results. The majority of ratings had a median of 7; the lowest was the difficulty of choosing an object file, with a mean of 5.98.

	Lab Session 1		Lab Session 2	
	Mean	Median	Mean	Median
Choose file	5.98	6.0	6.05	6.0
Locate file	6.20	7.0	6.52	7.0
Create password	6.77	7.0	6.83	7.0
Log in	6.73	7.0	6.63	7.0

Table 6: Perceived usability for password creation tasks

i) Real-world usage. Participants provided comments about their experiences using ObPwd during the at-home component, on three real-world websites. They indicated the kinds of websites they used ObPwd to create passwords for, which were wide-ranging and included video-hosting sites, blogs, webmail, news and sports sites, and social networks. They were asked to report on any problems encountered; four instances were reported (of a total of 96 sites). The first was a problem caused by changing a password: a participant who changed their Hotmail password had trouble using MSN messenger because they did not know that these two accounts used linked passwords (through Windows Live), and thus passwords for both accounts were changed. A second person had problems with a Flash login screen, which prevented them from right-clicking and launching ObPwd. A third had problems with a video file, stating that the process was slow for creating a password; investigation suggested a problem with an API in specific versions of Firefox, now resolved in later versions. Finally, when one participant used the generated password on a blog site, it failed the requirements for that site. (Possibly that site's password rules required non-alphanumeric characters, not supported by the version of ObPwd used.)

j) Overall user experience. Participants provided feedback about their overall experience with ObPwd through a questionnaire, which included a series of seven-point Likert scales (1="strongly agree," 7="strongly disagree."). Results are listed in Figure 5, which presents boxplots. (In boxplots, the heavy vertical bar is median, the box shows the two central quartiles and the dashed line shows outer quartiles.)

Participants also described any aspects of ObPwd that they particularly liked or disliked. The most commonly-cited disadvantages were: concerns over being able to use ObPwd on other computers; concerns over losing passwords when files changed; not being able to use ObPwd with other browsers; and time to log in. Advantages cited most frequently were the automatic pasting of the password; the wide selection of passwords; the ease of creating and remembering passwords; and increased perceived security. Some illustrative user comments describing the positive aspects were: "My password was a picture! That was cool."; "It was easy, convenient, and good for security."; "Easy to create, secure, only have to remember a file, not how it's spelled (i.e., uppercase or numbers after)."

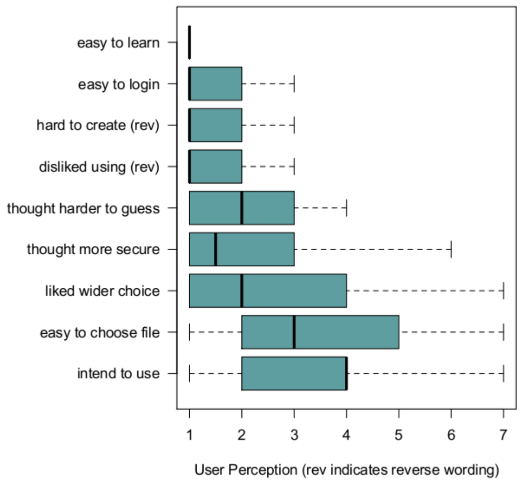


Figure 5: Perceived usability of ObPwd (Likert-scale)

k) Personal Verification Questions (PVQs). One potential use for ObPwd is for PVQs, which could allow a user to input her ObPwd password as a difficult-to-guess response for free-format PVQs. Although this use was not explicitly evaluated in this user study, participants were asked about their experiences with PVQs, and to predict their likelihood of using ObPwd for PVQs. All participants knew what PVQs were: 92% had experience setting PVQs on websites; 84% used them for password resets, and 54% for logins to sites. 44% logged into sites with PVQs using other people's computers (rather than their own) at least occasionally; 76% had created their own question for a PVQ. 52% predicted they would use ObPwd for future PVQs; on a scale from 1–7 (7 = "highly likely" usage of ObPwd), the median rating was 6. Another 32% of participants were unsure if they would use ObPwd for PVQs, but thought this might occur in some circumstances; these stated that it would depend on such factors as the security level of the web-based account, the availability of ObPwd and their password on the machine being used to log in, and the ability to create their own PVQ on a specific web site. In total, 84% of participants either expected to use ObPwd for PVQs, or could conceive of situations in which ObPwd could be so used.

l) Post-study usage. At the end of the study, we offered to uninstall the ObPwd extension, if participants wanted us to do so; of 32 participants, only one accepted this offer. The others kept the extension. Participants could also opt-in to a post-study questionnaire: 26 of 32 (81%) opted-in; questionnaires were emailed to these 26, and 16 responded. Out of these 16, 6 (38%) had used the ObPwd extension after the study, and 7 (44%) had recommended ObPwd to someone, such as a colleague or friend.

5 Interpretation of Results

a) Login times and errors. Over 90% of logins were successful within three attempts; by Lab Session 2, over 90% of were successful on the first attempt. However, the mean login time was 19 seconds (at the end of Lab Session 2). To place this time in context, we compare it to Forget et al. [8], who reported login times for 8-character text passwords as well as Persuasive Text Passwords (PTPs, described in Section 7). In the PTP variant that provided the optimal combination of security and usability, the login time was 17.1 sec.; in this variant, users must memorize two extra characters in addition to their password. For the ordinary 8-character passwords, the mean login time was 11.4 sec. Thus, the ObPwd login task took approximately twice as long as text passwords, and slightly longer than PTPs. A few participants commented on the longer time required, although overall, participants gave high ratings to the ease of logging in with ObPwd (see below).

In multi-account scenarios participants may try one or more wrong passwords before a correct one, thus increasing overall login time. When such multiple password interference is considered, ObPwd login times appear comparable or even slightly better than regular text password and the PassPoints graphical password schemes, as reported by Chiasson et al. [2]; see Table 7. Here *Text*, *PP*, and *OP* represent text password, PassPoints, and ObPwd respectively; *Recall-1* is login in the first test session (Login #2 in Lab Session 1 for ObPwd – see Table 2), and *Recall-2* is login to the accounts created in the first session after more than a week (7-10 days in ObPwd, and 12-15 days in Chiasson et al. [2]). Additionally, ObPwd login times include network delays (client-server communication time). Figure 6, which was generated from a distribution provided by Chiasson et al., shows login times for PassPoints as well as for the final ObPwd login in Session 2 — the stage at which the participants were most familiar with ObPwd. The login success rates for Session 2 of the ObPwd study were much higher ($> 90\%$, see Table 3) than those reported [2] for both text passwords and PassPoints (59% and 57% respectively, within three attempts, after 12-15 days). This suggests that ObPwd may provide advantages for password memorability over time; however, the experimental designs of these two studies varied too widely to make their data strictly comparable.

	Mean			Median		
	OP	Text	PP	OP	Text	PP
Recall-1	26.3	29.3	15.1	20.1	14.0	11.8
Recall-2	33.3	42.1	47.0	25.1	26.8	32.6

Table 7: Login times (sec) in ObPwd, text, and PassPoints

b) Files chosen and reasons for choice. The most commonly-chosen file type was images: in the lab sessions, almost half of files were some type of graphical file. The file types were constrained to objects the user had available at the time; for people with large photo or music collections, those file types are a natural source of password files. For some participants, the visual characteristics of images allowed them to associate the password file with a specific study website: e.g., one participant stated that for a site with a yellow background, he chose a photo of his girlfriend in a yellow dress. A wider variety of files was used for the real-world sites than for the lab sessions. This may be due to a wider range of websites being available, which gives a greater number of possibilities for locating files suitably associated with those sites. One participant used a journalism course document for a study guide site, and music for an online classified site: "[a] song that reminds me of my mom, who uses [that site]." When identifying the reasons for file choices, memorability was a key issue: the most commonly-cited factor influencing the choice was that it was easy to remember which file was picked, followed by the ability to associate the file with a specific website.

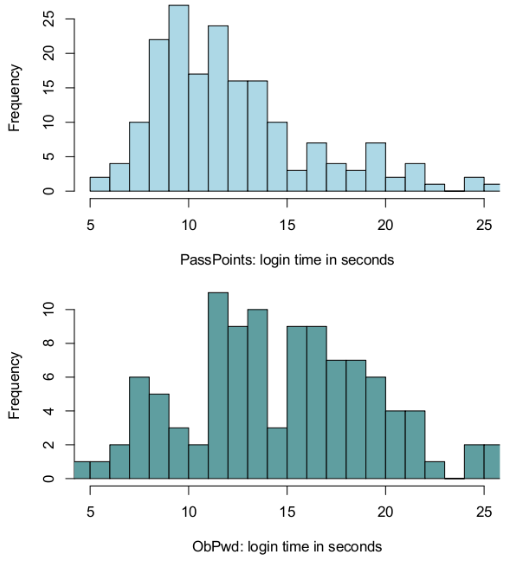


Figure 6: Distributions of login times for PassPoints [2] and ObPwd

c) Memorability. Participants were almost always able to log in successfully, although they did sometimes try to log in with a password from a different account. We asked them what tricks (if any) they used to keep track of which passwords were used on which sites. Visual cues, as mentioned above, were described, e.g., color matching between photos and websites, or a similar object pictured in both the photo and the site. Another participant described associating music titles with websites: "...insurance [site] (if you get hurt or die), I linked it with a song, 'Knocking on Heaven's Door'." Other participants used files from the same folder every time; this is facilitated when using the same media type repeatedly, which are often filed together within one folder. (Using the same folder simplifies both recall and locating the password file, as users need not browse through the file system to find a file in a separate location.) Others used letters to link sites and password files, choosing existing filenames that started with the same letter as the website's title. Participants who recorded password information, such as hints on paper, were able to use those hints instead of relying on mental associations to choose the right file.

d) Password reuse. As evident from Figure 7, password reuse was limited,[5] even though each user was dealing with at least 8 ObPwd passwords with a high ($> 90\%$) login success rate. This may be an indication that ObPwd may reduce the effect of multiple password interference. (However, we do not know whether ObPwd will reduce password reuse in practice.) In contrast, most proposed new authentication schemes are not tested under interference, even though password interference can lead to password reuse and choosing passwords as minor variants. We also found one password hash common among three individuals (but no other two-way collisions). Information gathered from participants suggests that the most likely cause was three participants selecting the same sample music installed with the OS; this likelihood may have been increased as such sample music was used as an example ObPwd password object during demo. Proactive blacklisting by ObPwd, of popular pre-installed files, may thus be prudent; alternatively, the use of ObPwd Variant 1 (Section 2.1) would address such collisions.

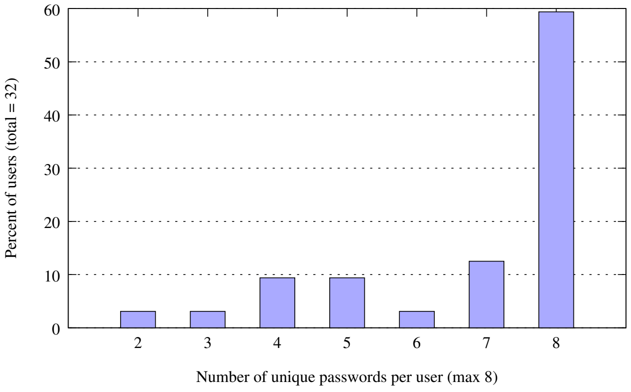


Figure 7: Password reuse

e) Password visibility. By the end of the study, all participants used the automatic paste feature for entering passwords, instead of copy-paste, thus did not view the generated password. Participants did not provide reasons for this preference; we speculate that this could be due to convenience (skipping the copy-paste step), or because the password itself is of limited perceived utility, given that it need not be typed in directly by the user. This finding is in contrast to that in a password manager study [3], in which participants were frustrated when they were unable to view their passwords; with ObPwd, this option is always available, even if it is not often used, which may give users a sense of control.

f) Perceived usability and user experience. Participants rated ObPwd highly on the four usability factors measured, including ease of login — despite the relatively long login times. Logins were generally successful within a few attempts, likely contributing to the high rating. However, there also appears to be a positive *affective* component to ObPwd: affect is the influence of emotional involvement, widely regarded as part of broader strategies in interaction design to promote user engagement and greater cognitive involvement [23]. Another website login tool, Passpets [40], leverages affect by giving users their own "pet" as an anti-phishing strategy. For those using ObPwd, although the task of finding an appropriate password-object file may take some time, it is not perceived as onerous. Instead, people appear to enjoy interacting with personally-significant objects; this is a pleasurable activity, and thus not perceived as burdensome, as illustrated by two examples from participant comments. One described why she had chosen a photo for a social networking site: "[it's my] friend's chain, whom I met on [this social networking site], who passed away." That photograph has a strong emotional connection for her. Another used an audio file containing music that he himself had composed; again, this file is deeply personal and has significance for that individual. Other participants mentioned photos of friends or family, as well as their favorite music. These objects are pleasant to interact with as well as memorable. ObPwd users have a positive affective experience when they create and use passwords.

ObPwd was rated highly on ease of learning, and ease of choosing files to use as passwords; participants disagreed with questionnaire assertions that ObPwd made it hard to log in or that they disliked using it.

[5] According to a large-scale study [6], each web password is shared across 3.9 different sites, with 25 accounts on average per user (cf. 1.2 sites/password in our test for 8 accounts).

Participants also thought that passwords created with ObPwd were harder to guess, and were more secure than, their usual text passwords, and that they had a wider selection of passwords to choose from. We asked if ObPwd would be adopted for use beyond the study. The Likert-scale answers were unclear on this point, with a mean rating of 3.44: this value tends towards agreement, but not strongly. For clarification, we asked about usage in the post-study questionnaire: 38% did use the extension once the study was completed, demonstrating that participants did find value in the tool outside of the study environment.

g) Summary of findings. Our user study demonstrated that ObPwd could be effectively used by our participants, with a high level of satisfaction. The login times were comparable to those of one of the faster graphical password schemes, and were acceptable to our participants. The login success rates were high, with over 90% of logins successful on the first attempt, during the second lab session. Through the study, we gained an understanding of the types of files users might select as password objects; the choices ranged across a number of types, with a majority being images. Participants also gave high ratings to many usability factors of ObPwd, including ease of selecting and locating password objects, as well as ease of creating passwords from the extension and ease of logging in to websites. ObPwd also demonstrated positive affect, with participants enjoying working with their password objects when creating password and logging in. An affective component can aid both user engagement with the password scheme, and can also facilitate memory. Post-study usage of the tool suggests that ObPwd provided value to our participants in their everyday tasks, many of whom chose to continue using the extension after the study was completed. Thus, ObPwd was shown to be usable and demonstrated a solid user acceptance.

6 Security Analysis

We provide a practical security analysis of ObPwd, and consider attacks, and other risks.

6.1 Entropy Estimation

Length constraints on password objects. ObPwd uses at most the first $n = 100,000$ bytes from an object, for three reasons: to reduce file or URL read/download time; to reduce hashing time; and to capture sufficient entropy (as some file types may have a large, e.g., few hundred-byte header structure with limited variation). We assume this limit is certainly sufficient to obtain 160 bits (recall that SHA-1 is used) of entropy from most user-chosen file/URL objects. The minimum object size of $m = 160$ bytes is designed to take into consideration user-selected text blocks that may not be rich in entropy. Under the pessimistic assumption that on average, each input byte of a selected text string provides at least one bit of entropy, this value of m provides entropy appropriate to the 160-bit output of SHA-1, to be used as input to Hash2Text.

Entropy metrics. Consider the threat of a brute-force guessing attack. Assume ObPwd passwords are l characters long, each independent and equi-probable from an alphabet of b characters, as consistent with the paragraph above. Then the *Shannon entropy* in bits is given by $H = l \cdot \log_2(b)$, and serves as an upper bound measure of security. Another metric is *min-entropy* [32]; informally, this measures the difficulty of guessing a password for any of a number of (un-targeted) accounts. Under the above assumption, implying ObPwd passwords themselves are equi-probable, their Shannon entropy and min-entropy are equal. A further metric, appropriate for a targeted attack on a selected user account, is *guessing entropy* G [17, 32] (see also Pliam's related discussion of *marginal guesswork* [24]): the expected number of guesses for a correct guess, ordering candidate guesses from most to least probable. Formally, $G = \sum_{i=1}^{K} i \cdot p_i$, where the elements $P_i, 1 \leq i \leq K$, in the password space are indexed in non-increasing order of their probabilities p_i. To execute the optimal strategy assumed by the guessing entropy formula, the attacker must have perfect knowledge of the probability distribution of passwords in the system. For the case of K equi-probable passwords, the formula simplifies easily to $G = (K + 1)/2$ guesses, in any order.

Entropy of ObPwd password (default settings). For the default alphanumeric character set (mixed-case letters plus digits), the SHA-1 hash output of a password object is mapped into the 62-character set. As above, approximating the entropy of user-selected objects to a full 160 bits matching SHA-1, we model each character in the output password as equi-probable and independent — that is, for practical purposes, such ObPwd passwords are random (non-redundant). This assumption is justified under the model in which an attacker, having no access to a password object (or its hash), has no attack better than to guess random strings from the character set. In this case, for an ObPwd password ($l = 12$, default character set), $H = l \cdot \log_2(62) = 71.45$ bits and $G = 2^{70.45}$. This model is reasonable for users choosing personally created objects that remain private, e.g., unshared photos or once-public photos locally altered (but not, e.g., for common public objects, such as shared images and popular soundtracks).

For a rough comparison, by NIST's empirically-derived password entropy estimate [1], assuming a 94-character alphabet (common printable characters excluding space), a 12-character user-chosen text password has roughly 24 bits of entropy: 4 bits for the first character, 2 each for the next 7, and 1.5 each for the last 4. If users must include both upper case and special characters, this rises to 30 bits (cf. 71.45 above). ObPwd entropy in bits increases linearly with pwd length (e.g., to 119 bits for length $l = 20$), with virtually no usability impact as users need not memorize the generated passwords. Expanding the ObPwd alphabet set (e.g., from the default 62 to 94 characters) increases entropy further, but many websites require alphanumeric-only characters.

Case of special characters allowed. If default settings are changed to insert special characters (as required by some sites), then beyond the default set of 62, ObPwd allows '+' and '/' (but no others, as some sites prohibit '?', '%', space, and other special characters). If a generated password contains '+' or '/', no further characters are added (password-space size: 64^{12}). Otherwise, '+' is inserted as character 12, and the password is rotated deterministically based on other bits of the SHA-1 output (password-space size: $62^{11} \times 12$). In the former case, $H = 72$ bits and $G = 2^{71}$; in the latter case, $H = 69.08$ bits and $G = 2^{68.08}$. Counter-productively here, the inclusion of a special character minorly reduces the entropy. The requirement of special characters by some websites, while others prohibit them, complicates the design and use of ObPwd. Rather than changing the default setting when a site requires a special character, an option perhaps impacting usability less is to recommend that users append a fixed special character of their choice to their ObPwd pwd when required (e.g., convert "R8aqV8sGZnDI" to "R8aqV8sGZnDI@"). The extra character is to meet password rules (vs. increase entropy); using a fixed character mitigates cognitive load.

6.2 Attacks on ObPwd and Possible Risks

Below we discuss attacks on ObPwd and other risks as may arise from a large scale adoption of this scheme.

a) Malware and guessing attacks. As for text passwords, ObPwd passwords/objects are vulnerable if the user platform/device is compromised. (Regarding phishing and keylogging, see Variants 3 and 4 in Section 2.1.) However, when password guessing attacks are used to compromise a system (e.g., SSH guessing attacks [26]) or spread to other systems (recall the Morris worm [29]), strong passwords as generated in ObPwd may delay or prevent the compromise.

b) Network-based/man-in-the-middle attacks. If ObPwd is used in regular web login, we strongly recommend that the password objects be stored in local media when passwords are generated on-the-fly (right before login) from public web objects. If a password is re-created from plaintext web content the following attack is possible. An attacker observes or records traffic from the intermediate network (e.g., a wireless access point, web proxy) looking for a user entering a content-hosting site right after or before requesting an authenticating website, thus capturing or narrowing down candidates for the password-generating content. In contrast, when ObPwd is used for encryption/decryption in a user's local media, network-based password objects do not allow access to protected content (a network attacker does not have access to a user's local encrypted files). Of course if the attacker already controls the user PC, no regular password schemes are safe.

c) Building an attack dictionary. A global password attack dictionary might be built as follows. Assume users who choose to use publicly available objects will do so mostly from highly popular websites. Photos, comments and other information from social networking and photo-sharing sites can be crawled regularly to harvest publicly available password objects. Such sites when compromised can also expose online private objects. Many search engine providers maintain an updated archive of the public Internet and are also in a vantage point to observe user choices (e.g., sites most visited for a certain search term). They may create dictionaries of global password objects, perhaps orders of magnitude larger than current text dictionaries. (While users may choose objects from a small subset of sites, some popular sites are quite large: as of Apr. 30 2009, Facebook stores more than 60 billion photos; each of over 15 billion user-uploaded images is saved in four different sizes.[6])

To build a custom dictionary on a target user, a user's network provider (ISP, wireless access point), proxy sites, certain nodes in anonymous browsing services (Tor exit nodes), or parties who can monitor the user's traffic may gather available data from their personal site, social-networking sites, frequently-visited sites, etc. To access private objects, a dictionary builder may seek to breach private storage (e.g., by malware infection of user machines).

d) Shoulder surfing. For the ObPwd Firefox extension, the generated password is directly placed into a password field if the extension is activated from the right-click context menu of the password field. In other cases (e.g., when right-clicked on other areas of a web page, or using the stand-alone Windows/Mac applications), the generated password is displayed on screen, unmasked, facilitating shoulder surfing especially if the password dialog box is open for an extended period of time or to attackers using a camera or videophone. Another risk during password generation is observing a user's object selection when the object is public (less so for private objects, due to an attacker's lack of object access).

e) Sharing favorite images. Despite recommendations to choose private objects (e.g., personal photos) not publicly shared, users may use in ObPwd precisely the favorite photos they are most likely to share with others. Such risks would be reduced if users posted a modified (e.g., resized) image on the public web, using the original for ObPwd in local media, but such an expectation fails usability goals.

f) Password update, mobility, and lost/stolen media. Password renewal with ObPwd is the same as for current text passwords. Using multiple computers (e.g., home/work PC, laptop) for login requires users have ready and constant access to password objects from multiple platforms. While not an issue for users who regularly carry laptops and mobile devices holding large collections of personal objects, for others, private objects may be used from mobile media such as USB storage. The use of protected online objects (e.g., email text) may also be preferred when mobility is critical. Losing password objects (e.g., lost media, accidental deletion) is equivalent to forgetting a password; users may resort to existing password recovery/reset mechanisms. Users may favor objects of special significance which they may already keep multiple copies of as backup (photos/videos of favorite trips, weddings, celebrations); this can reduce risks from lost or inaccessible media.

g) Folder containing all password objects. Users may choose to put all their password objects in one folder, easily available to family, friends, and attackers with momentary access (and simplifying malware attacks). For context with existing passwords in browsers, anyone can see plaintext passwords from Firefox's built-in password manager when (commonly) the manager is not protected by a master password. In both cases, access to a user PC allows installation of malware including a rootkit.

h) Risk of object modifications. ObPwd passwords depend on the first $n = 100,000$ bytes of a selected object. Modifications to content (updating document files, editing image files) preclude re-creating the password, unless the original object or generated password is backed up. Users updating metadata embedded in media files (e.g., ID3[7] tags) may also be problematic; such metadata is generally stored at the beginning or end of media files. To address this, ObPwd might be modified to drop (e.g., 1000) bytes from each end of such files.

[6]`http://www.facebook.com/note.php?note_id=76191543919&ref=mf`

[7]IDentify an MP3 (ID3), specifications available at: `id3.org`

7 Related Work, Comparison, and Features

Of countless publications on passwords, here we mention only a selective subset involving schemes designed to strengthen passwords (entropy) or enhance usability. We also summarize and discuss ObPwd features.

7.1 Schemes for Improving Password Strength/Usability

To improve password strength while maintaining usability, Forget et al. [8] proposed Persuasive Text Passwords (PTP) wherein system-generated characters are inserted at random positions into a user-chosen initial text password. Users can accept the proposed password, or request (until satisfied) alternative suggestions.

Florêncio et al. [7] argue that relatively weak passwords (e.g., with 20 bits of entropy) may provide enough security for web accounts assuming a "three-strike" rule (i.e., login blocking after failed attempts to counter brute-force attacks), a sparsely-populated user ID space, and valid user IDs not being readily available to attackers. Meeting these assumptions requires assistance from authenticating sites.

A user study by Yan et al. [39] compares regular user-chosen passwords, random passwords and mnemonic phrases. They found that mnemonic phrases are as good as random passwords, and easier to remember. However, passphrases (and mnemonic passwords generated from them) may also be attacked by building a dictionary from phrases available on the web [14].

Disk encryption software TrueCrypt allows users to optionally use any file from their local system or certain smart cards along with a possibly empty/weak password for generating keys for TrueCrypt encryption of disk volumes.[8] This feature is aimed at improving encryption keys which may be brute-forced if derived only from user-chosen passwords. ObPwd was conceived independently and is focused on creating strong text passwords from user-chosen content: these can be used to derive strong encryption keys, but also more generally for web logins. Users also cannot write down (for backup) the actual TrueCrypt encryption key.

Gibson et al. [9] proposed *Musipass*, an authentication technique relying on the universality of music and human ability to remember/recognize music. During enrollment, users are presented with nine icons corresponding to sound clips, from a set of pre-defined clips, listen to one or more of the presented clips, and choose one as part of their password. Four clips (from four such rounds) make up the user's password. For login, users select their clips from among decoy clips. A user study with 133 people (94 in phase two) showed promising usability. As reported, deployment of Musipass requires server-side changes. ObPwd allows users to choose music (or other digital content) files of their own (i.e., not system chosen), and uses these files in a different manner (e.g., listening is not required).

7.2 Personal Verification Questions (PVQs)

PVQs are used for resetting a forgotten password or as part of login. The availability of personal information on the web has made it easier to correctly guess PVQ answers [25, 10]; real-world attacks exploiting PVQs are also quite common (e.g., [35]). Decades apart user studies [41, 28] confirm that PVQs are highly susceptible to guessing attacks.

Rabkin [25] analyzed over 200 PVQs as used in 17 financial websites. Taking the 'era of Facebook' into account, different classes of attacks are considered (e.g., random guessing, attacks automatically using online information, dedicated human attackers, and knowledge through personal acquaintance). One defense suggested involves use of personal content: a user uploads an image of a person, and an answer to the question "What is the name of this individual?" However, as noted, any tagged photo of that person provides attackers an answer.

Schechter et al. [28] conducted a 130-participant user study of the security and memorability of personal questions as used by four popular online email services. Participants were asked to provide PVQ answers. Acquaintances (e.g., spouse, fiance, friend, co-worker) correctly guessed 22% of their answers, overall. Using the most popular answers of other participants (i.e., *statistical guessing*), 13% of answers could be

[8]This feature is apparently available since version 4.0 (Nov. 2005); see `www.truecrypt.org/docs/keyfiles`.

guessed correctly within five attempts. Measures suggested to defend against statistical guessing were: (i) penalize/disallow certain answers in proportion to their popularity; and (ii) remove questions that are currently statistically guessable over 10% of the time. Such measures, however, may decrease users' PVQ success rates.

7.3 Summary of ObPwd Features and Discussion

a) User-chosen strong passwords for ordinary users. Humans are inherently pattern-oriented, and most user-chosen passwords are weak: users are unable to create a string that is high in entropy but memorable over a long period. ObPwd is designed to ease this burden, offering the advantages of both system-generated (high-entropy) and user-chosen (easy to remember) passwords, without their disadvantages (respectively: hard to remember, easy to guess). Users get strong passwords (i.e., which are resistant to dictionary attacks) simply by choosing personally meaningful photos or other digital objects; and these passwords, while easily re-generated (by selecting the same objects), are not vulnerable to traditional guessing attacks.

b) Coarser memory tasks. ObPwd may be categorized as a *recall-based* scheme when users remember the location of their password object (e.g., a file's directory or URL). However, users not recalling the exact location or object may browse through objects (e.g., looking at possible images or thumbnails), creating more of either a *recognition-based* or *cued-recall* scheme. Regardless, ObPwd does not require exact password recall; instead, users must be able to locate their password objects — an activity involving less detailed memory than regular text or graphical password recall.

c) Avoids complex password rules. The concept of strong passwords is poorly understood. The usability benefits of an easy-to-remember weak password are obvious to users, but motivation for a strong password is less intuitive. Password generation in ObPwd appears less technical conceptually: users choose private or personally-meaningful digital objects to access their accounts, without being subjected to arbitrarily complex password rules about length, uppercase and special characters. Instead, stronger passwords are an inherent by-product of the underlying design.

d) Engagement promotes user acceptance. The mundane technical task of creating and recalling text passwords is replaced by selecting and recalling objects (e.g., personal photos) that are more both familiar and reportedly satisfying to users. This provides a positive affective experience, leading to strong user engagement with ObPwd. This engagement fosters user acceptance (see Section 4), which compensates for somewhat longer ObPwd login times than for text passwords.

e) Password backup and secure sharing. Backup and sharing of passwords (heavily relied upon in real-world usage [21]) is easy in ObPwd — e.g., writing down a 12-character alphanumeric generated password — whereas in most graphical password schemes it is awkward at best. ObPwd may also enable better password sharing than text schemes without sacrificing confidentiality to third parties — e.g., if two users pre-share digital photos (say through personal media), one can choose a specific image as the password object, and send the other a hint or description (e.g., "our whitewater kayaking photo") over public media or email. An eavesdropper seeing the hint cannot generate the shared password without access to the object itself, assuming the hint is not an obvious link to a publicly-accessible object. This form of *user-friendly codebook*, using meaningful objects, has advantages over sharing a list of randomly generated secret keys.

f) ObPwd as strong graphical passwords. ObPwd provides a middle ground between text and image-based password schemes, allowing use of images while retaining simple advantages of text passwords (low cost, no system-side changes, written records). While some still claim writing down passwords is poor practice, this depends on the threat model and usage. Certainly, being able to write down and backup infrequently used passwords has advantages, while prohibiting it may encourage choice of weak passwords.

In contrast to ObPwd, most graphical password schemes [30] use system-assigned images/random art, and require server side changes. Most offer a large password space in theory, but due to bias in human selection the password space used in practice is much smaller [4, 31]. Even if thousands of users choose their own picture of the Eiffel Tower as their password object, ObPwd will generate unique strong passwords as long as they do not share identical photo objects. While highly popular online public objects may be chosen by multiple users, the vast content on the Internet and its growth make building a global password dictionary an interesting challenge (see Section 6.2).

g) No changes to server or password interfaces, including PVQs. Deployment barriers are low, requiring no system-side changes at enrollment or login, nor to client-side software interfaces as alphanumeric character strings are produced. ObPwd passwords can be used as answers to PVQs (wherever alphanumeric characters are allowed in such answers) — e.g., in answer to "What is your mother's maiden name?", use a high-entropy ObPwd password generated from a memorable private image of your mother, instead of low-entropy [28], easily guessed or easily found answers that are sometimes worse than weak passwords. Users can use the question field as a hint to their password object, especially on sites that allow free-format user-chosen questions. Such hints may enhance the retrievability of PVQ answers used very infrequently (assuming the password objects are not lost).

h) ObPwd and password managers. The randomness of ObPwd passwords raises comparison to password manager tools that generate unique random passwords for individual login sites, and manage these, encrypted under a key derived from a user-chosen master password. (ObPwd does not store user passwords.) Such tools offer the simplicity of remembering only a single master password, although its strength becomes critical, and losing it may forfeit all passwords at once. In contrast, losing ObPwd objects (unless multiply-used) is comparable to losing a single-site password. An advantage of ObPwd is its positive reception from users, possibly due to pleasant associations users have with the underlying objects they choose. In other tools such as Password Safe (`passwordsafe.sourceforge.net`) and PwdHash [27] which generate random-looking passwords from a master password and a random salt and/or an authenticating site's URL, the master password becomes a more attractive target to malware, and to offline dictionary attacks by attackers having access to the generated password and the URL (e.g., via phishing).

8 Concluding Remarks

Choosing multiple, long-term memorable, high-entropy secrets is not a basic human capability. Widely-deployed current password generation techniques and password-restricting rules have largely failed to yield strong passwords. Creating passwords from personally meaningful/memorable digital objects offers a user-friendly alternative to more complex password rules.

While our user study provided insight into the types of local files chosen as ObPwd objects, a very large-scale field study is necessary to allow empirically-based quantification of the guessability of the resulting passwords, to search for usage patterns, and to develop best practice guidelines recommending or excluding certain classes of objects. As discussed herein, ObPwd passwords can heuristically be modeled as random strings, thereby providing security far outweighing conventional text passwords, but only under assumption that attackers do not have access to the password objects, and are unable to predict the use of popular objects that are publicly available. However, such very large-scale testing is more challenging; indeed, despite existing password crack papers (e.g., [36]), the community still lacks a good understanding of the empirical security of even ordinary text passwords as chosen by the mythical typical user. Studies of even as many as 500,000 users [6] are too small for the long-tailed distribution of user-chosen passwords, and obtaining or publishing ecologically valid passwords in such studies is complicated by privacy concerns.

Our hybrid user study exhibits strong ecological validity, including, beyond the usual return-to-lab sessions, a field component wherein participants used ObPwd passwords to access real-world web sites of their choice. The user study showed positive results: acceptable login times, very good login success rates, and extraordinarily positive user perception of the experience. Participants' comments showing a strong affective experience with ObPwd indicate a likelihood of both better engagement and memory. The user study and analysis suggest a novel combination in password authentication: the positive affective aspects associated with user-choice (plus acceptable performance), without the negative of password guessability typically accompanying user-choice [4]. Of independent interest, this motivates further study of affective password schemes in general.

Acknowledgments. We thank members of Carleton Computer Security Lab for enthusiastic discussion on this topic, especially David Barrera and David Whyte. This work is supported in part by NSERC ISSNet (Internetworked Systems Security Network – an NSERC Strategic Network Grant), and the NSERC Discovery Grant Program. The fourth author is Canada Research Chair in Internet Authentication and Computer Security, and is supported in part by the Canada Research Chairs Program.

References

[1] W. E. Burr, D. F. Dodson, and W. T. Polk. Electronic authentication guidelines. NIST Special Publication 800-63, Apr. 2006.

[2] S. Chiasson, A. Forget, E. Stobert, P. van Oorschot, and R. Biddle. Multiple password interference in text and click-based graphical passwords. In *ACM CCS'09*, Chicago, IL, USA, Nov. 2009.

[3] S. Chiasson, P. van Oorschot, and R. Biddle. A usability study and critique of two password managers. In *USENIX Security*, 2006.

[4] D. Davis, F. Monrose, and M. K. Reiter. On user choice in graphical password schemes. In *USENIX Security*, San Diego, USA, Aug. 2004.

[5] K. M. Everitt, T. Bragin, J. Fogarty, and T. Kohno. A comprehensive study of frequency, interference, and training of multiple graphical passwords. In *CHI'09*, Boston, MA, USA, Apr. 2009.

[6] D. Florêncio and C. Herley. A large-scale study of web password habits. In *WWW'07*, Banff, Canada.

[7] D. Florêncio, C. Herley, and B. Coskun. Do strong web passwords accomplish anything? In *USENIX HotSec'07*, Boston, USA.

[8] A. Forget, S. Chiasson, P. van Oorschot, and R. Biddle. Improving text passwords through persuasion. In *SOUPS'08*, Pittsburgh, USA.

[9] M. Gibson, K. Renaud, M. Conrad, and C. Maple. Musipass: Authenticating me softly with "my" song. In *NSPW'09*, Oxford, UK.

[10] V. Griffith and M. Jakobsson. Messin' with Texas, deriving mother's maiden names using public records. In *Applied Cryptography and Network Security (ACNS'05)*, New York, NY, USA, June 2005.

[11] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. *ACM TISSEC*, 2(3), Aug. 1999.

[12] N. Haller. The S/KEY one-time password system. In *NDSS'94*, San Diego, CA, USA, Feb. 1994.

[13] C. Herley. So long, and no thanks for the externalities: The rational rejection of security advice by users. In *NSPW'09*, Oxford, UK.

[14] C. Kuo, S. Romanosky, and L. F. Cranor. Human selection of mnemonic phrase-based passwords. In *SOUPS'06*, Pittsburgh, USA.

[15] M. Mannan and P. van Oorschot. Digital objects as passwords. In *USENIX HotSec'08*, San Jose, CA, USA.

[16] M. Mannan and P. van Oorschot. Using a personal device to strengthen password authentication from an untrusted computer. In *Financial Cryptography (FC'07)*, Trinidad and Tobago.

[17] J. Massey. Guessing and entropy. In *IEEE Symposium on Information Theory (ISIT)*, Trondheim, Norway, 1994.

[18] J. M. McCune, A. Perrig, and M. K. Reiter. Safe passage for passwords and other sensitive data. In *NDSS'09*, San Diego, USA.

[19] R. Morris and K. Thompson. Password security: A case history. *Communications of the ACM*, 22(11), Nov. 1979.

[20] OpenWall.com. John the Ripper password cracker. `http://www.openwall.com/john/`.

[21] A. S. Patrick. Monitoring corporate password sharing using social network analysis. In *International Sunbelt Social Network Conference*, St. Pete Beach, FL, USA, Jan. 2008.

[22] L. Peterson and M. Peterson. Short-term retention of individiual verbal items. *Journal of Experimental Psychology*, 58, 1959.

[23] R. W. Picard. Affective computing. Technical report, 1995. MIT Media Lab, Perceptual Computing Group.

[24] J. O. Pliam. On the incomparability of entropy and marginal guesswork in brute-force attacks. In *Indocrypt'00*, Calcutta, India.

[25] A. Rabkin. Personal knowledge questions for fallback authentication. In *SOUPS'08*, Pittsburgh, USA.

[26] D. Ramsbrock, R. Berthier, and M. Cukier. Profiling attacker behavior following SSH compromises. In *IEEE/IFIP Dependable Systems and Networks (DSN'07)*, Edinburgh, UK, June 2007.

[27] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *USENIX Security*, Baltimore, MD, USA, 2005.

[28] S. Schechter, A. J. B. Brush, and S. Egelman. It's no secret. Measuring the security and reliability of authentication via 'secret' questions. In *IEEE Symp. on Security and Privacy*, Oakland, CA, USA, May 2009.

[29] E. H. Spafford. The Internet worm: Crisis and aftermath. *Communications of the ACM*, 32(6), 1989.

[30] X. Suo and Y. Zhu. Graphical passwords: A survey. In *ACSAC'05*, Tucson, AZ, USA, Dec. 2005.

[31] P. van Oorschot and J. Thorpe. On predictive models and user-drawn graphical passwords. *ACM TISSEC*, 10(4), Jan. 2008.

[32] E. R. Verheul. Selecting secure passwords. In *CT-RSA*, San Francisco, CA, USA, Feb. 2007.

[33] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek. Password cracking using probabilistic context-free grammars. In *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2009.

[34] WindowsSecrets.com. Gmail flaw shows value of strong passwords. News article (Aug. 06, 2009). `http://windowssecrets.com/2009/08/06/01-Gmail-flaw-shows-value-of-strong-passwords/`.

[35] Wired.com. Palin e-mail hacker says it was easy. Sept. 18, 2008. `http://www.wired.com/threatlevel/2008/09/palin-e-mail-ha/`.

[36] T. Wu. A real-world analysis of Kerberos password security. In *NDSS'99*, San Diego, CA, USA, Feb. 1999.

[37] R. V. Yampolskiy. Analyzing user password selection behavior for reduction of password space. In *IEEE Carnahan Conferences on Security Technology (CCST'06)*, Lexington, KY, USA, Oct. 2006.

[38] J. Yan. A note on proactive password checking. In *NSPW'01*, Cloudcroft, NM, USA, Sept. 2001.

[39] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password memorability and security: Empirical results. *IEEE Security & Privacy*, 2(5), Sept.-Oct. 2004.

[40] K.-P. Yee and K. Sitaker. Passpet: Convenient password management and phishing protection. In *SOUPS'06*, Pittsburgh, USA.

[41] M. Zviran and W. J. Haga. Cognitive passwords: The key to easy access control. *Computers & Security*, 9(8), Dec. 1990.