

Strengthening Password-Based Authentication

Submitted by

Fatma Abdullah Amer AL Maqbali

for the degree of Doctor of Philosophy

of the

Royal Holloway, University of London



2019

Declaration

I, Fatma Abdullah Amer AL Maqbali, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed.....(Fatma Abdullah Amer AL Maqbali)

Date:

To my husband Nasser, my daughter Azzahraa, and my father.

Abstract

Authenticating humans to computers remains problematic despite decades of effort. Password-based authentication remains extremely common, in spite of its widely known shortcomings, and this seems likely to continue. Hence improving the security of password use is a topic of huge practical importance — this observation provides the motivation for the work described in this thesis. We focus in particular on two topics, namely password generators and password recovery.

Password generators provide site-specific passwords on demand, facilitating the use of site-specific and complex passwords; they are an alternative to password managers that avoid storing passwords long-term. We proposed a general model for such systems, and critically examine options for instantiating this model, including all those previously proposed. The model has also been used to help design a new scheme, AutoPass, intended to incorporate the best features of the prior art whilst also addressing many of the most serious shortcomings of existing systems through use of novel techniques. AutoPass is specified in detail, and a prototype implementation is described. The prototype has been user-trialled to test its usability and security.

Because passwords are very widely used for user authentication, most websites using passwords also implement password recovery to allow users to re-establish a shared secret if the existing value is forgotten; however, use of such a fall-back creates additional vulnerabilities. We present a model for such systems, and use this to analyse existing approaches. This leads naturally to a set of recommendations for system implementation. Many password recovery systems involve sending a special email to the user, e.g. containing a secret link, in which case security will depend on the email being acted upon correctly; unfortunately, such emails are not always well-designed and can introduce vulnerabilities. To understand better this serious practical issue, we surveyed password recovery emails for 50 of the top English language websites and investigated their design, structure and content. We found that many well-known web services, including Facebook, Dropbox, and Microsoft, suffer from recovery email design, structure and content issues. This study enabled us to formulate recommendations for the design of such emails.

Acknowledgements

The research in this thesis would not have been possible to complete without the encouragement of many people. It is a delight to acknowledge all those who have supported me over the last four years.

I am sincerely and heartily grateful to Professor Chris J. Mitchell, my PhD supervisor, for his endless support, guidance, motivation, patience and immense knowledge throughout my PhD study and related research. His guidance helped me throughout the time spent in performing the research for and writing up of this thesis. I could not have imagined having a better supervisor for my PhD study.

I cannot find words to express my heartfelt thanks to my soul mate, my workmate, my best friend, and my husband, Nasser, for his continued and unfailing love, support and understanding before and during our study of PhD degree. He is always been the one for me to lean on at all times.

I owe thanks to my lovely daughter, Azzahraa who is the pride and joy of my life. Having her during my PhD gave me strength and helped me to be more efficient.

I wholeheartedly thank my late father, Abdullah Amer AL Maqbali who could not see this thesis completed and who always believed in me, supported me in every decision I made and every step I took to ensure that I reach where I want to be. He was always keen to know what I was doing and how I was proceeding. I miss his smile of joy whenever a significant moment was reached.

A very special gratitude goes to my aunties Moza and Saleema who have always been on my side throughout my journey.

I would particularly like to thank Dr Wanpeng Li who dedicated his free time voluntarily to develop the AutoPass extension which is an important part of the thesis.

I am profoundly appreciative to my colleagues, Ms Angela Heeler, Mr Gaetan Pradel, Mr Zhaoyi Fan, Dr Wanpeng Li, Ms Zhang Xiao, Mr Mohammed Shafiq Alam Khan, and my husband Nasser for the friendly atmosphere they have created for our reading group, and for the invaluable feedback and insightful ideas I got from them.

I would like to show my gratitude to my siblings: Amur, Wafa, Maryam, Noof, Suleiman, Mohammed, Majed, Ashwaq, Ayman, Areem and Turkey.

I am very grateful to my mother Shaika AL Maqbali, my step-mother Shaika AL Maqbali, my mother-in-law Suad Al-Fannah, my father-in-law Mohammed Al-Fannah and all my relatives and friends.

Finally, I owe sincere and earnest thankfulness to Allah who gave me the blessing to have amazing people in my life and for giving me the strength to do my PhD.

Contents

1	Introduction	1
1.1	Passwords	1
1.2	Managing Passwords	2
1.3	Contributions	3
1.4	Joint Work	4
1.5	List of Publications	4
1.6	Structure of Thesis	5
I	Background	7
2	User Authentication	8
2.1	Introduction	8
2.1.1	Identification and Authentication	8
2.1.2	User Authentication to Computers	9
2.1.3	User Authentication Model	11
2.2	The User Authentication Lifecycle	13
2.2.1	The Main Steps	13
2.2.2	Registration	13
2.2.3	Performing Authentication	13
2.2.4	Account Maintenance	14
2.2.5	Account Closure	14
2.3	Passwords: Use and Vulnerabilities	15
2.3.1	Setting up a Password	15
2.3.2	Using Passwords	15
2.3.3	Password Vulnerabilities	17
2.3.4	Verifying Passwords	17
2.4	Password Managers	18
2.5	Password Generators	22

2.6	Password Recovery	22
2.7	Conclusions	23
3	Password Generators	24
3.1	Introduction	24
3.2	A Definition	24
3.3	Previous Work	25
3.4	A Taxonomy	33
3.5	Shortcomings of Existing Techniques	34
3.6	Conclusions	35
4	Password Recovery	36
4.1	Introduction	36
4.2	Previous Work	37
4.3	Issues with Password Recovery	41
4.4	Conclusions	42
II	Password Generation	43
5	Password Generators: Old Ideas and New	44
5.1	Introduction	44
5.2	The Model	45
5.2.1	Operation	45
5.2.2	Registration and Configuration	46
5.3	Components of the Model	46
5.3.1	Inputs to Password Generation	46
5.3.2	Generating the Password	47
5.3.3	Password Output and Use	49
5.3.4	Approaches to Implementation	49
5.4	Assessing the Options	50
5.4.1	Inputs	50
5.4.2	Generating the Password	51
5.4.3	User Interface Operation	52
5.4.4	Implementation	52
5.4.5	Other Issues	53
5.5	Improving System Operation	54
5.5.1	Novel Types of Configuration Data	54
5.5.2	Use of A Server	55

5.6	Conclusions	56
6	AutoPass: An Automatic Password Generator	57
6.1	Introduction	57
6.2	AutoPass: High-level Specification	58
6.3	AutoPass: Detailed Specification	59
6.3.1	Structure	59
6.3.2	Operation	59
6.3.3	Stored Data	63
6.3.4	Use of PRML	64
6.3.5	Password Offsets	65
6.4	Details of Operation	66
6.4.1	Application Installation	66
6.4.2	Operational Sessions	68
6.4.3	Use with a Website	70
6.5	Other Aspects	72
6.6	Security Evaluation	73
6.6.1	Trust Relationships	73
6.6.2	Threat Model	73
6.6.3	Security Properties	74
6.6.4	Security from An Attacker Perspective	75
6.7	Addressing Shortcomings	77
6.8	Conclusions	78
7	Testing AutoPass	79
7.1	Introduction	79
7.2	Prototype Design and Implementation	79
7.2.1	Overall Structure	80
7.2.2	AutoPass Server	80
7.2.3	AutoPass Client	80
7.2.4	Installing and Using the AutoPass Client	81
7.3	A User Trial	85
7.3.1	Methodology	85
7.3.2	User Base	87
7.3.3	Summary of Results	87
7.3.4	Analysis	89
7.4	Conclusions	92

III Password Recovery	93
8 Modelling the Password Recovery Process	94
8.1 Introduction	94
8.2 A General Model	95
8.2.1 Constituent Processes	95
8.2.2 Registration	95
8.2.3 Password Setup	96
8.2.4 Recovery	96
8.3 Model Components	97
8.3.1 Registration	97
8.3.2 Password Setup	99
8.3.3 Recovery Request and Validation	100
8.3.4 Password Re-establishment	101
8.3.5 After Password Re-establishment	102
8.4 Security and Usability Issues	103
8.4.1 Security Questions	103
8.4.2 Trustee-based Recovery	104
8.4.3 Email Reset	104
8.4.4 Verification Codes and Reset Links	105
8.4.5 SMS Reset	107
8.4.6 Other Issues	108
8.5 Towards Secure Password Recovery	109
8.5.1 Recovery Request Validation	109
8.5.2 Password Re-establishment	110
8.6 Conclusions	111
9 Real-world Email-Based Password Recovery	112
9.1 Email-Based Password Recovery	112
9.2 Motivation and Methodology	113
9.3 Password Recovery Emails	113
9.3.1 Use of Recovery Emails	113
9.3.2 Email Structure	115
9.3.3 Email Client Features	116
9.4 Risks from Recovery Emails	116
9.5 A Study of Existing Practice	117
9.5.1 Scope of Study	117
9.5.2 Methodology	119

9.6	Results	119
9.6.1	Example Recovery Emails	119
9.6.2	Email Header	120
9.6.3	Message Body	121
9.6.4	Message Signature	122
9.6.5	Summary of Risks	122
9.7	Recommendations	126
9.8	Disclosure	127
9.9	Conclusions	127
IV	Conclusions	128
10	Concluding Matters	129
10.1	Summary of Contributions	129
10.2	Directions for Future Work	130
10.3	The Future of Passwords	131
Bibliography		131
A	AutoPass Prototype and User Trial	I
A.1	AutoPass client – code extracts	I
A.2	AutoPass Testing Documents	XVIII
A.3	Consent Form	XVIII
A.4	Testing AutoPass: A user guide	XX
A.5	AutoPass: An Introductory Guide	XXVI
A.6	AutoPass evaluation questionnaire	XXXIV
A.7	AutoPass Survey Outcome	XXXVII
B	Password Recovery Emails – Disclosure and Responses	XLIII
B.1	Email Sent to the surveyed websites	XLIII
B.2	vimeo.com	XLV
B.3	wikipedia.org	XLV
B.4	Wix.com	XLVI
B.5	cnet.com	XLVII
B.6	dailymail.co.uk	XLVIII
B.7	IMDb.com	XLVIII
C	Other Issues Identified in Email-based Password Recovery	XLIX

List of Figures

2.1	The elements of an authentication system [82]	10
2.2	User authentication categories [70]	11
2.3	Digital Identity Model [37]	12
2.4	Activity diagram of password-based authentication [44]	14
3.1	Site-Specific Passwords (SSP),[54]	26
3.2	PwdHash, [77]	27
3.3	Password Multiplier, [39]	28
3.4	Password generation by PasswordSitter, [93]	29
3.5	PassPet operation, [94]	30
3.6	ObPwd operation, [66]	31
3.7	PALPAS password generation, [45]	33
3.8	Password management systems: a taxonomy (McCarney [68])	34
4.1	Trusted party based recovery, [84]	39
4.2	MITM attack on password recovery (Gelernter et al. [33])	40
6.1	AutoPass System Architecture	60
6.2	Password generation in AutoPass	62
6.3	AutoPass operational sessions	70
7.1	AutoPass extension from Google Play	81
7.2	AutoPass login window	83
7.3	AutoPass pop-up for user consent	83
7.4	AutoPass password generation window	84
7.5	Use your own password window	85
7.6	Participant age distribution	87
7.7	password generation strategies	88
7.8	Level of satisfaction with passwords generated using AutoPass	90
7.9	Did you find the ability to use existing passwords useful?	90

7.10	Have you ever used a computer program to generate your passwords?	91
7.11	If you ever used a computer program to generate your passwords, how does AutoPass compare with methods you have used previously?	91
8.1	Google password change hint	100
8.2	Wikipedia password recovery using a temporary password	102
8.3	Partial leakage of user contact number	106
8.4	Partial leakage of alternative email address	106
9.1	Example password recovery email (from dropbox.com)	114
9.2	Message preheader example	115
9.3	Example recovery email: Lacking information	119
9.4	Example recovery email: Readability issue	120
9.5	Facebook password recovery email	121
1	Response from vimeo.com	XLV
2	Response from wikipedia.org	XLV
3	Response from Wix.com	XLVI
4	Response from cnet.com	XLVII
5	Response from dailymail.co.uk	XLVIII
6	Response from IMDb.com	XLVIII

List of Tables

2.1	Password managers compared	21
4.1	Password recovery privacy violations (Guri et al. [38])	40
5.1	Inputs to password generation process	48
7.1	Issues user face with using password	88
7.2	User password generation techniques	89
9.1	The 50 websites tested in the study	118
9.2	The 50 websites with issues identified (Part 1)	124
9.3	The 50 websites with issues identified (Part 2)	125
1	Other issues identified in email-based password recovery (Part 1)	L
2	Other issues identified in email-based password recovery (Part 2)	LI

List of Abbreviations

2FA	Two-factor Authentication
ASCII	American Standard Code for Information Interchange
AWS	Amazon Web Services
CAPTCHA	Completely automated public Turing test to tell computers and humans apart
CSPs	Credential Service Providers
DKIM	Domain Keys Identified Mail
DMARK	Domain-Based Message Authentication, Reporting and Conformance
FIDO	Fast IDentity Online
GDPR	General Data Protection Regulation
IMAP	Internet Mail Access Protocol
IMSI	Identifier Mobile Subscriber Identity
ISE	Independent Security Evaluators
MFA	Multi-Factor Authentication
MIME	Multipurpose Internet Mail Extensions
MitM	Man-in-the-Middle Attack
NIST	National Institute of Standards and Technology
PALPAS	PasswordLess Password Synchronization
PRD	Password Requirements Descriptions
PRML	Password Requirements Markup Language
RP	Relying Parties
SMS	Short Message Service
SMTP	Simple Mail Transfer (or Transport) Protocol
SPF	Sender Policy Framework
SSP	Site-Specific Passwords
SS7	Signaling System No. 7
SSL	Secure Sockets Layer
TLS	Transport Layer Security
URL	Universal Resource Locator
KBA	Knowledge-Based Authentication
QR	Quick Response Code
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 Passwords

Today users communicate and interact with many types of Internet service, e.g. commercial websites, news websites, social networking websites and blogs. Many such services require users to create an account, which may have a huge amount of personal data associated with it, such as: pictures, documents, files, and histories of shopping/transactions. To protect these sensitive data from unauthorised access and restrict access to paid-for services, online accounts are typically protected by passwords [22]. Password-based authentication has many advantages for both users and service providers. Users can use passwords across multiple platforms, devices, and applications, and service providers can implement password-based authentication with little effort and minimal cost per user, as discussed by many authors — see, for example, [42, 74].

Despite their widely-discussed shortcomings, passwords are widely used to authenticate users to online services [23]. Many attempts have been made to replace simple password authentication, e.g. using biometrics, tokens and multi-factor authentication [42]. However, single-factor password-based authentication remains very widely used. Moreover, in recent years the number of widely-used password-protected services has grown significantly, in turn increasing the number of passwords users are expected to remember. There are a huge range of issues associated with the use of text passwords, including the need to manage passwords for large numbers of accounts, as discussed by Florêncio et al. [27], Bonneau et al. [11], Adams et al. [7], and Robert et al. [69]. In general, these issues can be categorised as either user-related or online service-related.

User-related issues include:

- users are likely to be overwhelmed by the large number of passwords needed

for Internet services, which can lead to use of the same password for multiple accounts (referred to as password fatigue by Sanchez et al. [79]);

- users will often choose guessable passwords, e.g. date of birth, name of pet, or anniversary date;
- users will often make minimal modifications to an existing password, e.g. by including a serial number, when forced to make a change.

Online service-related issues, which can make it almost impossible for users to remember all their passwords, include:

- many sites enforce a complex password policy, e.g. requiring passwords to contain a minimum number of characters, or include or exclude specific characters;
- some services force users to change their password regularly, e.g. every 90 days.

These issues are all well-known and have been widely discussed (see, for example, [28, 43]), as have the many possible alternative means of user authentication. However, for convenience and cost reasons passwords remain very widely used, and in practice the issues identified above ensure that in practice password authentication is often very insecure. For example, users often select easily guessed passwords and/or re-use the same password with many sites. The goal of this thesis is to help make it possible in practice for users to use passwords in a more secure way, and in particular to use strong (i.e. not easily guessed) passwords that vary from site to site.

1.2 Managing Passwords

As noted above, despite its well-known shortcomings, password-based user authentication remains in very widespread use. This is despite the many possible alternative authentication technologies, including biometric-based authentication, the use of secure tokens, and identity management systems such as FIDO¹ and FIDO2². Rather than look at yet more alternatives, or try to find better ways to deploy these technologies, this thesis is based on the premise that passwords will continue to be widely used, and that it is therefore vitally important to find ways of making their use more secure. That is, this thesis is concerned with the general topic of *password management*.

One approach to improving the security of password-based authentication is to provide means for users to manage their passwords. So called *password managers* (see,

¹<https://fidoalliance.org/>

²<https://fidoalliance.org/fido2/>

for example, [68, 81]) seek to achieve this by storing passwords for users and supplying them on demand. Commercial password management services are widely available — such as LastPass³, Dashlane⁴, and RoboForm⁵ — and password management functionality is also included as part of most commonly used browsers. However, the fact that password managers need to retain copies of user passwords give rise to both security and usability issues. If the passwords are stored locally, then they will no longer be available when the user switches to a different platform. Alternatively, if they are stored in the cloud (as is the case, for example, for Safe In Cloud⁶) then there is the danger of compromise as a result of a web-based attack, examples of which have been widely discussed by Silver et al. [81] and the Independent Security Evaluators (ISE)⁷ group. These observations have led the development of what we call password generators, i.e. systems which can generate a site-specific password on demand, but which avoid the need to store a database of user passwords. Such systems can help address the usability and security issues that arise for password managers. Password generators are the main focus of Part II of this thesis.

Major problems can arise when a user forgets their password. Because this is such a common event, almost all websites supporting user authentication also offer a means for password recovery, to enable the user to re-establish a password for authentication. However, there are clear security threats arising from such a process, notably that an unauthorised party might trigger or somehow subvert the process to gain access to a user account. It is therefore vital that such systems are well-designed. This is the issue we address in Part III of this thesis.

1.3 Contributions

In this thesis we address the general problem of improving the security and usability of password-based user authentication to web services. This is an area of huge practical importance, given the current ubiquity of password-based authentication. As noted above, the research described in this thesis makes contributions in two specific areas within this field: password generators and password recovery systems.

Password generators provide site-specific passwords on demand, facilitating the use of site-specific and complex passwords; they are an alternative to password managers that avoid storing passwords long-term. We propose a general model for such sys-

³<https://www.lastpass.com>

⁴<https://www.dashlane.com>

⁵<https://www.roboform.com>

⁶<https://safe-in-cloud.com/en/>

⁷<https://www.securityevaluators.com/casestudies/password-manager-hacking/>

tems, and critically examine options for instantiating this model, including all those previously proposed. This analysis reveals a number of practical issues with existing password generator schemes, for which we propose possible solutions. The model has also been used to help design a new scheme, AutoPass, intended to incorporate the best features of the prior art whilst also addressing many of the most serious shortcomings of existing systems through use of the novel features we identified as a result of our analysis. AutoPass is specified in detail, and a prototype implementation is described. The prototype has been user-trialled to test its usability and security.

Because passwords are very widely used for user authentication, most websites using passwords also implement password recovery to allow users to re-establish a shared secret if the existing value is forgotten; however, use of such a fall-back creates additional vulnerabilities. We present a model for such systems, and use this to analyse existing approaches. This leads naturally to a set of recommendations for system implementation. Many password recovery systems involve sending a special email to the user, e.g. containing a secret link, in which case security will depend on the email being acted upon correctly; unfortunately, such emails are not always well-designed, and can introduce vulnerabilities. To understand better this serious practical issue, we surveyed password recovery emails for 50 of the top English language websites and investigated their design, structure and content. We found that many well-known web services, including Facebook, Dropbox, and Microsoft, suffer from recovery email design, structure and content issues. This study enabled us to formulate recommendations for the design of such emails.

1.4 Joint Work

I have performed all the research described in this thesis, under the supervision of Professor Chris Mitchell, with the following single exception. Coding for the prototype implementation of AutoPass described in Chapter 7 was performed by Dr Wanpeng Li, although decisions about functionality remained under my control.

1.5 List of Publications

The following published papers describe much of the work addressed in this thesis.

- F. Al Maqbali and C. J. Mitchell. Password generators: Old ideas and new. In S. Foresti and J. Lopez, editors, *Information Security Theory and Practice — 10th IFIP WG 11.2 International Conference, WISTP 2016, Heraklion, Crete*,

Greece, September 26-27, 2016, *Proceedings*, volume 9895 of *Lecture Notes in Computer Science*, pages 245–253. Springer, 2016.

- F. Al Maqbali and C. J. Mitchell. Autopass: An automatic password generator. In *International Carnahan Conference on Security Technology, ICCST 2017, Madrid, Spain, October 23-26, 2017*, pages 1–6. IEEE, 2017.
- F. Al Maqbali and C. J. Mitchell. Web password recovery: A necessary evil? In K. Arai, R. Bhatia, and S. Kapoor, editors, *Proceedings of the Future Technologies Conference, FTC, Vancouver, Canada, November 15-16, 2018*, pages 324–341. Springer, 2018.
- F. Al Maqbali and C. J. Mitchell. Email-based password recovery — risking or rescuing users? In *2018 International Carnahan Conference on Security Technology, ICCST 2018, Montreal, QC, Canada, October 22-25, 2018*, pages 1–5. IEEE, 2018.

A further paper (based on Chapter 7 of this thesis) is about to be submitted.

- F. Al Maqbali, W. Li and C. J. Mitchell, AutoPass: Implementation and testing.

1.6 Structure of Thesis

The remainder of this thesis is divided into four parts, as follows.

1. Part I provides background material. It contains the following three chapters.
 - Chapter 2 introduces user authentication, and covers the following topics: an overview of user authentication, the user authentication lifecycle, password use and vulnerabilities, password managers, password generators and password recovery.
 - Chapter 3 provides a description of password generators, a review of the prior art and a discussion of the shortcomings of existing password generators.
 - Chapter 4 introduces password recovery and identifies a range of issues associated with it.
2. Part II is concerned with password generators.
 - Chapter 5 introduces a general model for the operation of password generators. It considers existing password generator mechanisms within the context of this model, and also considers certain novel mechanisms designed to address identified shortcomings in existing schemes.

- Chapter 6 provides a detailed specification and analysis of AutoPass, a password generator scheme designed using concepts introduced in Chapter 5.
 - Chapter 7 describes in detail the design of the AutoPass password generator prototype, and also gives the results of a preliminary user trial of this prototype.
3. Part III addresses the password recovery and in particular email-based password recovery. It contains two chapters, as follows.
- Chapter 8 introduces a general model for the password recovery process, and uses this to give a structured analysis of usability and security issues for such systems. It also provides a set of recommendations designed to help develop secure and usable password recovery mechanisms.
 - Chapter 9 describes and analyses the results of a real-world study of the operation of email-based password recovery for 50 major English language websites.
4. Part IV concludes the thesis by summarising the main contributions as well as highlighting possible areas for future work. This part of the thesis consists of a single chapter, Chapter 10.

Part I

Background

Chapter 2

User Authentication

2.1 Introduction

This first main part of the thesis, consisting of three chapters, provides the background material and review of previous work necessary for the remainder of the thesis. This chapter is concerned with introducing user authentication, covering what it is and how it works. It also introduces password-based user authentication, and reviews some of the major threats to its security. Finally, the topics of password managers, password generators and password recovery are briefly introduced. Previous work on password generators is discussed in greater detail in Chapter 3. This is followed, in Chapter 4, by a review of the state of the art in password recovery.

2.1.1 Identification and Authentication

For the purposes of this thesis we define *user identification* as a claim by a *user* (or *subject*¹) of possession of a *user identifier* to a requesting party (the *verifier* or *authenticating party*). That is, user identification enables the verifier to learn who the user might be, but does not provide assurance that this knowledge is correct. Typically, user identification will involve learning who the user is from amongst a pre-defined group of users, e.g. those who have registered for the service concerned. *User authentication* is then the provision of assurance to the verifier that the user is the legitimate owner of that identifier. That is, user authentication provides assurance in the correctness of the identifier that is learnt from user identification.

The nature of the identifier will, of course, depend on the context, but could include an email address or a value assigned to an employee by an organisation. User identification and user authentication are fundamentally important technologies for en-

¹ The terms user and subject are used interchangeably throughout the thesis.

forcing information security policies; in particular, without reliable identification and authentication of users, it is impossible to enforce access control or accountability.

User identification is an age-old concept, with humans apparently pre-programmed to recognise people from their faces and voices. The need for user authentication has grown as societies have become larger and more complex, giving rise to the need for one human to authenticate another when they were not previously known to each other. With communications technology enabling remote interactions, e.g. by phone or computer, and where the identifying party is a machine not a human, user authentication now needs to be performed very frequently.

Of course, there are more general notions of user authentication, not requiring identification as a precursor. For example, ISO/IEC 27000 [50] defines *authentication* as “provision of assurance that a claimed characteristic of an entity is correct”, where this characteristic (or attribute) could be an identifier but could also be any other information associated with the user, e.g. that the user is over 21, is an employee of a specific company, or is female. Use of a non-identifying attribute allows access to a service to be authorised in a privacy-friendly way, although this more general type of user authentication is outside the scope of this thesis. Note also that the terms authentication and entity authentication are commonly used to cover technologies designed to enable one machine to prove its identity to another machine (see, for example, Boyd and Mathuria [13]). Typically, such mechanisms involve exchanges of cryptographically-protected values. However, such techniques are outside the scope of this thesis, since we are only concerned with the case where the party being authenticated is a human being. For convenience, in the remainder of this thesis we sometimes drop the word user and just refer to authentication when we implicitly mean user authentication.

There are many ways of achieving authentication, i.e. ways of providing assurance in the correctness of the identifier to the authenticating party. In this thesis we focus on the case where the verifier is a computer, the subject is a human, and the means of providing assurance is a password. We look at these issues in greater detail immediately below.

2.1.2 User Authentication to Computers

As noted above, in this thesis we are concerned with the case where the verifier is a computer. In this context, the main goal of user authentication is typically to support *authorisation*, i.e. verification of the subject’s right to access a particular service. In such a case the computer typically has an authentication database containing information associated with each authorised subject, that is used as part of the user authentication process. The subject is typically equipped with credentials of some kind, and

these credentials are then employed by the subject to provide the necessary assurance to the verifier. This is illustrated in Figure 2.1, which shows the main elements of an authentication system.

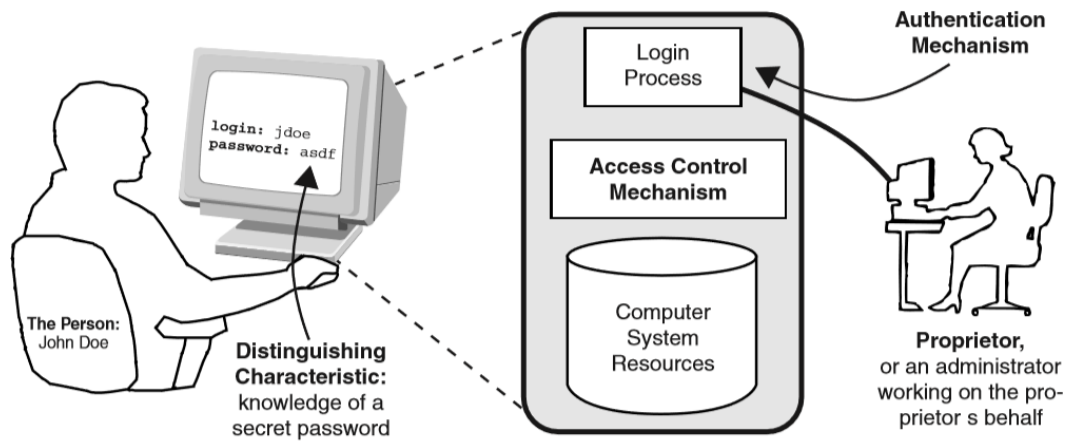


Figure 2.1: The elements of an authentication system [82]

As discussed by many authors (see, for example, O’Gorman [70], Smith [82] and Stallings et al. [83]), the techniques used for user authentication in this man-machine context can be divided into three main categories, depending on the nature of the user credentials:

- *what you know*: knowledge-based authentication, where the user credential is information held by the user;
- *what you have*: token-based authentication [20], where the user credential is a physical token; and
- *what you are*: ID-based or biometric authentication, where the user credential is some physical or behavioural characteristic of the user [40].

This classification is shown diagrammatically in Figure 2.2. Of course, for greater assurance it is possible to use more than one type of credential, giving rise to the notion of *dual-* or *multi-factor authentication* (as, for example, discussed by Smith [82]).

In this thesis we are concerned with knowledge-based authentication, and in particular where the user has a password which he/she is typically requested to remember and supply on demand for authentication purposes. The verifier database will either

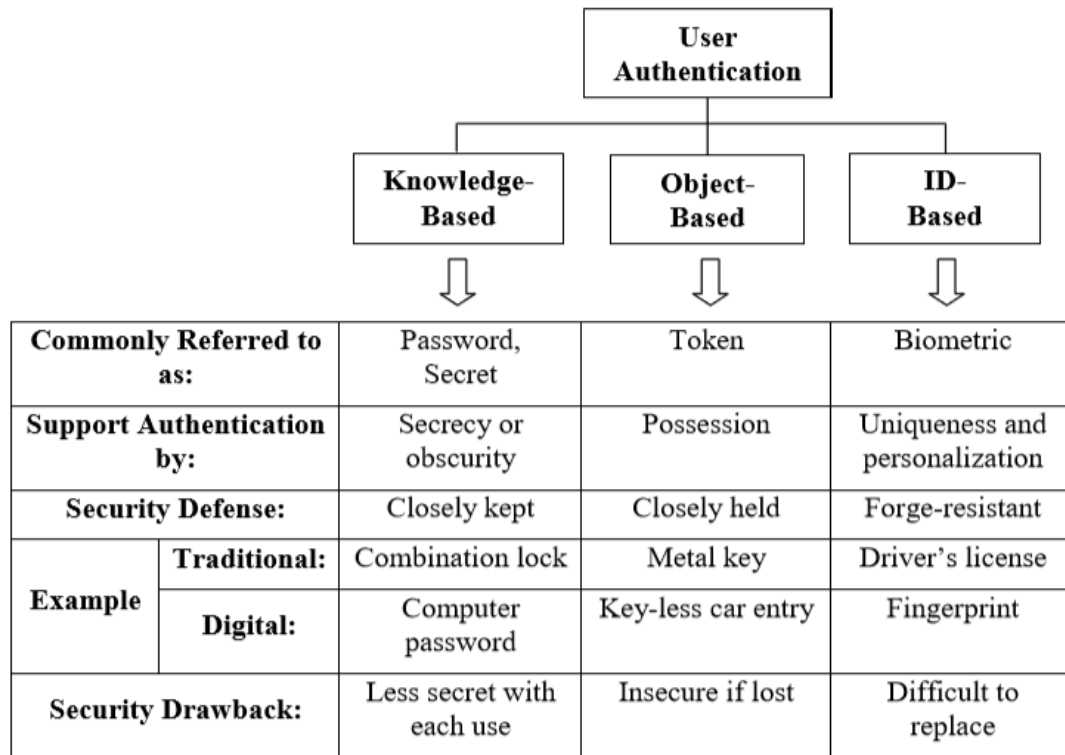


Figure 2.2: User authentication categories [70]

contain a copy of the password or, more commonly, information which can be used to verify whether or not an offered password is correct (e.g. the hash of a password).

2.1.3 User Authentication Model

The user authentication step, involving use of credentials to verify a claimed identity, is typically only part of a larger authentication system. To illustrate this larger context we briefly introduce the model of the user authentication process provided by NIST SP 800-63-3 [37]. The main entities in this model that directly relate to authentication are as follows.

- The *subject* (or user in our terminology) is the human user who wishes to access a service provided by the verifier.
- The *claimant* is the party to be authenticated (for the purposes of this thesis, the subject and claimant are the same entity).

- The *verifier* is the party that verifies the claimant's identity.

The NIST model also introduces two further entities, the *subscriber* and the *applicant*, that relate to a process known as *identity proofing*. Identity proofing, where the subject's claim to an identity is verified, is a step that occurs prior to routine user authentication, when a new subject is to be added to a system (part of registration, discussed in greater detail in 2.2.2 below). We observe that in a typical web application in which a password is used for user authentication (i.e. the scenario that forms the main focus of this thesis), the identity proofing step may be omitted since there are no pre-existing resources (or relationship) to which access is being controlled.

Figure 2.3 (taken from SP 800-63-3 [36]) shows the various entities and interactions in the NIST model. The figure shows two other entities that may be involved in some authentication scenarios, although they are beyond the scope of the systems we consider in this thesis:

- *Credential Service Providers* (CSPs) are involved in generating credentials and other functions relating to registration;
- *Relying Parties* (RPs) depend on the results of authentication, e.g. in the provision of services.

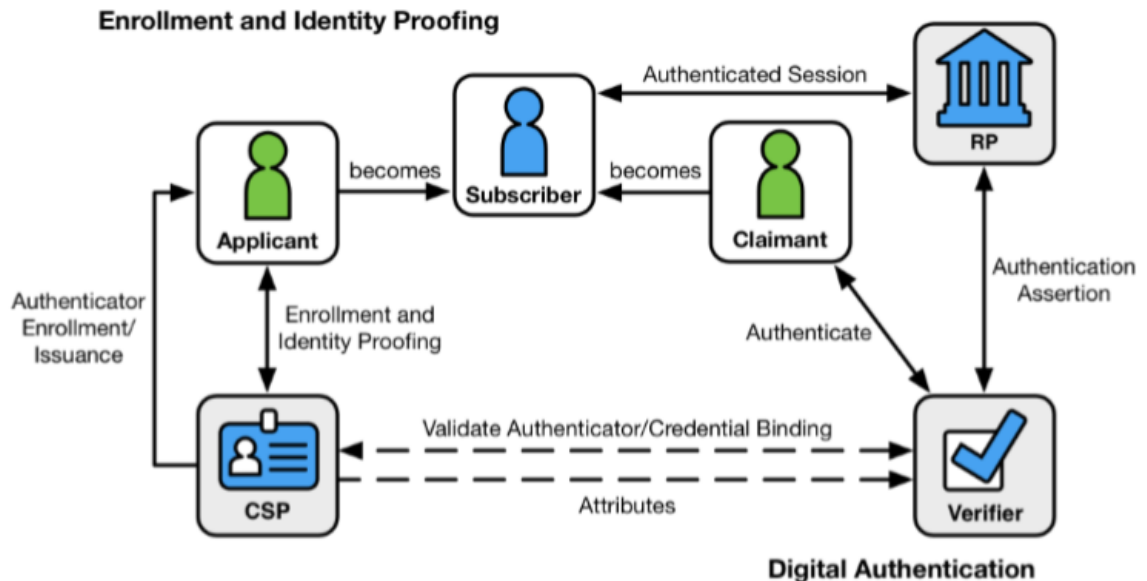


Figure 2.3: Digital Identity Model [37]

2.2 The User Authentication Lifecycle

2.2.1 The Main Steps

The various parties in an authentication system, as introduced above, are involved in a range of activities during the lifetime of a subject's relationship with a verifier. We summarise these as part of the activities that we refer to as the *user authentication lifecycle*. This involves the following four main stages, described in greater detail below.

- *registration*, i.e. the establishment of a relationship between the subject and the verifier, which includes identity proofing, credential establishment, and adding the subject to the verifier's authentication database;
- *performing authentication*, i.e. the everyday use of credentials to enable the identity of a user to be authenticated by a verifier;
- *account maintenance*, including credential change and account recovery, where the verifier re-establishes its relationship with a subject if the subject forgets or loses his or her identifier and/or credentials; and
- *account closure*, i.e. where the subject terminates his or her relationship with the verifier.

2.2.2 Registration

Registration (also sometimes referred to as enrolment) involves the user and verifier setting up their relationship, including establishing the user credentials and adding appropriate information to the verifier's authentication database. In general, registration includes identity proofing but, in the context of interest here, i.e. interactions between a user and a website, this step is typically omitted. As part of registration, a website will also typically require the user to choose a unique user name (or give an email address to function as a user name). Examples of the types of personal information that may be gathered during registration include first name, last name, gender, and date of birth.

2.2.3 Performing Authentication

Once registration is complete, a user can employ the established credentials to authenticate him/herself to the verifier. In the case of interest in this thesis, this means providing the password when requested. Figure 2.4 (due to Horsch [44]) shows the authentication process.

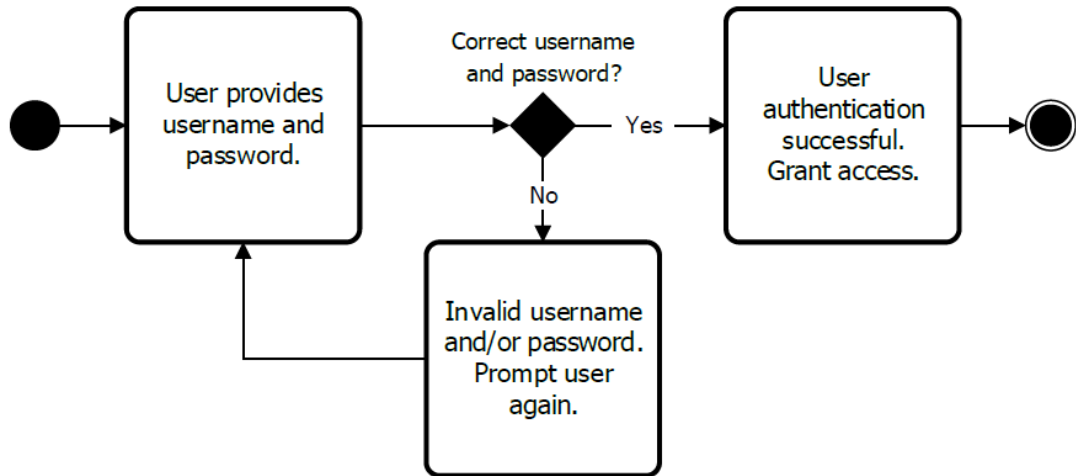


Figure 2.4: Activity diagram of password-based authentication [44]

2.2.4 Account Maintenance

During the lifetime of the relationship between a user and a verifier it may be necessary to update the credentials or recover a lost identifier or lost credentials. Credential update is clearly an important step, although we do not consider this in detail here. However, addressing the situation where the user and verifier relationship needs repair is a topic addressed in detail later in this thesis. We use the term *account recovery* to cover any steps taken to re-enable identification and authentication.

One case of account recovery of particular interest here is what we term *credential recovery*, relevant if the user no longer has access to the credentials necessary to complete authentication. How this situation might arise, along with the possible methods for recovery, very much depend on the nature of the credentials. This thesis is primarily concerned with the case where the credentials are passwords, in which case we refer to *password recovery*. We discuss password recovery in greater detail in Section 2.6.

2.2.5 Account Closure

Account closure is performed when a user wishes to terminate his or her relationship with the verifier. As a result of this step the user credentials will be invalidated, and the user will no longer be able to access the services provided by the verifier that require users to log on. What happens to any personal information of the user after account closure is an important issue. European data privacy law, notably the General Data

Privacy Regulation (GDPR) [18], requires personal data to be deleted when it is no longer required for the purpose for which it was gathered. This notion is consistent with principles in international data privacy standards, notably ISO/IEC 29100 [49], which, as part of the data minimization principle, requires that ICT systems ‘delete and dispose of PII whenever the purpose for PII processing has expired, there are no legal requirements to keep the PII or whenever it is practical to do so’. Erasing redundant personal information is also consistent with the GDPR notion of the Right to erasure (also known as the right to be forgotten), which is covered in detail in Article 17 of the GDPR regulation [18]. In particular this article specifies the conditions under which a user has the right to insist on the erasure of personal data. However, whilst clearly of key importance, such privacy issues are not addressed in detail in this thesis.

2.3 Passwords: Use and Vulnerabilities

2.3.1 Setting up a Password

If passwords are being used for authentication, a user will be required to choose a password as part of registration. Some websites may enforce a *password policy*, i.e. a set of rules which selected passwords must satisfy, typically intended to try to make the password more difficult for someone to guess. Once a password has been selected, and after passing any policy checks, the verifier will store information necessary to verify it, e.g. the result of applying a cryptographic hash function to the password, in its authentication database.

2.3.2 Using Passwords

As has been discussed by many authors (e.g. [9, 69, 78, 92]), text-based passwords have been employed for user authentication for over 50 years, and password remains the most commonly used authentication method for online services [9]. This is likely because of its ease of implementation by both services and users, as described in [1, 11, 88], avoiding the need for costly and inconvenient token or biometrics-based solutions.

We focus primarily here on the case where the password is used as the sole means of authentication (the single-factor case). *Personal Identification Numbers* (PINs), i.e. short, fixed-length, numeric-only passwords, are typically used in conjunction with a token of some kind, i.e. in a dual-factor scheme; by contrast we are primarily concerned with the case where the password is of variable length and can include a wide range of characters.

In practice, the following security issues arise. Many of these problems derive from

the fact that users are expected to remember a large number of complex passwords, which in practice is impossible for most people. Recent National Cyber Security Centre (NCSC) statistics² reveal that UK citizens have on average 22 online passwords.

- Users are typically encouraged to use a different password with each service with which they interact. The reasons for this are clear — if a password is used with multiple sites and one of these is compromised, then there is the risk that the user can be impersonated to all the sites with which the user employs the same password. Since compromises of password databases are commonplace (see, for example, the Have I Been Pwned site³), password re-use represents a major risk to the security of authentication. Unfortunately, it is also well-established (see, for example, [21, 63]) that users widely re-use passwords. The same NCSC survey referred to above suggests that UK users employ the same password for an average of four sites⁴.
- Experimental evidence [44] suggests that, in order to resist brute force attacks on compromised hashed password databases, passwords should contain at least 12 characters; however, in practice users often choose much shorter passwords. This exposes them to brute force attacks if a password database is exposed (see Section 2.3.4).
- In an attempt to make password security more robust, websites impose a variety of password policies, i.e. sets of rules which user-chosen passwords must satisfy. Typical constraints cover the sets of characters which the password must contain, and minimum (and maximum) password lengths. However, there is no agreement on what the best policy is, meaning that there is a very wide variety of such policies — as Florencio and Herley [29] have observed, this actually has the perverse effect of making password management more difficult and potentially less secure for end users. Also, some websites enforce regular password changes, a practice which has recently been discouraged by a number of influential security advisory bodies⁵.

In practice most users find it very difficult to remember a large number of different *complex* passwords, i.e. passwords not easily guessed. This causes many users to re-use passwords with multiple sites and/or to store copies of passwords. However, there

²<https://cyforsecure.co.uk/blog/password-security-infographic-ncsc/>, Accessed: 29/04/12/2019

³<https://haveibeenpwned.com/>

⁴<https://www.ncsc.gov.uk/guidance/password-guidance-simplifying-your-approach>

⁵<https://www.ncsc.gov.uk/articles/problems-forcing-regular-password-expiry>

are technical solutions which help users manage multiple passwords, namely password managers (see Section 2.4) and password generators (see Section 2.5).

2.3.3 Password Vulnerabilities

There are many ways in which unauthorised parties could seek to learn passwords for other users' accounts. Key examples include the following.

- *Online guessing attacks.* The attacker could try random combinations of user names/passwords (perhaps using likely values for the user name and/or password) at a site in the hope of guessing a correct pair [89, 35]. Alternatively, if some user names are known (e.g. as might be the case if the site uses email addresses as user names), then multiple guessed passwords could be entered for each valid user name, in the hope of guessing the correct one. This might be made even simpler for the attacker if a valid password for this user name is known for a different site, and this could be used as a guess in the knowledge that password re-use is common. This approach is made less effective for the attacker if a website restricts the number of attempted logins to an account, e.g. by imposing increasing delays after the first three or four failed attempts.
- *Offline guessing attacks.* It is common for entire authentication databases to be compromised (see, for example, the Have I Been Pwned website⁶). As discussed in Section 2.3.4, these databases typically contain the results of applying a hash function to a password, often in combination with a random 'salt' value. This allows the attacker to test large numbers of guesses for the password for each account, e.g. using tables of 'likely' passwords. Indeed, hardware specifically designed for such large scale searches is available (see, for example, Horsch [44]).
- *Eavesdropping attacks.* If a password is sent across an insecure link (e.g. using HTTP rather than HTTPS across a wifi network operated by an attacker) then it might be intercepted.

2.3.4 Verifying Passwords

As we have discussed, we suppose that the verifier maintains an authentication database, containing the information necessary to verify a supplied password. This could actually involve storing the passwords themselves; however, this is generally considered bad practice since, if the database is compromised e.g. as a result of an attack on the server, then so are all the passwords. Instead, it is common practice to store the result

⁶<https://haveibeenpwned.com/>

of applying a cryptographic (one way) hash function to each password; this notion is very well-established, having been in use for the Unix account database since the 1970s (see, for example, the 1979 paper by Morris and Thompson [69]).

When an offered password needs to be verified, it is submitted to the same hash function and the result is compared to the value in the database. Of course, this still allows ‘brute force’ offline searches for a password if the authentication database is compromised. Moreover, if the same hash function is used for every password, this allows for simultaneous searching for all passwords in the database by hashing a guessed password and comparing it with all the hashes in the database. Such an attack can be made even more effective by precomputing hashes of widely used passwords; such a table can be used to quickly search all compromised authentication databases that use the same hash function.

A simple countermeasure, also discussed by Morris and Thompson [69], is to generate a random *salt* (a short bit string) for each account, and to combine this with the password (e.g. by concatenating them) before it is hashed. The salt is stored in the database along with the hashed password. This effectively means every password is hashed in a different way, meaning each hashed password has to be ‘brute forced’ individually. Also, as long as the salt is long enough (i.e. so that there are sufficiently many different salt values), this makes the calculation of precomputed tables of hashed passwords infeasible, since one would need to be created for each possible salt.

2.4 Password Managers

Many technologies have been devised to improve on the level of security provided by passwords. This includes: providing alternative means of user authentication, using an identity management system (such as OAuth [41] or FIDO [26]), or simply by making the use of strong, i.e. not readily guessable, passwords easier. In line with the previous remarks about the continuing ubiquity of passwords, we focus on the third of the above categories.

In particular, the fact that users are expected to memorise large numbers of different strong passwords simply to go about their daily business on the Internet, clearly forces users to compromise their own security; around 10 years ago, Florêncio and Herley, [28], reported that in a large scale study each user had about 25 accounts that required passwords, and typed an average of 8 passwords per day — these numbers have probably risen significantly since their study. The hope is that, in the short-to-medium term at least, systems can be devised to make the user workload manageable whilst still enabling the use of strong and diverse passwords.

One important category of schemes making strong passwords easier to use is made up of the *password managers* (what McCarney [68] calls *retrieval password managers*). A password manager is an online or offline application, possibly implemented as a browser extension, that captures and stores user passwords for online accounts. Along with the password, password managers typically also store the website URL and the user name. Depending on where it is stored, the password database can be protected in a variety of ways; if stored locally, it is typically encrypted using a key derived from a user-supplied master password.

Password managers come in many types, differing in the way they encrypt passwords, where they store passwords (locally or in the cloud), and in the additional features offered. One possible additional feature is automatic form-filling to minimise manual errors and protect against keyloggers. Table 2.1 summarises the features offered by nine widely discussed password managers. Table 2.1 indicates the presence or absence of 14 different possible features for a password manager for each of the nine chosen schemes. These 14 features are summarised below.

- *Offline Mode* indicates that the password manager can operate in the absence of an Internet connection.
- *Two-Factor Authentication* indicates that the scheme supports an additional authentication mechanism.
- *Browser Integration* indicates that the scheme supports integration with one or more web browsers.
- *AutoFill Forms* indicates that the scheme also enables auto-completion of forms, e.g. storing and filling in personal details.
- *Password Generation* indicates the scheme provides a facility to generate random passwords.
- *Security Alert* indicates that the scheme will provide a user with an alert whenever it has been used to provide a password for a website.
- *Portable Application* indicates that the scheme can be used without installing any special-purpose software.
- *Mobile Application* indicates that it has been implemented as a smart phone app.
- *Security Audits* indicates the ability of the scheme to verify the security of saved password, e.g. to highlight weak or duplicate passwords.

- *Import Password* indicates the ability of the scheme to import saved passwords from another password management scheme.
- *Export Password* indicates the ability of the scheme to export saved passwords to another password management scheme.
- *Single Sign-On* indicates the ability of the password manager to act as a central point of authentication for a range of web services.
- *Password Sharing* indicates that the scheme supports password sharing with nominated third parties.
- *Cloud-Based* indicates that the scheme stores user passwords in the cloud.

Table 2.1: Password managers compared

Password Manger	Features							
	Offline Mode	Two-Factor Authentication	Browser Integration	Autofill Forms	Password Generation	Security Alert	Portable Application	Mobile Application
LastPass	•	•	•	•	•	•	•	•
KeePass	•	•	•		•		•	•
1Password	•	•	•	•	•	•		•
Dashlane	•	•	•	•	•	•	•	•
Passwordstate	•	•	•	•	•		•	•
Keeper	•	•	•	•	•		•	•
Sticky Password	•	•	•	•	•		•	•
Devolutions Password Server	•	•	•	•	•	•		
RoboForm	•	•	•	•	•		•	•

Password Manger	Features					
	Security Audits	Import Password	Export Password	Single Sign-On	Password Sharing	Cloud-Based
LastPass	•	•	•	•	•	•
KeePass		•	•		•	
1Password	•	•	•		•	•
Dashlane	•	•	•		•	•
Passwordstate	•	•	•		•	
Keeper	•	•	•	•	•	•
Sticky Password		•	•		•	•
Devolutions Password Server	•	•	•	•	•	
RoboForm	•	•	•	•	•	•

However, the shortcomings of password managers have also been widely documented (see, for example, McCarney [68]). Passwords stored only on the user’s platform restrict user mobility, since they will not be available when a user switches, for example, from use of a laptop to a tablet or phone. However, if passwords are stored remotely ‘in the cloud’, then there is a danger of compromise through poorly configured and managed servers. Sadly there are real-world examples of compromises of such password managers [17, 55, 72, 75, 61].

These issues have led some authors to consider another, related, category of techniques designed to help improve the security of password-based authentication, namely password generators. These schemes, introduced immediately below, avoid the need to store individual passwords long-term.

2.5 Password Generators

A *password generator* is a client-side scheme which generates (and regenerates) site-specific strong passwords, with the minimum of user input. These schemes operate by generating site-specific passwords on demand from a combination of inputs, including those supplied by the user and those based on the nature of the site itself. A number of individual schemes of this type have been proposed but, apart from a brief summary by McCarney [68], they have not been studied in a more general setting. Password generators are one of the main focuses of this thesis, and we provide a more detailed introduction to the prior art in Chapter 3.

2.6 Password Recovery

Web password recovery, enabling a user who forgets their password to re-establish a shared secret with a website, is very widely implemented. Most websites requiring users to create an account support password recovery, enabling legitimate users who forget their password to continue to use the website. In order to support password recovery, a website will typically gather a range of information about the user, which can be divided into two main categories:

- *personal information*, i.e. information about the individual that may be used for a range of purposes apart from password recovery;
- *recovery information*, i.e. information used only for password recovery.

The types of information established purely for password recovery purposes vary widely, depending on the detailed operation of the recovery process. We can identify

the following general categories.

- **Recovery authentication information**, i.e. information that can be used to authenticate the user, e.g. answers to security questions.
- **Recovery contact details**, i.e. special contact details, such as an email address or phone number, used exclusively for password recovery. Example of such details include an email address.

Password recovery forms the second main focus of this thesis, and we provide a more detailed introduction to the prior art in Chapter 4.

2.7 Conclusions

In this chapter we provided an overview of user authentication, including the definition and lifecycle of user authentication; we also discussed the use and vulnerabilities of passwords. Finally we introduced the notions of password managers, password generators and password recovery.

Chapter 3

Password Generators

3.1 Introduction

This chapter is concerned with *password generators*, i.e. schemes designed to simplify password management for end users by generating site-specific passwords on demand from a small set of inputs, at least one of which is secret. Note that the term has also been used to describe schemes for generating random or pseudorandom passwords which the user is then expected to remember; however, we use the term to describe a system intended to be used whenever a user logs in and that can generate the necessary passwords on demand and in a repeatable way.

The remainder of the chapter is organised as follows. Section 3.2 lists the main components of a password generator, and Section 3.3 then gives an overview of previously proposed password generator schemes. Finally, in Section 3.5, we discuss the shortcomings of existing schemes; this provides the motivation for the research described in Part II of this thesis.

3.2 A Definition

A *password generator* is a client-side scheme which generates (and regenerates) site-specific strong passwords on demand, with the minimum of user input. These schemes operate by generating site-specific passwords from a combination of inputs, including those supplied by the user and those based on the nature of the site itself. A number of individual schemes of this type have been proposed but, apart from a brief summary by McCarney [68], they have not been studied in a more general setting. One well-known example of a password generator is PwdHash¹.

¹<https://pwdhash.github.io/website/>

A password generator is typically implemented as functionality on an end-user platform to support password-based user authentication to a remote server (assumed to be a website, although most of the discussion applies more generally). This functionality generates, on demand, a site-unique password for use in authentication.

The password is generated as a combination of a number of values, where the exact choice of values, the method used to combine these values, and the method used to format the output varies widely between schemes (as is made clear in Section 3.3). However, in each case at least one of the inputs must be secret to ensure the generated password is secret, and at least one of the inputs must be site-specific, to ensure the password is unique to that site. A general model for password generators is presented in Chapter 5.

Perhaps the most significant difference between a password generator and a password manager is that, while password managers store user passwords, password generators generate them on demand. This potentially eases cross-platform portability whilst avoiding the risks associated with a locally or cloud-located repository of passwords. A more detailed comparison of their relative merits has been provided by Bonneau et al. [11] and McCarney [68].

3.3 Previous Work

We next review previous work on the design and properties of password generation schemes. The work is presented in chronological order of publication.

- The *Site-Specific Passwords (SSP)* scheme proposed by Karp [54] in 2002/03 is one of the earliest proposed schemes of this general type. SSP is a stand-alone application that generates a site-specific password by combining a long-term user master password (referred to by Karp as a user password) and an easy-to-remember name for the website, as chosen by the user. These two inputs are concatenated and hashed using the MD5 algorithm to produce a 16-byte string, which is then converted to ASCII by Base64 encoding and truncated to 12 characters. The generated password can be then typed manually into the password field of the web site. A screenshot of the SSP window is shown in Figure 3.1.



Figure 3.1: Site-Specific Passwords (SSP),[54]

SSP suffers from some obvious shortcomings, as discussed by the author, who also proposes possible remedies [54]. The main shortcomings of SSP are listed below.

- The site name can be easily guessed, and if the user-chosen password is weak then an attacker may be able to generate the site password.
 - The user needs to remember the chosen name for each site.
 - The generated password may not match the password policy requirements of the website, which could make the system unusable for many sites.
 - The need to enter the password manually means that the system offers no protection against phishing attacks. The need for manual password entry arises from the fact that SSP is implemented as a stand-alone application, preventing automatic password entry.
- *PwdHash*, due to Ross et al. [77], is a browser extension (for Firefox, Chrome and Opera) that generates a site-specific password by combining a long-term user master password, data associated with the web site (the site domain name), and (optionally) a second global password stored on the platform. The site-specific password is then generated by hashing the concatenation of the inputs. The hash is implemented using a Pseudo Random Function keyed by the password.

PwdHash incorporates functionality to block JavaScript-based attacks, e.g. keyboard monitoring, domain rewriting and password reflection attacks. This protection can be enabled either by prefixing an entered password with @@ or by pressing the F2 key. Since PwdHash is a browser extension, it automatically has access to the site domain name, and uses this as part of the input to generate the password. This protects against phishing attacks. Figure 3.2 shows the PwdHash window.



Figure 3.2: PwdHash, [77]

PwdHash (along with ObPwd) is one of the few active password generator schemes which is maintained by its developers. A web version is available at <https://pwdhash.github.io/website/>. The web version is intended to address the portability issue; however, use of the web version potentially gives access to the user passwords to the operators of the pwdhash site, giving rise to a possible trust issue.

Other possible limitations are discussed by Ross et al. [77], including possible DNS Attacks. Chiasson et al. [14] conducted a usability study of two password managers, one of which was PwdHash. This study revealed that users had an incorrect and incomplete mental picture of PwdHash. This was due to the interface design which can lead to security issues; for example, users are required to prefix the first password with @@, and they (falsely) assumed that subsequent further

passwords will be protected too. Another issue is the lack of feedback from the PwdHash interface, leaving users unsure whether the task completed successfully and their password is protected.

- The 2005 *Password Multiplier* scheme of Halderman, Waters and Felten, [39], is an extension to the Mozilla Firefox web browser which computes a site-specific password as a function of a long-term user master password, the web site name, and the user name for the web site concerned. This scheme use two levels of iterated hash computations. The first level is executed when the user first uses platform (i.e. it is performed once per user per system). This computation takes around 100 seconds and the result is cached for future password computations. The second level computation yields the site-specific password by combining the input from the first stage of computation with the account name for which the user wants to generate a password.

After the site-specific password has been generated, it is automatically copied to the targeted website. Figure 3.3 shows the use of Password Multiplier to generate a password for Amazon.



Figure 3.3: Password Multiplier, [39]

Two key shortcomings of Password Multiplier are handling password changes and

platform mobility. To handle the password change issue, Halderman et al. [39] proposed use of an index value alongside the account name. Password change could be achieved by incrementing this password index. The index value could be stored with the name of the account, and if the user wishes to switch to a different platform he/she will only need to input the index value.

A further possible issue is that the passwords created by the system are of a fixed format, i.e. eight alphanumeric characters; this could breach the password policy of some websites.

- Wolf and Schneider’s 2006 *PasswordSitter* [93] generates a site-specific password as a function of a long-term user master password, the user identity, the application/service name, and some configurable parameters, as shown in Figure 3.4. After generation of a password, PasswordSitter automatically copies the generated password to the clipboard; the user then has to copy it to the website.

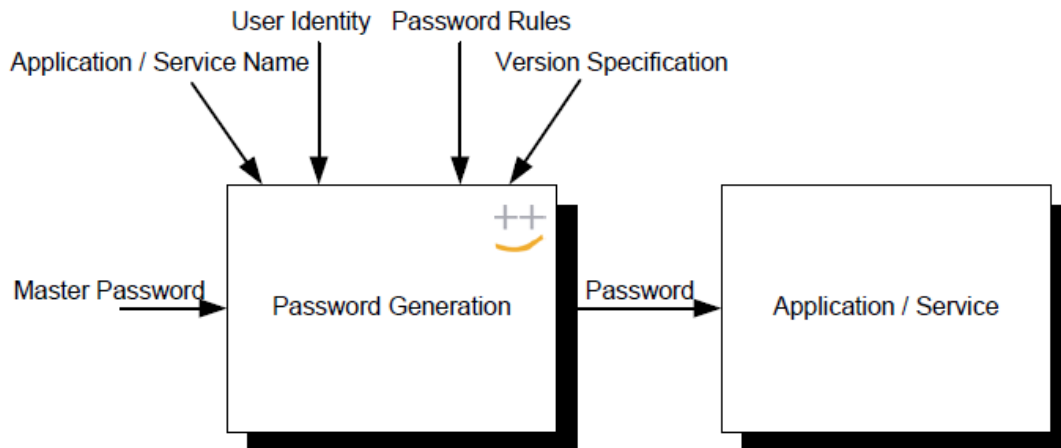


Figure 3.4: Password generation by PasswordSitter, [93]

PasswordSitter combines the inputs using AES encryption, with a key derived from the master password. The application/service password is generated from the encryption result by applying the user-specified password rules. After the password has been generated it transferred to the application or service for authentication.

PasswordSitter incorporates a server to store user data (*the user profile*), including the user identity, the application or service name, the version specification,

and the password generation rules. The user profile is available to download on demand to enable the generation of passwords on multiple devices. The user profile is encrypted using AES employing a key derived from the master password, ensuring the profile is only accessible to the legitimate user. The master password is stored on and never leaves the user platform, and hence the server does not need to be trusted; however, this means that compromise of a user platform puts the security of all the passwords at risk.

- *Passpet*, due to Yee and Sitaker [94] and also published in 2006, takes a very similar approach to SSP, i.e. the site-specific password is a function of a long-term user master password and a user-chosen name for the web site known as a *petname*. Each petname has an associated icon, which is automatically displayed to the user and intended to reduce the risk of phishing attacks.

Passpet uses a combination of techniques: password hashing, petnames, password strengthening, and UI customisation. User-assigned site labels (petnames) help users securely identify sites to mitigate phishing attacks. Password-strengthening measures defend against dictionary attacks. Customising the user interface defends against user-interface spoofing attacks. Figure 3.5 shows the Passpet user interface.

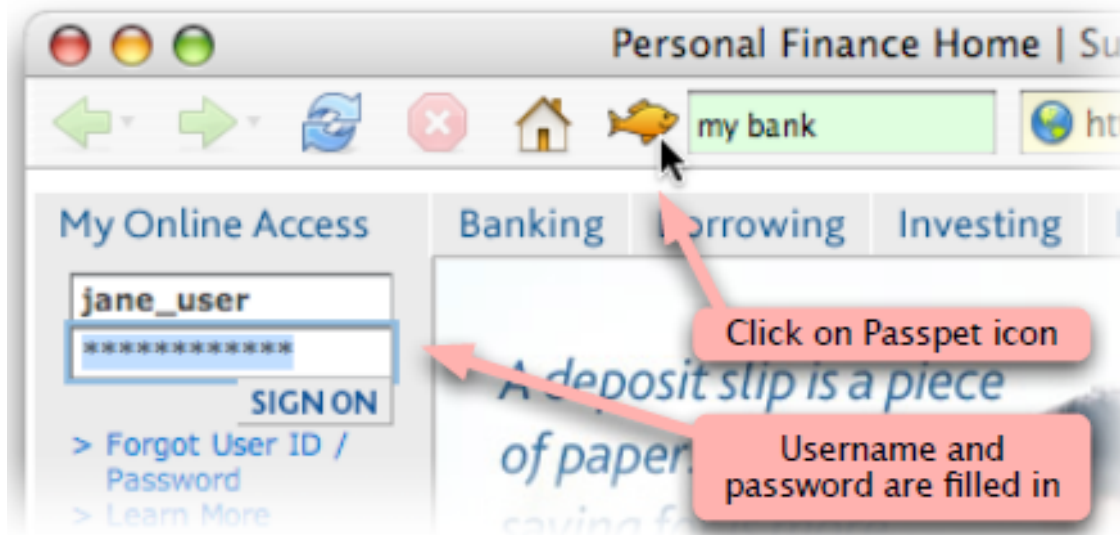


Figure 3.5: PassPet operation, [94]

- *ObPwd*, due to Mannan et al. [8, 65, 66, 67], first surfaced in 2008. It takes a

somewhat different approach by generating a site-specific password as a function of a user-selected (site-specific) object (e.g. a file), together with a number of optional parameters, including a long-term user password (referred to as a *salt*), and the web site URL. The goal of ObPwd is to use content meaningful to the user to generate a high-entropy password, where the user simply needs to remember which object is associated with a particular site. To generate a password, the user selects a memorable object which must exceed a minimum size (e.g. 160 bytes) to ensure a high-entropy password. The ObPwd tool then hashes the chosen object (truncated if necessary, to avoid excess computations), concatenated with the optional long-term user password. The output is then converted to a text string of user-selectable length. The password needs to be manually copied to the desired webpage, since ObPwd is a stand-alone application.

Figure 3.6 represent the steps for password generation in ObPwd, along with an example.

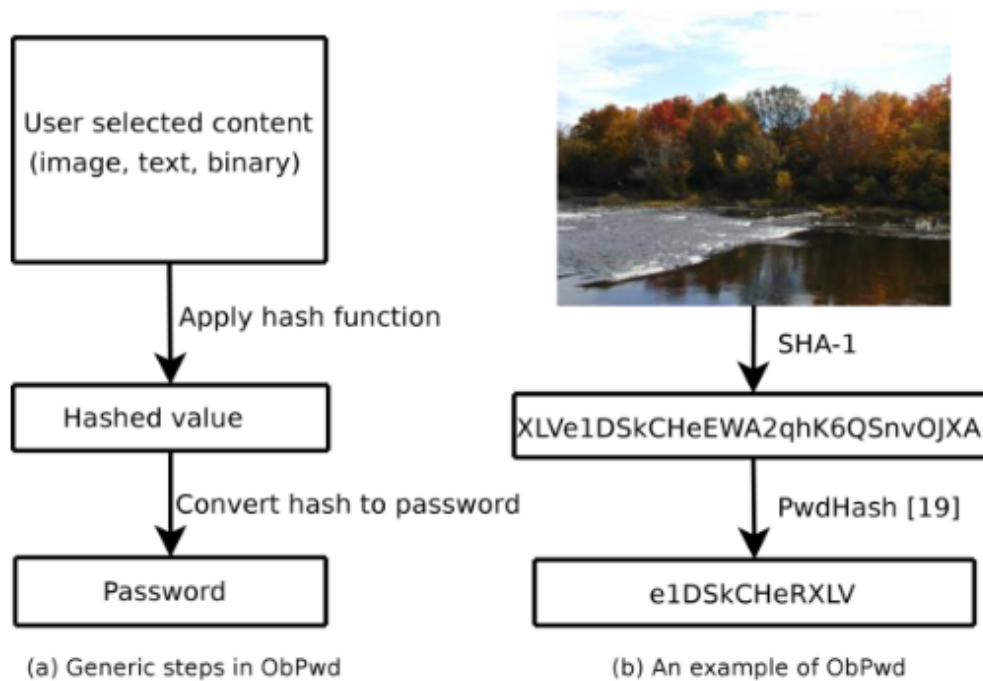


Figure 3.6: ObPwd operation, [66]

A small scale user trail found that ObPwd has good usability, with excellent memorability, acceptable login times, and very positive user perception. However,

as the authors note [67], ObPwd has a number of potential usability and security issues.

- ObPwd does not protect against long-term compromise of the user platform (of course, this is true of all schemes in which passwords are entered via the user device).
 - If a digital object used to generate a password is changed, then the password will no longer be the same. This means that the user must be careful to choose objects that are unlikely to be modified. Related problems arise if the object is not available on the platform currently in use.
 - There is a danger that an attacker might guess which object is being used, e.g. if a user chooses images as digital objects. Of course, deployment of the (optional) long-term user password as a second input to password generation would help mitigate such an attack.
- *PasswordLess Password Synchronization (PALPAS)*, described in Horsch’s 2015 paper, [45], generates passwords complying with site-specific requirements using server-provided password policy data, a stored secret master password (the *seed*), and a user-specific secret value (the *salt*) that is synchronised across all the user devices using the server. Figure 3.7 summarises password generation in PALPAS. The password is computed in two stages. First, a cryptographically secure deterministic Pseudorandom Bit Generator (PGR) generates a pseudorandom value using a seed and a salt value. The pseudorandom value and the password policy are input to the password generator to produce a site-specific password. The salt used in the first step serves two purpose: use of a different salt for each site ensures that each password is distinct, and password change is enabled simply by changing the salt.

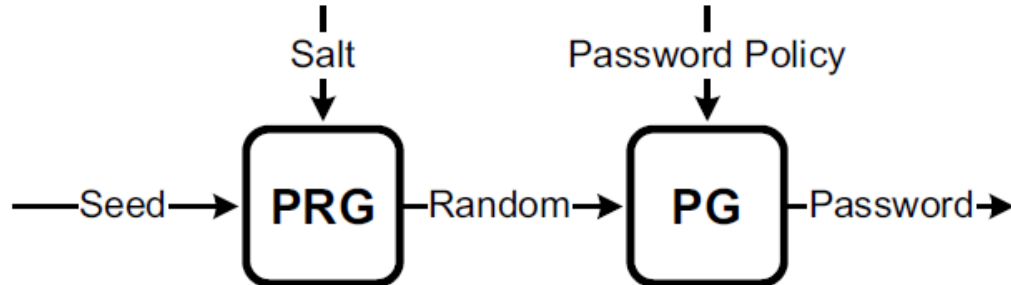


Figure 3.7: PALPAS password generation, [45]

Apart from the above published work, a number of password generator applications exist, some of which are implemented as browser extensions and others as phone apps. The following schemes are available as browser extensions.

- *RndPhrase*² is a Firefox add-on and web-based password generator. It generates site-specific passwords as a function of a predefined salt (unique per user), the host name, and a user-entered password. The user only needs to remember the password.
- *PwdHash port*³ is an Opera add-on based on PwdHash.

Of course, there are many apps and browser extensions which simply generate random or pseudorandom passwords, such as the Android phone apps Advanced Password Generator⁴ and Password Generator⁵. Although they are commonly referred to as password generators, they are not password generators in the sense used in this thesis since the user is responsible for remembering or storing the passwords after they have been generated.

3.4 A Taxonomy

In 2013 McCarney [68] provided a taxonomy and a comparative evaluation of password management systems (including both password managers and password generators)

²<https://rndphrase.appspot.com/>

³<https://addons.opera.com/en-gb/extensions/details/pwdhash-port/>

⁴<https://goo.gl/MF0z1D>

⁵<https://goo.gl/SNVtJY>

using the Usability-Deployability-Security (UDS) framework of Bonneau et al. [11].

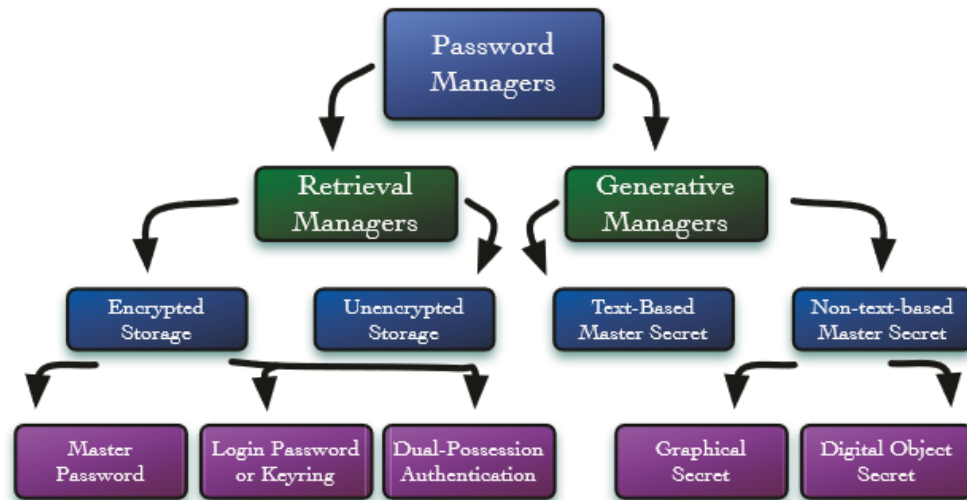


Figure 3.8: Password management systems: a taxonomy (McCarney [68])

3.5 Shortcomings of Existing Techniques

We conclude this chapter by observing certain fundamental problems that affect all (or almost all) previously proposed password generators. These issues motivate the design of a novel scheme that we propose later in this thesis, which incorporates novel features designed to overcome these problems.

Setting and updating passwords If a user is already using the password generator when newly registering with a website, there is clearly no problem — the user can simply register whatever value the system generates. However, if the user has selected and registered passwords with a range of websites before starting use of the password generator, then all these passwords will need to be changed to whatever the password generator outputs. This could be highly inconvenient if a user has established relationships with many sites, and could present a formidable barrier to adoption of the system. Somewhat analogous problems arise if a user decides to change a website password, e.g. because the site enforces periodic password changes. The only possibility for the user will be to change one of the inputs used to generate the password, e.g. the object (if a digital object is used as an input) or a user site name. Password change could even be impossible if

the user does not choose any of the inputs used to generate a password.

Using multiple platforms If a user employs multiple platforms, e.g. a desktop and a smart phone, then problems will arise if any locally-stored configuration data is used.

Password policy issues A further general problem relates to the need to automatically generate passwords in a site-specific form, a problem not satisfactorily addressed by any of the previously proposed schemes except PALPAS. Some existing schemes have the option for the user to customise a generated password, but the user has to identify the requirements for the website manually and configure the options accordingly. Automatically generating a password tailored to meet a website's specific requirements has been explored extensively by Horsch and his co-authors [44, 45, 46].

3.6 Conclusions

In this chapter we provided an overview of password generators including a definition, examples of existing schemes and a description of the main components of a password generator. Finally we discussed key shortcomings of previously proposed password generator schemes.

Chapter 4

Password Recovery

4.1 Introduction

Despite their widely-documented limitations, passwords remain very widely used for user authentication. However, although passwords are meant to be memorised, humans often forget or mislay them. In some contexts this is easily managed; for example, in an office environment, a user who forgets their password for access to a multi-user system can simply see a system administrator, who authenticates them and issues a new password. However, for web authentication, the main focus of this thesis, it is clearly not so simple.

To address this problem, most websites that require users to create an account support password recovery, enabling legitimate users who forget their password to re-establish a shared secret, thereby enabling them to continue to use the website. This can be achieved in many ways, including use of a pre-registered email address or mobile phone number, and/or involving other pre-set fall-back means of authentication. However, many of these techniques introduce vulnerabilities which may enable an impostor to falsely change a password, either causing a denial of service or, in the worst case, enabling the impostor to authenticate as the user.

The remainder of this chapter is structured as follows. In Section 4.2 we review existing work examining the security issues arising from a range of possible approaches to password recovery. We also briefly look, in Section 4.3, at certain issues which have not been previously examined, motivating the research described in Part III of this thesis. Finally, the chapter concludes in Section 4.4.

4.2 Previous Work

Many authors have looked at web password recovery, mostly focussing on particular classes of recovery system or the shortcomings of widely used approaches.

- Several authors have looked at ways to securely store backup copies of passwords on a client device; this is outside the scope of password recovery as we define it, since the web service itself is not involved. We briefly mention two papers of this type. Ellison et al. [25] describe how a locally stored copy of a password can be held in encrypted form, protected using answers to personal questions; the encryption key is derived from these questions in such a way that correct answers to k of n questions enables it to be reconstructed, allowing the user to forget some of the answers. Somewhat analogously, Frykholm and Juels [31] propose a provably secure technique for fault-tolerant password recovery; a secret password is stored protected by a collection of low entropy secrets, such that recovery is possible with only a subset of these secrets. This again enables the user to maintain secured backup copies of passwords, and hence, like the Ellison et al. scheme, is outside the scope of this thesis.
- Chmielewski et al. [15] focus on what they call client-server password recovery. They propose a series of protocols that allow a user to automatically recover a password from a server using partial knowledge of the password, and prove their security in a formal model. This can be regarded as a contribution to theoretical cryptography, rather than as a practical solution to the everyday problem of web password recovery.
- Mannan et al. [64] propose a scheme for password recovery rather different from many commonly used approaches. They propose that websites maintain copies of user passwords encrypted under a public key for which the user holds the private key on a personal mobile device (PMD). When the user invokes the recovery service, the website sends the encrypted copy to the user, who uses his/her her PMD to decrypt it.
- Kharudin et al. [57] describe a graphical user authentication method and propose its use for password recovery. However, the main focus is on the authentication technique rather than on the password recovery process.

All the prior art we have so far described, whilst relating to password recovery, falls outside our scope here, since the proposals are either independent of the web server or require major changes in how clients and servers interact. However, as we describe next, some authors have addressed the problem we consider here.

- A 2016 study by Stavova et al. [84] examines the usability of two password recovery techniques, namely backup codes and the use of trusted associates (social authentication). They examined a particular scheme where the backup value is stored as a QR code to address the issue of backup codes being forgotten and/or written in clear text. They also considered a case where the account holder and a trusted associate can each retrieve a password share from a call centre, where the shares must be combined to obtain the password to get access to the service. The two approaches to password recovery that were studied can be summarised as follows.

- *QR-code-based recovery.* This approach simplifies password recovery for the user by storing a one-time password in the form of QR code. Each user is equipped with a printed copy of a QR code containing a one-time password generated by the website. If a user forgets his/her password, the user scans the QR code and submits it with the username to the website to initiate password recovery.
- *Trusted party based password recovery.* This approach relies on social authentication, where the user nominates a trusted person to help the user regain access to his/her account in the event of password loss. When a user registers for the service, he/she is asked for a trustee phone number for password recovery. If a user forgets his/her password, he/she calls the client centre and if the registered phone number for the user matches the registered number, the user is given part of a password recovery code. The user must then ask his/her chosen trustee to call the client centre to get the other half of the code. The trustee then calls the client centre to obtain the other half of the recovery code, with the registered phone number being used to authenticate him/her. If successful, the trustee will get the other half of the code and can send it to the user. After getting the code from the trustee, the user now supplies the two halves of the recovery code along with his/her username to complete password recovery. The process is illustrated in Figure 4.1.

Stavova et al. conducted a study of 186 student participants to compare the usability of the two techniques. The study revealed that users preferred QR password recovery and it takes less time to perform.

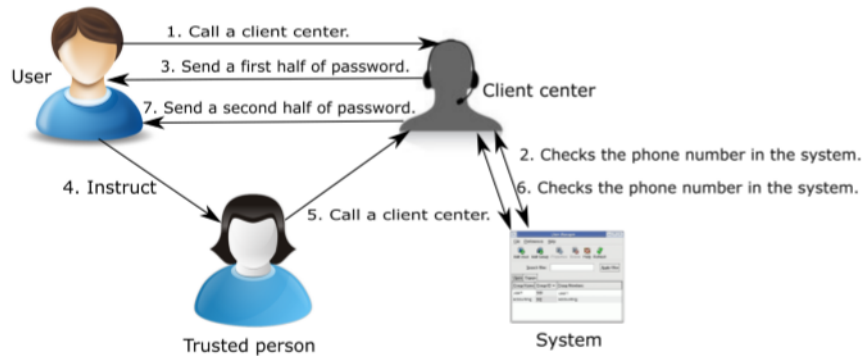


Figure 4.1: Trusted party based recovery, [84]

- A number of authors have examined the security of challenge questions, a secondary means of user authentication widely used in real-world password recovery. In 2008, Rabkin [74] examined leaks of answers to these questions via social media. In the following year, Just et al. [52] performed extensive surveys of user behaviour to analyse the relative security offered by various questions. In parallel work, also published in 2009, Schechter et al. [80] looked at the security properties of security questions as a secondary means of user authentication in the context of password recovery. Finally, in 2015, Bonneau et al. [10] reported on a study using a real-world data set to examine the security and memorability of security questions.
- Guri et al. [38] point out how details revealed during widely-used password recovery procedures for several major Internet services, including Gmail, Facebook, and Twitter, can be used to learn potentially sensitive personal user information. This information includes the full (or partial) email address, phone number, friends list, and street address, as shown in Table 4.1. They examined a range of scenarios and demonstrated how the details revealed in the password recovery process could be used to deduce more focused information about users.

SERVICE	INFORMATION
Facebook	Last two digits of phone number
	Parts of associated email address
	Subsets of users friends (randomly generated)
Gmail	Last two to three digits of phone number
PayPal	Last three digits of phone number
	Parts of associated email address
Twitter	Parts of email address
	The first two letters of email username
	The first letter of the email domain name
Yahoo!	Last two digits of phone number
Microsoft	Last two digits of phone number

Table 4.1: Password recovery privacy violations (Guri et al. [38])

- A recent study by Gelernter et al. [33] describes a possible man-in-the-middle (MitM) attack on password recovery by using the registration process for a malicious MitM website to gather the information needed (e.g. answers to personal questions) to conduct recovery for a different website. The operation of the attack is summarised in Figure 4.2.

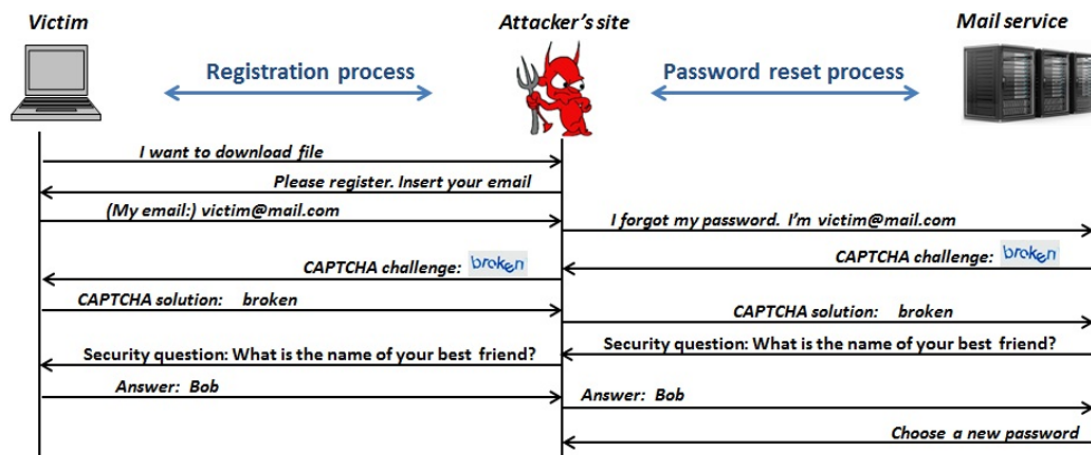


Figure 4.2: MITM attack on password recovery (Gelernter et al. [33])

- A recent study by Li et al. [60] investigated account recovery at popular websites by examining their recovery protocols. Through extensive analysis of the security

features of those websites, they observed that 92.5% of these websites rely on emails to reset user passwords and 81.1% of user accounts at the websites reviewed can easily be compromised by mounting an email recovery attack. In addition, many websites have not taken serious actions to protect recovery emails, leading to a single point of failure. To mitigate the issues they identified, they proposed a mechanism to provide protection for account recovery emails. That is, whilst related to the work we describe later in this thesis, it goes beyond the scope of the work described here by proposing additional mechanisms that need to be implemented by end users.

4.3 Issues with Password Recovery

As can be seen from the previous section, there is relatively little prior art examining the security of real-world password recovery. However, there are very well-known examples of real-world attacks exploiting password recovery to gain unauthorised access to the accounts of well-known individuals. One example of such an attack¹ involved Sarah Palin (the US vice-presidential candidate in 2008). Her personal Yahoo! email was hacked using a password recovery feature in Yahoo!. The email hack occurred in September 2008, during the US presidential election campaign. The hacker, David Kernell, gained access to Palin's account by looking up personal details, such as her high school and birth date, and using Yahoo!'s account recovery for forgotten passwords. Leaked data, including emails and pictures, was posted on WikiLeaks².

The (in)security properties of challenge questions is probably the only topic that has been examined reasonably thoroughly. The penultimate piece of work discussed in Section 4.2 is significant, and reveals major security issues arising with SMS-based password recovery; in particular, the authors performed an analysis of real-world use of SMS-based password recovery and discovered a number of serious issues.

However, email-based password recovery is also very common, and very few authors appear to have examined this previously. Whilst many of the issues are the same as for SMS-based recovery, some are not, and this observation has motivated much of the work in Chapter 9, where the results of a detailed examination of real-world email-based password recovery are reported. This analysis has been used to help develop a set of recommendations for the design of password recovery emails.

It is also interesting to observe that none of the previous research has taken a high-level look at what password recovery involves, and the various options for the steps in

¹https://www.theregister.co.uk/2008/09/17/anonymous_hacks_sarah_palin/

²https://wikileaks.org/wiki/Sarah_Palin_Yahoo_account_2008

this process. This gap has been addressed by the development of a general model for password recovery, which is presented in Chapter 8. This general model helps identify where security issues may lie, and also what options are available to the designers of a password recovery system.

4.4 Conclusions

In this chapter we reviewed existing work examining the security issues arising from a range of possible approaches to password recovery. In Section [4.3](#) we examined certain issues which have not been previously examined, motivating the research described in Part III of this thesis.

Part II

Password Generation

Chapter 5

Password Generators: Old Ideas and New

5.1 Introduction

In this part of the thesis we consider ways in which password generators can be developed which avoid the shortcomings of previously proposed schemes that were identified in Chapter 3. We do this by:

- providing a general model within which a range of password generation schemes can be analysed;
- using this general model to propose new techniques for enhancing the operation of password generation schemes;
- giving a detailed specification of a new password generation system, AutoPass, incorporating the novel techniques and the best features of existing schemes (see Chapter 6);
- reporting on the results of practical trials using a proof of concept implementation of AutoPass (see Chapter 7).

The rest of this chapter is organised as follows. Section 5.2 introduces a general model for *password generators*. This is followed in Section 5.3 by a review of the options for the chief components of the model, referring to existing example schemes. Section 5.4 then addresses the advantages and disadvantages of these options. In Section 5.5 we consider novel enhancements to the operation of the system which mitigate some of the identified disadvantages. The lessons from our assessments, together with

these novel enhancements, are incorporated into a design for a novel scheme, AutoPass, described in detail in Chapter 6.

5.2 The Model

A variety of password generator schemes have been proposed in recent years — this chapter focusses on the general properties of such schemes, and the various options for their operation. We start by presenting a general model for such schemes, which we then use as a framework for analysing possible scheme components.

5.2.1 Operation

As observed in Chapter 3, a password generator is functionality implemented on an end-user platform to support password-based user authentication to a remote server (assumed to be a website, although most of the discussion applies more generally). This functionality generates, on demand, a site-unique password for use in authentication. Clearly this password also needs to be available to the website authenticating the user — the nature of the *registration* step, in which the password is set up, is discussed further in Section 5.2.2 below.

A password generator has the following components.

- A set of *input values* is used to determine the password for a particular site. Some values must be site-specific so that the generated password is site-specific. The values could be stored (locally or online), based on characteristics of the authenticating site, or user-entered when required. Systems can, and often do, combine these types of input.
- A *password generation function* combines the input values to generate an appropriate password. This function could operate in a range of ways depending on the requirements of the website doing the authentication. For example, one website might forbid the inclusion of non-alphanumeric characters in a password, whereas another might insist that a password contains at least one such character. To be broadly applicable, a password generation function must therefore be customisable.
- A *password output method* enables the generated password to be transferred to the authenticating site. This could, for example, involve displaying the generated password to the user, who must then type (or copy and paste) it into the appropriate place.

All this functionality needs to be implemented on the user platform. There are various possibilities for such an implementation, including as a stand-alone application or as a browser extension or plug-in. Each of these aspects of the operation of a password generator are discussed in greater detail in Section 5.3 below.

5.2.2 Registration and Configuration

In this thesis we only consider schemes whose operation is completely transparent to the website which is authenticating the user. As a result, the ‘normal’ website registration procedure, in which the user selects a password and sends it to the site, is assumed to be used. This, in turn, typically means that the password generation process needs to be in place *before* the registration procedure, or at least that introduction of the password generator requires the user to modify their account password. This potential disadvantage of password generators, together with possible ways of avoiding the need to change passwords, is examined in Sections 5.4.5 and 5.5.1 below.

There is a potential need for a password generator to store configuration data. Such data can be divided into two main types:

- *global configuration data*, i.e. values unique to the user and which are used to help generate all passwords for that user, and
- *site-specific configuration data*, i.e. values used to help generate a password for a single website, which are typically the same for all users.

Not all schemes use configuration data, although producing a workable system without at least some global configuration data seems challenging. However, the use of configuration data is clearly a major barrier to portability. That is, for a user employing multiple platforms, the configuration data must be kept synchronised across all these platforms, a non-trivial task — exactly the issue addressed by Horsch, Hülsing and Buchmann [45].

5.3 Components of the Model

We next consider in greater detail options for the components of the model.

5.3.1 Inputs to Password Generation

We first consider the input values to the password generation process. The following data input types have been employed in existing schemes.

- A **master password** is a user-specific long-term secret value. This could either be a **user-entered password**, i.e. entered by the user whenever a password is to be generated, or a **stored password**, i.e. a user-specific secret value stored as global configuration data. Note that this could be augmented by use of a (not necessarily secret) **user constant**, i.e. a further global configuration value entered by the user which ensures that the passwords generated by this instance of the scheme are different to those generated by another instance, even if the same master password is used.
- A **site name** is a name for the site that is using the password for user authentication. This could take a variety of forms, including a **user site name**, i.e., a name for a site chosen by a user, all or part of the site's **URL**, or a **site-specific secret**, e.g. a random value associated with the site URL.
- A **digital object** is anything available on the user platform which could be used as input to the password generation process, e.g. a file or a selected block of text on the target website. Typically the user would be expected to use a different object (or set of objects) for each website.
- A **password policy** is information governing the nature of the generated password, e.g. the set of acceptable symbols and/or the minimum length.
- A **server secret** is a secret value held by a supporting server and used as input to all password generation computations.
- A **user name** is a user-specific value held by a supporting server which could be the user account name or a more opaque value (or some combination of the two).

Table 5.1 summarises the sets of data inputs used by existing password generator schemes. Items given in square brackets ([thus]) are optional inputs.

A further possibility would be to provide a local front end for a graphical password system (see, for example, [24, 85]), and to use this to generate a bit-string to be input to password generation. This possibility is not explored further here.

5.3.2 Generating the Password

A number of approaches can be used to combine the various inputs in order to generate a password. They all involve a two-stage process, i.e. first combining the inputs to generate a bit-string, and then formatting and processing the bit-string to obtain a password in the desired format. Typically the target format is a string of symbols

Table 5.1: Inputs to password generation process

Scheme	Input data
ObPwd, [66]	digital object, [URL]
PassPet, [94]	user constant, user-entered password, user site name
Password Multiplier, [39]	user-entered password, user site name
PasswordSitter, [93]	user-entered password, user site name
PwdHash, [77]	user-entered password, URL
Site-Specific Passwords, [54]	user-entered password, user site name
PALPAS, [45]	stored password, site-specific secret, password policy

of a certain length, where each symbol is, for example, numerals only, alphanumeric characters, or alphanumerics together with punctuation.

Possibilities for the first stage include the following.

- **One-level-hash**, i.e. concatenating the various inputs and applying a cryptographic hash-function. An alternative of this one-level type would be to use part of the input as a key, and to then generate an encryption or MAC on the remainder of the input. Examples of schemes using this general approach include PwdHash [77], ObPwd [66] and SSP [54]. PasswordSitter, [93], uses AES encryption as an alternative to a hash-function, where the AES key is derived from the master password.
- A widely discussed alternative is the **two-level-hash**. In this case, one or more of the inputs are concatenated and input to a cryptographic hash-function which is then iterated some significant number of times. The output of this multiple iteration is then concatenated with the other inputs and hashed to give the output. This two-level multiple iteration process is designed to slow down brute force attacks. Examples of systems adopting such an approach include PassPet [94] and Password Multiplier [39].

The main approach to the second stage employs some form of **encoding**, in which the output from the first stage is formatted to obtain the desired password from a site-specific character set, perhaps also satisfying certain rules (e.g. mandating the inclusion of certain classes of character). Some existing schemes, such as ObPwd [66], allow this to be parameterised in order to meet specific website requirements. Horsch et al. [46] go one step further and propose an XML syntax, the *Password Requirements Markup Language (PRML)*, designed specifically to enable such requirements on passwords to be formally specified. Such password policy statements constitute site-specific configuration data.

5.3.3 Password Output and Use

There are a number of ways in which a generated password could be transferred to the password field of a website login page.

- The simplest is **manual copy and paste**, where the password generation software displays the generated password to the user, who manually copies it into the login page. This approach is used by SSP [54].
- A slightly more automated approach is **copy to clipboard**, in which the generated password is copied into the clipboard for future use. For security reasons the password can be made to only reside in the clipboard for a limited period, e.g. in PasswordSitter the generated password is saved to the clipboard for 60 seconds before being deleted [93].
- The simplest approach for the user is probably **automatic copying to the target password field**. This can be done automatically with no user intervention, as is the case for PwdHash in the web page implementation [77] and the ObPwD Firefox browser extension [66]. Alternatively it can require the user to perform an action such as clicking a specific key combination before copying; for example, PassPet requires the user to click on a screen button, [94], and Password Multiplier, [39], requires the user to double click the password field or press *ctrl+P* to trigger password copying.

5.3.4 Approaches to Implementation

Password generation software can be implemented in a range of ways.

- There are a number of advantages to be derived from implementing the password generator as a **browser add-on**, e.g. as a **browser plug-in**, **browser extension** or **signed browser applet**. Many existing password generator schemes adopt this approach, at least as one option, including Password Multiplier [39], PwdHash [77], PasswordSitter [93], and PassPet [94].
- An alternative is to implement the scheme as a **stand-alone application**, e.g. to run on a phone, tablet or desktop. The user would need to install the application. Such an approach is adopted by SSP and Amnesia; ObPwD, [66], is also available as both a browser extension and a mobile app.
- A somewhat different approach is to implement the scheme as a **web-based application**, either running on a remote server or executing on the user plat-

form in the form of dynamically downloaded JavaScript. PwdHash [77] has been implemented as a web application¹.

5.4 Assessing the Options

For each of the main components we assess the main advantages and disadvantages of the possibilities described in Section 5.3.

5.4.1 Inputs

Three main options for the inputs to the password generation process were described in Section 5.3.1. We consider them in turn.

- The use of a **master password** of some kind seems highly advantageous; the main issue is how it is made available when required, i.e. whether to store it long-term in the software as global configuration data or to employ a user-entered value. Both possibilities have advantages and disadvantages. Long-term storage maximises user convenience, but the master password is now at greater risk of exposure and the system is now inherently less portable, not least because the user may well forget the value if it does not need to be entered regularly. Conversely, user entry reduces convenience but possibly improves security, although the entry process itself is now prone to eavesdropping, either visually or using some kind of key-logger. Perhaps the best possibility might be to combine the two, i.e. to use two secrets — one stored long-term on every device employed by the user and the other entered by the user whenever the system is activated. Such an approach is implemented in the PALPAS system, [45], where a user-entered password is employed to generate a key for encrypting and decrypting the locally stored master secret (or *seed*).

The addition of some kind of **user constant**, i.e. a not necessarily secret user-specific value, also seems reasonable. The obvious location for this is as global configuration data, although this again may have an impact on portability.

- The inclusion of a **site name** was the second option considered in Section 5.3.1. The use of such an input is highly desirable since it will make generated passwords site-specific. The site name could be a **user site name**, i.e. a name for the site selected by the user, the site's **URL**, or a value indexed by the URL. One disadvantage of a **user site name** is that it must be remembered (and entered) by

¹<https://pwdhash.github.io/website/>

the user. Use of the **URL** avoids the latter problem by potentially being available automatically to the password generator. It also prevents phishing attacks in which a fake site attempts to capture user credentials for the site it is imitating, since the URL of the phishing site, and hence the generated password, will be different to that of the genuine site, [77]. However, use of the site URL also has issues, [54], since the URL of a site can change without notice, meaning the generated password would also change and the system would fail. This latter issue can be at least partially addressed by using only the first part of the URL.

- The third possibility is use of a **digital object**. Perhaps its main advantage is that it potentially introduces a major source of entropy into password generation [66, 67]. Such an input also offers a way of making passwords site-specific, although it requires users to choose a different object for every site (and it is not clear that all users will do so). However, there are two major disadvantages with such an approach. Firstly, the object used must always be accessible, significantly restricting user choice especially on platforms with a constrained user interface, such as phones. Secondly, the user must remember which object is used with which site, a task which users may find as hard as remembering site-unique passwords (depending on the psychology of the individuals involved).

5.4.2 Generating the Password

As discussed in Section 5.3.2, password generation is typically a two-stage process: first combine the inputs to generate a bit-string, and second use the bit-string to generate a password. The first stage involves either a single level or a two-level hash. The two-level approach has the advantage of offering a limited degree of protection against brute-forcing of a master secret, [14, 39, 94]. The only disadvantage is a slight delay in the password generation process itself, but this can be made small enough to be barely noticeable.

The second **encoding** step is more problematic. Websites have widely differing password requirements and rules. The encoding scheme must generate passwords tailored to site-specific requirements (the password policy). This in turn typically requires the password generator to store site-specific password policies as site-specific configuration data, potentially reducing the portability of the password generator since each instance must be locally configured.

5.4.3 User Interface Operation

Section 5.3.3 describes a range of possible approaches for transferring a generated password to the password field in the website login page.

- The **manual copy and paste** approach is clearly the simplest to implement. However, apart from being the least user-friendly, it has security deficiencies. Most seriously, a user could be tricked by a fake page to reveal their password for a genuine site. There is also a serious possibility of eavesdropping (or ‘shoulder-surfing’).
- The **copy to clipboard** technique is much more convenient for the user, but is still prone to fake website attacks. Also, an attacker might be able to launch an attack to learn the clipboard contents. It would appear to be good practice to restrict the period of time during which the generated password is in the clipboard, as implemented in PasswordSitter [93].
- The most convenient approach for the user is undoubtedly **automatic copying to the target password field**. If the password generator is aware of the URL of the web page (which is likely if it can access the page to autofill the password) then this can mitigate fake website attacks. The major disadvantage relates to implementability — practicality depends very much on how the password generator is implemented (as discussed in Section 5.4.4 below). Such a solution might require the user to perform a specific action to trigger automatic copying of the password; this would have the advantage of giving the user some control over the process.

5.4.4 Implementation

We conclude this assessment of the various password generator options by considering the ways in which it might be implemented.

- The **browser add-on** approach has been widely advocated in the literature, and has a number of advantages. When implemented in this way, a password generator can readily automate key tasks, including detecting the password field, discovering the site URL, and filling in the password field automatically. However, the use of multiple browsers across multiple platforms may cause incompatibility problems, and require the development and use of multiple instances of the scheme. Also, the lack of an easily accessible user interface may also make configuration difficult for non-expert users.

- A **stand-alone application** is the obvious alternative to a browser add-on. One advantage of such an approach compared to a browser add-on relates to the user interface; a stand-alone application is likely to have a richer user interface, easing its use and configuration. However, such an application may be unable to automatically perform some of the tasks which can readily be performed by a browser add-on, such as automated password field detection and password input. Stand-alone applications will also have portability issues, with a different application needed for each platform type.
- The third possibility is a **web-based application**. Such an approach has the great advantage of seamless portability — it would be instantly usable on any platform at any time. Just like a browser add-on, it could also enable automation of key tasks such as URL detection and automatic password completion, [39]. However, there are also serious disadvantages. A usability study conducted by Chiasson et al., [14], revealed that users had difficulty in locating the PwdHash website. Most seriously, the web implementation will potentially have access to all the user's passwords, as well as the values used to generate them. Of course, an application could be implemented, e.g. using JavaScript, to perform all the necessary calculations on the user machine, and not on a remote server — however, the capability would remain for the website to transparently eavesdrop on the process whenever it wished.

5.4.5 Other Issues

Before proceeding we mention certain other usability and security issues which can arise, potentially regardless of the options chosen. These issues were discussed in Chapter 3, but we repeat them here given their importance in motivating the work described below.

- *Setting and updating passwords*, i.e. none of the schemes allow a user to choose their password for a site, and in most cases it is problematic to arrange for a password to be changed.
- *Using multiple platforms* is a problem with many existing schemes, since client-stored secrets need to be migrated between platforms.
- *Password policy issues* exist with many schemes, i.e. ensuring that generated passwords meet site-specific requirements on character set and length.

5.5 Improving System Operation

We next consider a range of ways of addressing some of the problems we have identified. Some of these techniques have already been proposed, although not precisely as we describe them here. In Chapter 6 we consider how these measures might be integrated into a novel system.

5.5.1 Novel Types of Configuration Data

We have already mentioned a range of types of configuration data, including classes of global configuration data, such as master secrets, and categories of site-specific configuration data, such as password policy values (possibly specified in PRML [46]). We now introduce three new configuration data types, whose use can address some of the issues identified in Section 5.4 .

- A *password offset*, a type of site-specific configuration data, can address a range of issues relating to cases where the user wishes to employ a password different to that computed by the password generator. For example, a user may already have a set of well-established passwords which he/she does not wish to change, or a specific password value may be imposed by the website. Additionally, users may wish, or be required, to change their passwords from time to time. As we have already observed, addressing such requirements with a password generation scheme is problematic.

A password offset acts as an input to the second stage of password generation. The first stage generates a bit-string, and the second stage converts this to a password with specific properties, as specified by the password policy. The password offset induces the second stage to generate a specific password value. For example, suppose that a password policy dictates that a password must be a string of lower and upper case letters and numerals, and suppose each such character is internally represented as a numeric value in the range 0–61. After converting the bit-string to a string of alphanumeric characters of the desired length, and given a ‘desired password’ consisting of an alphanumeric string of the same length, the password offset could simply be the character-wise modulo 62 difference between the two strings². Changing a password can now be readily implemented by changing the offset, either to a random value (thereby randomising the password choice), or to a chosen value (if the new password value is to be fixed by the user).

²Such an idea is widely implemented to enable credit/debit card holders to select their own PIN value — see, for example, https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.csfb300/pinkeys.htm#PINvalue.

If implemented appropriately, this offset is not a hugely sensitive piece of data, since it need not reveal anything about the actual password value. Of course, if an ‘old’ password is compromised, and the old and new offsets are also revealed, then this could compromise the new password value.

- It is also possible to envisage a scheme where a password for one site is generated using a different set of input types to those used to generate a password for another site. For example, a password for a particularly mission-critical site (e.g. a bank account) might be generated using a large set of input values, e.g. including a digital object, whereas a password for a less sensitive site could be generated using a master secret and site name only. Such a possibility could readily be captured using site-specific configuration data which we refer to as *password input parameters*.
- A system might also store *password reminders* as site-specific configuration data. For example, when choosing a digital object to generate a password, the user could be invited to specify a word or phrase to act as a reminder of the chosen value (without specifying it precisely). This could then be revealed on demand via the password generator user interface.

5.5.2 Use of A Server

We have already observed that storing configuration data on a user platform creates a major barrier to portability. It also poses a certain security risk through possible platform compromise, although, apart from the master secret, much of the configuration data we have discussed is not necessarily confidential.

The ‘obvious’ solution to this problem is to employ a server to store configuration data, or at least the less sensitive types of configuration information, much as many password managers keep user passwords in the cloud. That is, while it would seem prudent to at least keep a master secret on the user platform, all the site-specific configuration data could be held in the cloud. This type of solution is advocated by Horsch and his co-authors [45, 46].

If the scope of the site-specific configuration data can be kept to non-confidential values, then there is no need for a highly trusted server, a great advantage by comparison with some of the server-based password managers. Also, use of a server need not significantly impact on password availability, since a password generator application could cache a copy of the configuration data downloaded from the server, addressing short term loss of server availability. Loss of network availability would not be an issue,

since in such a case remote logins would in any event not be possible, i.e. passwords would not be needed.

5.6 Conclusions

This chapter has provided a general model for the operation of a password generator, providing the basis for a systematic review of how the components of the model might operate. The various options for implementing the model components were assessed. Novel approaches to addressing issues with current approaches to implementing the model were then described and discussed.

Chapter 6

AutoPass: An Automatic Password Generator

6.1 Introduction

This chapter provides a detailed specification and analysis of AutoPass, a password generator scheme that incorporates concepts introduced in Chapter 5. AutoPass has been designed to address issues identified in previously proposed password generators, and incorporates novel techniques to address these issues. Unlike almost all previously proposed schemes, AutoPass enables the generation of passwords that meet important real-world requirements, including forced password changes, use of pre-specified passwords, and generation of passwords meeting site-specific requirements.

The remainder of this chapter is structured as follows. First, in Section 6.2, we provide a high-level overview of AutoPass. Then, in Section 6.3, we give a detailed specification of AutoPass in the context of the model introduced in the previous chapter; this involves providing detailed descriptions of the system components, the stored data, and the method of password generation. This is followed in Section 6.4 by a detailed specification of its operation. Section 6.5 provides an analysis of the properties of AutoPass, and in particular highlights how it addresses known shortcomings of previously proposed password generators. Then, in Section 6.6, we discuss the security properties and the threat model of the scheme. Section 6.7 describes how the design of AutoPass addresses the issues identified in previously proposed schemes. Finally, the chapter concludes in Section 6.8.

6.2 AutoPass: High-level Specification

AutoPass (from ‘automatic password generator’) is designed to combine the best features of the prior art together with the novel ideas introduced in Chapter 5, particularly those devised to address some of the shortcomings of previously proposed schemes. AutoPass uses most of the types of input given in section 5.3.1 to generate a password, since they all contribute to security in different ways. Following the approach of PALPAS, [45], we also make use of a server to store non-sensitive configuration data, such as website password policies.

Following Section 5.2, to describe AutoPass we must define: (a) the input types, (b) how the password is generated, and (c) how the password is output, together with the implementation strategy. We briefly cover these points in turn; we give a full specification of each of these features in Section 6.3. Since we also propose the use of a cloud service to support AutoPass operation, we also briefly sketch its operation.

- For **inputs**, we propose the use of a **master password**, stored by the system (as global configuration data), and a **user password** (or PIN) to be entered by the user. We also propose use of the first part of the **URL** of the site, where, depending on the implementation, this should also be stored as part of the site-specific configuration and used to retrieve the other site-specific data. The master password can be held encrypted by a key derived from the user password. We also propose the optional use of a **digital object**, where use of this option is indicated in the site-specific configuration data.
- The first stage of **password generation** adopts a two-level hash approach, giving some protection against brute force attacks. The second stage, i.e. **encoding**, uses the AutoPass cloud service to retrieve the password policy for the website being visited (cf. PALPAS [45]); this policy could be encoded using PRML, [46]. It also uses other cloud-stored configuration data, notably the password offset, password input parameters, and password reminders introduced in Section 5.2.2.
- The precise option for **password output and use** depends on the implementation. Where possible, auto-filling the password is desirable; where this is impossible, the copy to clipboard/paste buffer approach is followed.
- **Implementation** as a browser add-on is probably the best option, not least in giving simple access to the web page of the target site, although a range of options may need to be pursued depending on the platform type.

We next consider the AutoPass Cloud Service, which will be required to store two

main types of data. *User-independent data* will be accessed by AutoPass users, and will include non-sensitive site-specific data, e.g. password policies. Even if corrupted by a malicious party, it would at worst cause a denial of service. *User-specific data* will only be accessed by a single user, and includes a range of password configuration data. Although this data is not highly confidential, access to it will need to be restricted to the user to whom it belongs, e.g. via a one-off login process in the local AutoPass application (with access permissions encoded in a cookie stored in the user platform).

Any cloud service has associated risks arising from non-availability; however, this can be addressed through caching. The local AutoPass app should maintain a copy of the data downloaded from the cloud service; since this data is not likely to change very quickly, the cached data should normally be sufficient. To avoid risks arising from fake AutoPass services, e.g. using DNS spoofing, the cloud service could sign all provided data, and the AutoPass app could verify signatures using a built-in copy of the cloud service public key.

6.3 AutoPass: Detailed Specification

We next provide a detailed specification of AutoPass, building on the high-level specification provided in Section 6.2.

6.3.1 Structure

AutoPass has two main parts: the AutoPass server and the AutoPass client. The AutoPass server stores less sensitive user data, e.g. user names and website-specific password policies (specifying the types of password a particular site will accept). The AutoPass client software provides a user interface, and automatically generates site-specific user passwords by combining the specified set of inputs. Some inputs are stored locally and some are stored in the AutoPass server, with which the client software interacts as necessary. Where possible, the generated password is automatically inserted into login forms. Figure 6.1 depicts the AutoPass architecture, showing the main components of the scheme.

6.3.2 Operation

We next describe the detailed operation of AutoPass, based on the high level specification provided in the previous section. We first describe the three main components of the scheme, i.e. the input values, the password generation function, and the output method, together with an initial approach to implementation.

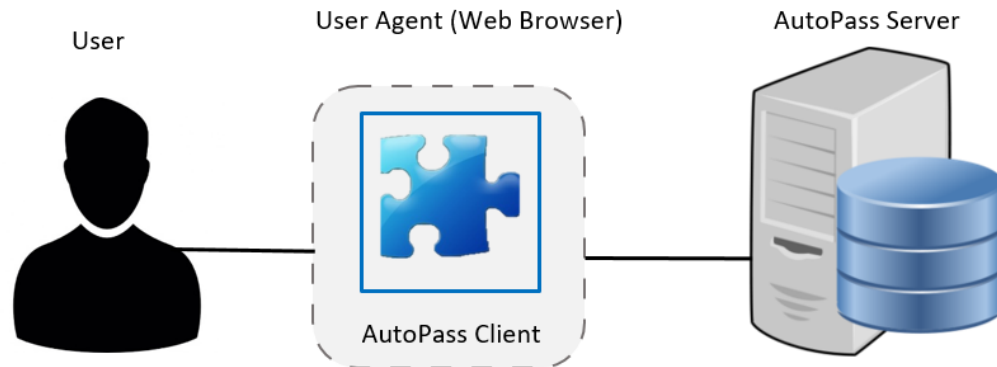


Figure 6.1: AutoPass System Architecture

Input values

AutoPass makes use of a range of types of input, as listed below, incorporating those used in previous schemes.

- A **master password** is a long-term strong password selected by the user or generated by the system. It is stored in encrypted form on the AutoPass server as part of the user-specific configuration data. Since it does not need to be remembered by the user, it could, for example, be a 128-bit random value; the precise choice is implementation-dependent. The user should make a written record of this value when it is initially chosen, and store it securely for backup/recovery purposes.
- The **site name** is the URL of the site for which AutoPass is generating a password. To overcome the issue of changes to URL sub-domains, AutoPass only uses the first part of the URL (i.e. up to the first / character).
- A **password policy** specifies the site-specific requirements for a password (e.g. length constraints and/or minimum numbers of certain classes of character). Many websites enforce highly specific policies, reflecting somewhat ad hoc decisions made by system designers. The policy is specified using the Password Requirements Markup Language (PRML) [46].
- A **digital object** is a text fragment, picture, or audio sample, typically in the form of a digital file. This is an optional input that potentially adds significant entropy to the password generation process, e.g. for use when generating passwords protecting high-value resources. Such objects need to be available on all

user platforms — this means that objects should be selected with care to avoid causing cross-platform mobility issues.

Password generation

The process of combining input values to produce the site-specific password occurs in two stages, as follows.

- The first stage involves combining the input values, including the master password and the URL, to produce a bit string. Following Kelsey et al. [56], this computation involves a two-level hash computation, as follows.
 1. The master password is submitted to a cryptographic hash-function, e.g. SHA-256, [47], that is iterated n times, where n is chosen to be as large as possible without making the client software too unresponsive. The value n can be user-dependent, although in such a case it needs to be held in the server to enable it to be synchronised across all user platforms. The output, e.g. a 256-bit string, is then cached by the client. Since this value is independent of the website, it can be computed once when the client software is started up and cached locally while the client is active.
 2. The 256-bit string is concatenated with the website-specific inputs (the site name and the optional digital object) and hashed once more to yield a site-specific bit string.

The two-level process gives protection against brute force attacks by slowing them down. Suppose an opponent knows a site-specific password and wishes to use this to brute-force the user's master password. Of course, if the master password is a randomly chosen 128-bit string then there is no danger of such a search succeeding, but some users may not select their master passwords to possess high entropy. In such a case, a brute-force search might be practical. Use of a multiply-iterated hash means that any brute force search will involve significantly more computational effort than it otherwise would; however, since the iterated part is only computed once per session, the additional load on the genuine client will be manageable.

- The second stage (encoding) involves constructing a password of the desired form from the bit-string output from the first stage, using the PRML policy specification to ensure the password meets the website-specific requirements. Other possible inputs include the password offset. How encoding operates is discussed in detail in 6.3.4 below.

Figure 6.2 summarises password generation in AutoPass.

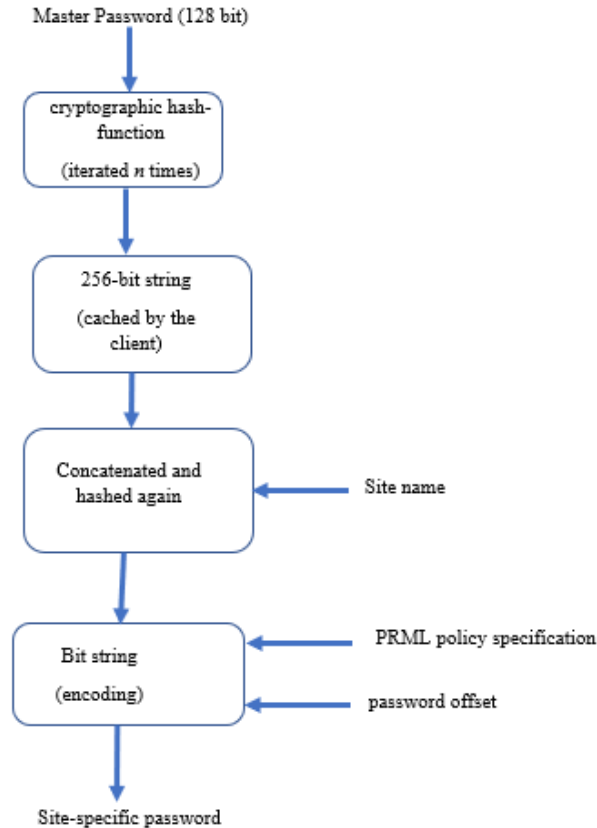


Figure 6.2: Password generation in AutoPass

Password entropy

As discussed above, the inputs to password generation include the master password, the site name, and (optionally) a digital object. If the master password is a 128-bit random string (as recommended), then there will be at least 128 bits of entropy in the input to password generation. As specified in Section 6.3.4 below, the default (minimum) length of passwords is 16 characters. If we assume that the password character set is of size 62 (upper- and lower-case letters and numerals), and the minimum password length is chosen, then there are a $62^{16} \approx 4.8 \times 10^{28} \approx 2^{95.3}$ possible passwords. Assuming the outputs of the hash function used to combined the inputs are distributed in a random-looking fashion, and given that there are at least 2^{128} possible inputs, then each possible password will be approximately equally likely to occur, i.e. password entropy will exceed 95 bits.

Password output and use

This is achieved by automatically copying the generated password to the targeted password field. AutoPass uses secure filling techniques to prevent sweeping attacks [81].

Implementation

We propose to implement AutoPass (at least initially) as a browser add-on. This enables automation of key tasks, including fetching the website URL and inserting passwords into login forms. In the future, for use with platforms not permitting add-ons, we plan to examine stand-alone applications and web-based functionality.

6.3.3 Stored Data

AutoPass needs access to a variety of configuration data, and this data clearly needs to be stored somewhere. There are two possible locations for storage, namely the AutoPass server and the AutoPass client, and AutoPass uses both. The configuration data stored at the server is held long-term, i.e. for the lifetime of the user account; data held on the client may be held either short-term, e.g. for the life of a session, or long-term, i.e. while the software remains installed.

Server-stored Data

The following user-specific configuration data is held at the AutoPass server:

- the user account name;
- an email address for the user;
- the (encrypted) master password;
- the result of applying a cryptographic hash-function to the concatenation of the master password and a fixed string (e.g. a single ‘0’ bit), where the addition of the fixed string ensures that the hash-value held by the server cannot be used to generate user passwords;
- a (salted) hash of the login password (see 6.4.1);
- for each website for which a password has been generated for this user:
 - the (first part) of the URL of the website;
 - the types of input used to generate the password for this site;

- the password offset for this site (see 6.3.5).

The following site-specific configuration data is held at the AutoPass server: (a) the (first part) of the URL of the website; and (b) the password policy of the site, encoded in PRML (see 6.3.4).

Note that the site-specific data could be maintained by a server separate from that used to store the user-specific data. Indeed, since this data is completely non-confidential, it could be provided by a service independent of AutoPass, e.g. the Password Requirements Description Distribution Service (PRDDS) [46], which provides an online interface to meet requests for PRML-based Password Requirements Descriptors (PRDs) for websites identified by their URL.

Client-stored Data

The following data is held short-term on the client, i.e. for the life of a browser session: (a) the login password (see 6.4.1); and (b) a multiply-iterated hash of the master password (see 6.3.1). The AutoPass client also caches recently downloaded password policies.

6.3.4 Use of PRML

The Password Requirements Markup Language (PRML) [46] is an XML-based syntax for use in specifying password requirements, including minimum and maximum lengths, the permissible character set, and minimum required number(s) of specific sub-classes of characters. It has been designed to address the diversity of password requirements arising in practice, and enable password generators to automatically generate site-specific passwords matching password requirements.

A website's PRML specification is one of the two inputs for the second stage of password generation described in 6.3.2, the other being the bit string output from the first stage. The second stage of password generation operates as follows.

1. The size C of the password character set is derived from the PRML specification; we suppose that a mapping is chosen from the set of integers $\{0, 1, \dots, C - 1\}$ to the characters in the password character set.
2. The length L of the password is chosen to be the minimum of 16 and the minimum length prescribed by the PRML policy.
3. The input bit string is converted to a positive integer (by regarding the string as the binary representation of a number), and this number is converted to its

C -ary representation $d_t d_{t-1} \dots d_0$, for some t , where $0 \leq d_i \leq C - 1$ for every i ($0 \leq i \leq t$).

4. The final L digits of the above sequence of numbers, i.e. $d_{L-1} d_{L-2} \dots d_0$, are then converted to characters using the mapping established in step 1.
5. The password is tested to verify that it satisfies the other constraints in the PRML specification. If not, then the input bit string is rehashed and the process is recommenced; otherwise the process is complete.

The above procedure assumes that the length of the input bit-string is significantly greater than $\lceil L \log_2 C \rceil$. Since the likely value of L is 16, and a typical value for C is at most 64, this means that $\lceil L \log_2 C \rceil$ is likely to be less than 100, i.e. much less than the length of the output of a modern hash function such as SHA-256 (which gives a 256-bit output). It also assumes that a random password with characters from the specified password set has a reasonable chance of satisfying the PRML requirements. If this is not true, then a more elaborate second stage algorithm could be devised.

6.3.5 Password Offsets

As noted in 5.5.1, a major issue with existing password generators is that they do not allow a user to choose a password (e.g. to allow continuing use of a password established prior to use of the system), or to change a password without changing the set of inputs. We propose the use of *password offsets* to support these requirements. A password offset works in the following way.

1. A password $d_{L-1} d_{L-2} \dots d_0$ is first generated in the normal two-stage way (as described in 6.3.4), and suppose D is the positive integer which has $d_{L-1} d_{L-2} \dots d_0$ as its C -ary representation.
2. Let the user-chosen password (of length M , say) be encoded as an M -digit sequence $e_{M-1} e_{M-2} \dots e_0$, where $0 \leq e_i \leq C - 1$ for every i ($0 \leq i \leq M - 1$), and suppose E is the positive integer having $e_{M-1} e_{M-2} \dots e_0$ as its C -ary representation.
3. The password offset is simply $E - D$.

When generating a password, if a password offset exists then the password is generated in the normal way and the offset is added; the result will be the C -ary encoding of the desired password. A similar approach can be used for password changes, where

a new password can be generated at random (in accordance with the PRML specification) and the password offset is set to the difference between the new password and the value generated using the standard procedure.

Note that the use of a password offset is to some extent like a cloud-based password manager, in that password-related information is stored by a server. Indeed, it can be regarded as a way of combining the best features of both approaches. What distinguishes an offset-based password generator from a password manager is that use of an offset is optional, and users are only expected to employ it if they wish to either continue to use an existing or externally chosen password or make frequent password changes. Also, the AutoPass server only stores non-sensitive information, unlike the case for a cloud-based password manager.

6.4 Details of Operation

To simplify the description of AutoPass operation, we assume AutoPass is implemented as a browser add-on running on a Windows platform. Alternative implementation scenarios, e.g. as a stand-alone application on a phone or tablet, are likely to be very similar, but may vary in some minor details.

6.4.1 Application Installation

We divide the discussion into two cases, i.e. where a user installs AutoPass for the first time (and creates an account on the AutoPass server), and where a user installs the software and already has an AutoPass account.

First Installation and Account Creation

Suppose a user decides to start using AutoPass, and so installs the client software. Once the AutoPass add-on is installed, the set-up procedure involves the following steps.

1. When the AutoPass add-on is activated for the first time, e.g. by clicking a toolbar button, it first asks the user whether he/she has an existing account. In this case the user indicates that a new account is to be created. This involves the client contacting the AutoPass server, and various registration details need to be completed, as described below.
 - (a) *Login password*: This is chosen and entered by the user, who must memorise it. It serves two main purposes: user authentication to the server and derivation of a key used to encrypt a copy of the master password stored at

the server. After entry of the login password, the client computes a salted hash of the value, which is sent to the AutoPass server as a means of authenticating the user.

- (b) *User name*: The user must select a unique name. The AutoPass server checks that the name is not already in use, and if necessary requests the user to choose a different value. The user name is stored by the AutoPass server, and serves as an identifier for the user.
- (c) *Master password*: This value functions as a cryptographic key, and can be generated by the user or the AutoPass client software (perhaps at the choice of the user). We assume here that it is a 128-bit value, represented as a string of 32 hexadecimal characters. If the client software generates it, it is displayed to the user and the user is advised to keep a copy somewhere secure so that system recovery is possible (see below). The login password is used to generate a cryptographic key, e.g. by hashing a concatenation of the login password and a fixed value. This key is then used to encrypt the master password using an appropriate technique, e.g. AES [48] in an authenticated encryption mode, prior to uploading it to the server. The server retains this encrypted master password for downloading to a client whenever a user logs in. The AutoPass client also generates the result of applying a cryptographic hash-function to the concatenation of the master password and a fixed string (e.g. a single '0' bit), which is sent to the AutoPass server and stored with the encrypted copy; this hash value is used for recovery purposes (see below).
- (d) *Other information*: To allow recovery if a user forgets his/her user name or login password, an email address (or addresses) should also be collected. Other contact details could also be given, e.g. a mobile number. The server holds this information as part of the user account information.

2. After successful user account creation, the user is requested to log in using his or her newly established user name and login password (see 6.4.2).

Installing AutoPass on a Newly Acquired PC

Once an AutoPass account has been established (as described immediately above), the following step is performed to set up AutoPass on a new machine. We suppose that the client software has already been installed.

As previously, when the AutoPass add-on is activated for the first time, it first asks the user whether he/she has an existing account. In this case the user indicates that he/she already has an account. The AutoPass client then asks the user for his or

her user name and login password, and the process continues exactly as in a normal operational session (see 6.4.2).

Recovery

The system needs to provide a recovery mechanism for the case where a user forgets their user name and/or password. We suppose that the client software has a recovery function, which a user can invoke in the event of a forgotten user name or password. We consider the operation of this recovery function for the two cases separately.

- If a user forgets their user name, he/she can request a copy from the server by entering their registered email address. The server checks the email address is registered, and emails the user name for this address to the user.
- If a user forgets their login password, then it cannot be recovered since neither the server nor the client retain a copy. However, if the user has kept a copy of the master password, then system recovery is possible. The user is prompted to enter his or her user name, the master password and a new login password. The new login password is used to generate a key which is used to encrypt the master password, exactly as described above. A salted hash of the newly selected login password, the user name, the encrypted master password, and the result of applying a cryptographic hash-function to the concatenation of the master password and a fixed string (e.g. a single ‘0’ bit) are all sent to the AutoPass server. The server authenticates the user by comparing the master password hash with its stored value, and, if successful, replaces the current encrypted master password, the master password hash and login password hash with the new values. Finally, the server communicates the success of the recovery operation to the AutoPass client, which informs the user.

6.4.2 Operational Sessions

We next consider what occurs when the AutoPass software is activated, e.g. after the host platform has been rebooted. We suppose that the set-up process described in 6.4.1 has already been performed.

1. The user is prompted for his or her user name and login password.
2. The user name is sent to the AutoPass server, which responds with the salt value for its stored copy of the hashed login password for the identified user.

3. The AutoPass client software uses the salt value to hash the login password entered by the user, and the resulting hash-value is sent to the server.
4. The AutoPass server checks that the login password hash is the same as its stored value, and by doing so authenticates the user.
5. The AutoPass server sends back to the client the encrypted master password. The AutoPass server also sends the following information for each site for which the user has created a password using AutoPass:
 - the first part of the URL of the site (this is used as the site identifier);
 - the password policy for the site (in PRML);
 - the set of input types used to generate the password for this site (e.g. whether a digital object is used);
 - the password offset for this site, if it exists;
 - any other parameters used to control password generation for this site.
6. The AutoPass client decrypts the master password using a key derived from the login password, and multiply hashes the master password; the result is cached and the master password can then be deleted. Note that the multiple hashing process takes as input just the master password (i.e. not concatenated with any value) and hence the hash value held by the server cannot be used to compute this value.
7. Once activated, the AutoPass add-on will run continuously in the background, examining each web page to see if it is a login page. It does this by using various heuristics, including looking for the string *input type="password"*.
8. The add-on will then work as required, generating passwords automatically, until the session ends, e.g. when the browser is terminated, at which point the short-term stored configuration stored data, including the multiply-hashed master password, is securely deleted.

Figure 6.3 summaries the AutoPass client and server interactions and information flow.

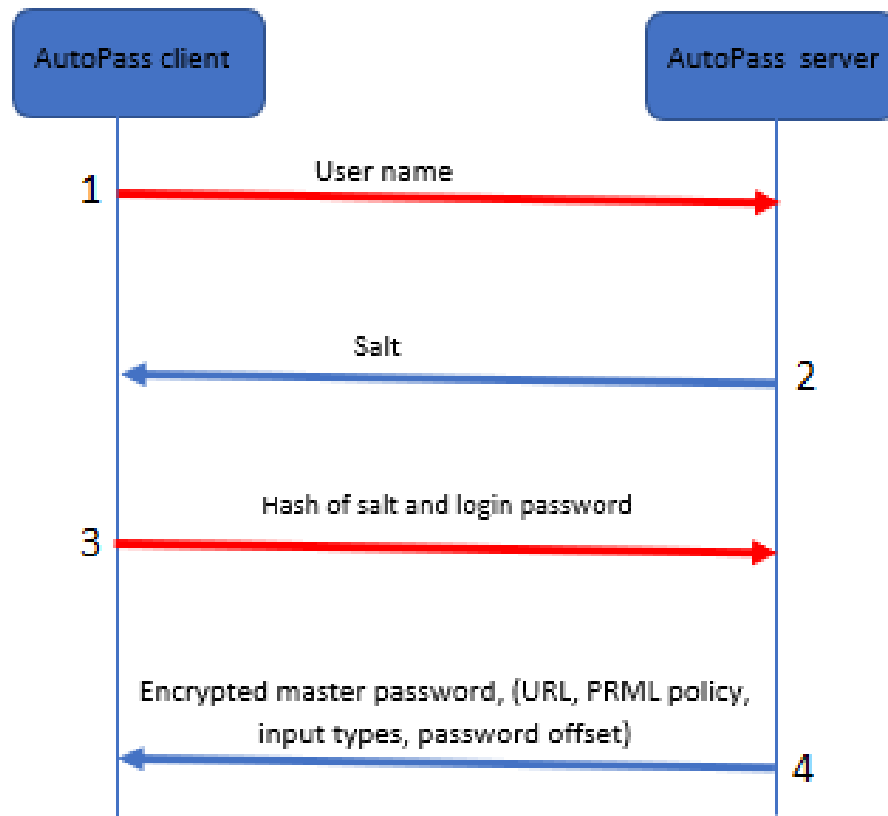


Figure 6.3: AutoPass operational sessions

6.4.3 Use with a Website

When AutoPass is used with a new website, the following procedure is executed. We suppose that the AutoPass software is already executing, i.e. the procedure in 6.4.2 has been followed.

1. If the AutoPass add-on detects a login page, it cross-checks the first part of the site URL with the data from the AutoPass server to determine whether it is a known site. In this case, we suppose that AutoPass has not previously been used to generate a password for this site.
2. The AutoPass add-on then communicates with the user (e.g. via a pop-up) to indicate that it has detected a login page for a website for which a password has not previously been generated, and asks the user whether it would like AutoPass to manage generation of a new password for this site.

3. If the user declines, then the AutoPass add-on goes back to looking for login pages. If the user accepts, AutoPass next asks what types of input the user would like to use to generate the password from amongst those listed in 6.3.1.
4. The user selects the input types; if use of digital objects is selected, the user is also asked to select an object. The AutoPass client assembles the inputs, including the first part of the website URL and the multiply-hashed master password, to be used to generate the site password. The client also offers the user the option to select the password — if the user requests this option then the user is prompted for the pre-chosen value.
5. The AutoPass client retrieves the PRML password policy for this site from the AutoPass server.
6. The password is generated using the procedure specified in 6.3.1 and 6.3.5 and automatically copied to the password field. If the user chose to select the password value, then the appropriate password offset is computed during password generation.
7. The user preferences and the password offset (if appropriate) are sent to the AutoPass server for storage.

The following steps are executed when AutoPass is in everyday use, i.e. after a website has already been set-up. As previously, we suppose that the client AutoPass software is already active.

1. If the AutoPass add-on detects a login page, it uses the first part of the site URL to check whether a password has previously been generated for this site — in this case we suppose it has.
2. The AutoPass add-on now communicates with the AutoPass server to request the site-specific configuration data for this site, including the list of data input types to be used to generate the password, the password policy encoded in PRML, and the password offset (if it exists).
3. The AutoPass add-on then assembles the set of inputs to be used to generate the password; if the user preferences for this site indicate that a digital object is to be used, the add-on prompts the user for the object.
4. The AutoPass add-on generates the password, using the password offset if available, and automatically copies the value to the password field.

6.5 Other Aspects

Client-Server Communications

Whilst the data exchanged between AutoPass client and server is not necessarily highly confidential, some is privacy-sensitive and the integrity of all the data is crucial for correct operation. We therefore propose that all data exchanged between client and server is protected using a server-authenticated TLS channel established at the beginning of a client session. To make the authentication of server to client robust, the client is assumed to use certificate pinning for the AutoPass server.

Client Caching

The set of website data downloaded by the AutoPass server to the AutoPass client at the beginning of every session (see step 5 of Section 6.4.2) is not likely to change very rapidly. It therefore makes sense for the client to cache the most recently downloaded copy of this data, potentially improving system availability even if the AutoPass server is unavailable for a period.

Of course, if this data is cached long term, then it is at risk of compromise if the host device is compromised. To understand the nature of this risk we need to consider the data that would be cached. This information is enumerated in step 5 of Section 6.4.2, namely, for each website for which the user has chosen to use AutoPass to help manage the password:

- the first part of the URL of the site (this is used as the site identifier);
- the password policy for the site (in PRML);
- the set of input types used to generate the password for this site (e.g. whether a digital object is used);
- the password offset for this site, if it exists;
- any other parameters used to control password generation for this site.

Apart from the privacy issue of revealing with which sites the user interacts, the first and second pieces of information are completely non-sensitive, public information. The third and fifth items are also minimally sensitive. That is, the only piece of information of significance from a security perspective is the password offset (if it exists). As discussed in Section 6.6.3, if a password and the corresponding password offset are compromised then any future password for that particular site can be determined if the corresponding offset is known. That is, compromise of a single password offset is

not damaging, even if the corresponding password is discovered by some means. Only if the password is changed and the new offset is also compromised does this cause a threat. This suggests that caching does not add significantly to the risks. Of course, if users are likely to be concerned then an option to disable caching could easily be implemented, e.g. as a user option.

6.6 Security Evaluation

We next consider the security properties of the scheme.

6.6.1 Trust Relationships

Clearly, the AutoPass server must be trusted to some extent by the user, since if it sends incorrect data then passwords cannot be generated correctly. It also learns which websites the user interacts with, and hence it must be trusted to respect user privacy. On the other hand, it does not have the means to learn user passwords, since it only has access to an encrypted copy of the master password, which is used to generate all passwords, and thus it can be regarded as being partially trusted.

6.6.2 Threat Model

The security and correct operation of AutoPass depend on a number of key assumptions, which we enumerate.

- The client device is assumed to be uncompromised, since passwords are generated in and used by this device. If, for example, the browser is compromised then clearly the generated passwords may be compromised.
- The AutoPass client is assumed to be correct and without exploitable vulnerabilities. As for the previous assumption, if a corrupted version of the client software is present on the client device, then user passwords may be compromised.
- The AutoPass server provides correct information (see also Section 6.6.1 above).

Given the above assumptions, AutoPass is designed to resist the following types of attack:

1. active attacks on the communications link between the AutoPass client and server, including masquerading as the server to the client or vice versa, e.g. as made possible by an untrustworthy wireless access point or by DNS poisoning;
2. attacks on password secrecy conducted by the AutoPass server;

3. attacks on password secrecy conducted by a valid site against user passwords for other sites;
4. attacks on password secrecy conducted by any party with temporary access to the AutoPass server database.

6.6.3 Security Properties

We conclude this discussion by considering whether the desired properties are realised by the AutoPass design. We consider the four attacks in the previous section in turn.

1. Attacks of the first type are prevented by the assumption that all communications between the client and server are TLS-protected. The server will be authenticated by a pinned certificate. The client end of the link will not be explicitly authenticated to the server, but the presence of the correct user is verified by checking that the correct hash of the login password is sent over the link.
2. The server only has access to password metadata, a hash of the master password, and an encrypted copy of the master password. If the master password is automatically generated by a process using sufficient randomness, use of a 128-bit value will prevent direct brute forcing guessing attacks. However, the encryption of the master password is based on a key derived from the user-selected login password. If the login password is poorly chosen, then it can be brute-forced, meaning that the server could gain access to the master password. Thus it is vital for the user to choose a login password with high entropy. This is a reasonable assumption since this is the only secret the user is required to memorise. It is also worth noting that the use of password offsets means that if a password (and its offset) are compromised then any future password for that particular site can be determined if the corresponding offset is known. That is, whilst offsets will not be divulged to any party, a dishonest AutoPass server that (by some means) learns a user's password for a website will be able to determine all future passwords for that site.
3. The AutoPass system is completely transparent to authenticating websites, and passwords for distinct sites are computed separately as a function of the master password. If a website guesses that AutoPass is in use, it could use the password to try to perform a brute force search for the master password. However, if the master password is chosen at random then such a search is infeasible.
4. If an unauthorised party has access to the AutoPass server database, then it will not have immediate access to any user passwords. However, as discussed under 2

above, if the unauthorised party obtains the encrypted master password and the login password used to encrypt the master password is poorly chosen, then the attacker might be able to brute-force the login password and learn the master password. This argues in favour of the AutoPass server providing additional encryption of the database, giving protection against compromise of stored user data.

6.6.4 Security from An Attacker Perspective

To complement the above security discussion, we give an analysis of the security of the system from an attacker perspective. We divide this discussion into a number of cases depending on the nature of the attacker and the resources they might have.

- **Passive eavesdropper.** Such an attacker could eavesdrop on communications between the client and the server. However, we require all such communications to be TLS-protected (see Section 6.5). That is, no useful information, except for the fact that the client machine is talking to the AutoPass server, will be available to the eavesdropper.
- **Server compromise.** As noted in Section 6.6.3, the server only has access to password metadata for each website for which passwords are generated, the user account name and email address, a (salted) hash of the login password, a hash of the master password concatenated with a fixed string, and a copy of the master password encrypted using a key derived from the login password. We consider the possible threat posed by compromise of each of these values in turn.
 - Password metadata for each website. As discussed in Section 6.5 below, the only data of security significance is the password offset. The use of password offsets means that if a password (and its offset) are compromised then any future password for that particular site can be determined if the corresponding offset is known. That is, whilst offsets will not be divulged to any party, a dishonest AutoPass server that (by some means) learns a user's password for a website will be able to determine all future passwords for that site. However, this means that two distinct server compromises would be required for this to have an impact on password security.
 - User account name and email address. These are not of major security significance.
 - (Salted) hash of the login password. If the login password is poorly chosen, then a brute-force search for the login password could be successfully per-

formed using the salted hash, meaning that the server could gain access to the master password. Nonetheless, the use of a salt should make this more difficult by ruling out the use of precomputed tables of hashes of ‘likely’ passwords. Thus it is vital for the user to choose a login password with high entropy; this is a reasonable assumption since this is the only secret the user is required to memorise.

- Hash of the master password concatenated with a fixed string. If the master password is automatically generated by a process using sufficient randomness, use of a 128-bit value will prevent direct brute forcing guessing attacks based on this hash value. Concatenation with a fixed string prior to hashing prevents use of this hash value to help compute passwords.
 - Master password encrypted using a key derived from the login password. The encryption of the master password is based on a key derived from the user-selected login password. As above, if the login password is poorly chosen, then it could be brute-forced using the salted hash of this password, meaning that the server could gain access to the master password.
- **User device compromise.** There are a number of ways a user device might be compromised.
 - In the worst case, the hardware or operating system could be compromised to allow monitoring of user behaviour (e.g. keystrokes) by a third party. This might enable complete compromise of the system. However, it would also be potentially disastrous even if AutoPass was not in use, and addressing such a threat is way beyond the scope of a password generator or password manager.
 - Data stored on the device during a live session might be compromised. In this case, the multiply-hashed master password might be revealed. This would then potentially enable user passwords to be generated. As a result, it is important that, while stored during a session, the multiply-hashed master password is protected by the system as much as possible. Of course, more generally, compromise of a session while the user is logged on could be highly damaging in other aspects, and the user would be expected to take normal precautions for personal device protection such as using password-protected screen lock.
 - The least bad case is where data stored on a device long-term is compromised. As specified in step 8 of Section 6.4.2, at the end of an AutoPass ses-

sion all short-term stored configuration stored data, including the multiply-hashed master password, is securely deleted. That is, the only AutoPass-specific data that might be compromised in this case is cached data downloaded from the AutoPass server. As discussed in Section 6.5, the security impact of the compromise of cached data is limited.

- **Server impersonation.** Impersonation of the AutoPass server to a client device is prevented by the use of a TLS session with certificate pinning.
- **User impersonation.** Impersonation of a user to the AutoPass server is prevented by use of the login password for user authentication. Compromise of this password is prevented by the establishment of a TLS session before transmitting the salted hash of the login password to the server.

6.7 Addressing Shortcomings

In Section 3.5 we identified three major problems with existing password generators. These issues are all addressed by the AutoPass scheme, with the aid of a partially trusted server that does not have the means to recover individual user passwords. We examine the three issues in turn.

- *Setting and updating passwords.* As we have observed, all existing password generation schemes cause major difficulties for users with a large body of existing passwords, since they are obliged to change them all; AutoPass avoids this through the use of password offsets, allowing continued use of existing passwords.
- *Using multiple platforms.* The use of the AutoPass server allows seamless cross-platform working. At the same time, this server is only partly trusted, and does not have the means to recover individual user passwords, as discussed in Section 6.6.3.
- *Password policy issues.* The use of server-provided PRML statements allows passwords to be automatically generated to meet website-specific requirements.

Of course, whilst AutoPass works in theory, it is important to verify that the system also works in practice. A prototype implementation has been developed and used to conduct user trials. The prototype and the results of the trials are discussed in the next chapter.

6.8 Conclusions

In this chapter we described in detail the design of the AutoPass password generator. First we provided a high level description of AutoPass. We then provided a detailed specification of AutoPass, covering the types of inputs used to generate a password, the method of password generation, and the types and storage locations of configuration data. This was followed by a security evaluation, and a discussion of how the scheme addresses shortcomings identified in previous schemes.

Chapter 7

Testing AutoPass

7.1 Introduction

In Chapter 6 we described in detail the design of the AutoPass password generator. Its use of a server, in particular to store password offsets, PRML specifications and user preferences, is designed to remove the shortcomings present in all previously proposed password generators. Of course, whilst the system works in theory, it is important to also test that the system works in practice. To verify that the system is viable we first developed a prototype implementation and then used this prototype to conduct a small-scale informal user trial, with the goal of verifying that the prototype software is usable. In this chapter we first describe the prototype implementation, and then summarise the results from the user trial.

The remainder of the chapter is structured as follows. In Section 7.2 we introduce the design and implementation of the AutoPass prototype; this involves providing detailed descriptions of the AutoPass server, AutoPass browser extension and how the browser extension is used. This is followed in Section 7.3 by a description of the user trial, including: the methodology used, a description of the user demographics, and a summary and analysis of the data obtained from the trial. Finally, the chapter concludes in Section 7.4.

7.2 Prototype Design and Implementation

In this section we describe the AutoPass prototype; this involves providing detailed descriptions of the AutoPass server, the AutoPass browser extension and how the browser extension is used.

7.2.1 Overall Structure

AutoPass has two main parts: the AutoPass server and the AutoPass client. The AutoPass server stores less sensitive user data, and the AutoPass client software provides a user interface and automatically generates site-specific user passwords by combining the specified set of inputs.

7.2.2 AutoPass Server

The AutoPass server was developed using Python on the Django¹ platform; Django was chosen since it is easy to use, scalable, and appears to be reasonably secure. The AutoPass server is hosted on the Amazon Web Services (AWS) Elastic Beanstalk² cloud service. This cloud service was chosen for its ease of use and low cost. The service automates the process of setting up applications on the AWS infrastructure; as a result, setting up and executing the application was made very simple and took only a few minutes. It is also free for the first year of use. The AutoPass code is available on GitHub at <https://github.com/wanpengli/AutoPass>.

7.2.3 AutoPass Client

Because of the ubiquity of the Chrome browser, and the ease with which Chrome extensions can be developed and developed, the prototype client was developed as a Chrome extension written in JavaScript. It is available at <https://google-url.com/068vq>. Currently users need to know this link to install it, but the intention is to make it available via Google Play once it has been finalised. Key extracts from the client code are given in Appendix A.1.

The current version, which was used in the trial reported in the next section, offers the following functionality:

- generation and regeneration of a password on demand;
- use of password offsets to enable users to choose their own password, and to change passwords;
- automatic creation of passwords meeting website password policies;
- autofilling of passwords (this only works if a user has just one account at the website).

¹<https://www.djangoproject.com/>

²<https://aws.amazon.com/elasticbeanstalk/>

Two other features of AutoPass as specified in the previous chapter have not yet been implemented, namely:

- optional use of user-selected digital objects (e.g. files or images) as an input to password generation;
- use of a default password policy for websites for which a PRML policy file is not available at PRDDS³.

Implementation of these features is planned as future work.

7.2.4 Installing and Using the AutoPass Client

The following steps must be followed in order to install and use the AutoPass client. Note that a description of how to use the AutoPass prototype is also included in the user manual, a copy of which is given in Appendix A.5.

1. Download and Install from Google Play

The Chrome extension can be installed from the following link: <https://chrome.google.com/webstore/detail/autopass/oapnffdchomlagblkehioepojjkjkcjb/related>. The AutoPass icon is shown in Figure 7.1. After installation, Chrome will show the new extension next to the search bar.

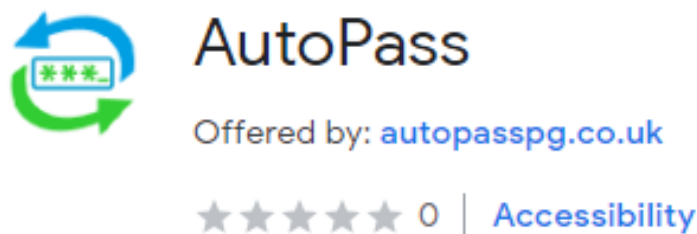


Figure 7.1: AutoPass extension from Google Play

2. AutoPass Registration

After installation of the extension has been successfully completed, the user must next register with the AutoPass server to create an account at the server. This involves the following steps.

³<https://api.ppdds.passwordassistance.info/v1/ppds//>

- (a) The user must first cause Chrome to visit <http://www.autopasspg.co.uk/login/>.
- (b) The user should then click on the *Sign Up* button, which will lead to the registration page. The user must then complete following information.
 - i. *Account name*: the user must choose an account name after checking its availability using the *check account availability* option.
 - ii. *First and last names*.
 - iii. *Email address*: this is used for password recovery.
 - iv. *Password*: this is used to log in to the application (i.e. it is the login password in the terminology used in the previous chapter) and must contain at least 8 characters.
 - v. *Password confirmation*.
 - vi. *Master password*: if the user clicks on *Generate Master Key*, the client will generate a random master password. This will be displayed to the user and the user is recommended to keep a copy (as it cannot be recovered if the user forgets their login password).
- (c) The user is then requested to solve a CAPTCHA⁴ to try to reduce the risk of automated account creation by bots.
- (d) Registration is completed by pressing the *Registration* button. Once pressed the extension will allow the user to create a printed copy of the registration information.

3. Logging In

Once the user has registered, the user will be required to log in every time Chrome is started up. This involves the following steps.

- (a) To commence the process the user should click on the AutoPass extension logo in the Chrome browser. This will cause the AutoPass login window to appear (see Figure 7.2).

⁴A CAPTCHA (see, for example, [87, 6] and <http://www.captcha.net/>) is a type of challenge-response test used to determine whether or not a remote user is human and not a malicious program, such as a bot.

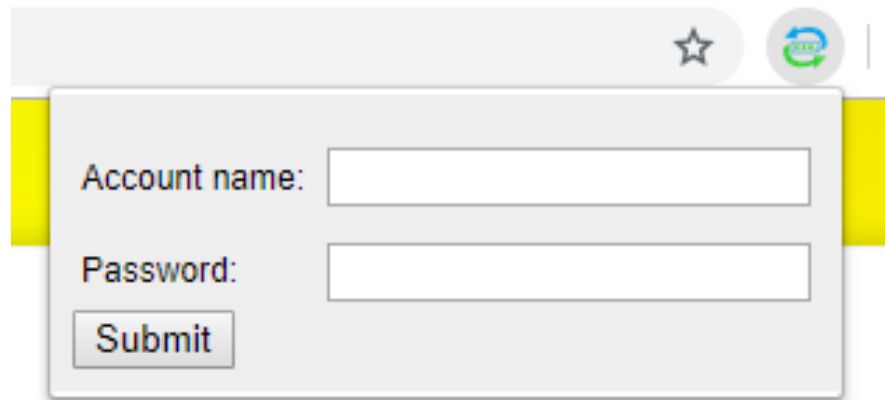
A screenshot of a web browser's address bar area. On the right side of the address bar, there is a star icon for bookmarks and a circular refresh icon. Below the address bar, a light gray dialog box is displayed. The dialog box contains two text input fields. The first field is labeled "Account name:" and the second is labeled "Password:". Below these fields is a button labeled "Submit".

Figure 7.2: AutoPass login window

- (b) The user should enter his or her account name and password. The user will then have access to the AutoPass functionality.

4. First Time Use with A Website

We next consider the case where a user navigates to a website login page (or website registration page) which has not previously been set up with AutoPass. Of course, AutoPass may fail to detect the login page since it relies on a set of heuristics to detect such pages — if so then the user will be forced to proceed without AutoPass support. We therefore suppose that AutoPass has detected a login page, in which case AutoPass will display a popup of the type shown in Figure 7.3.

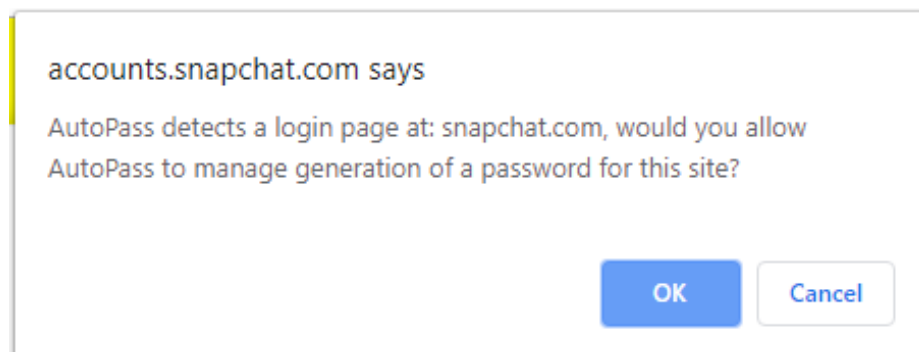
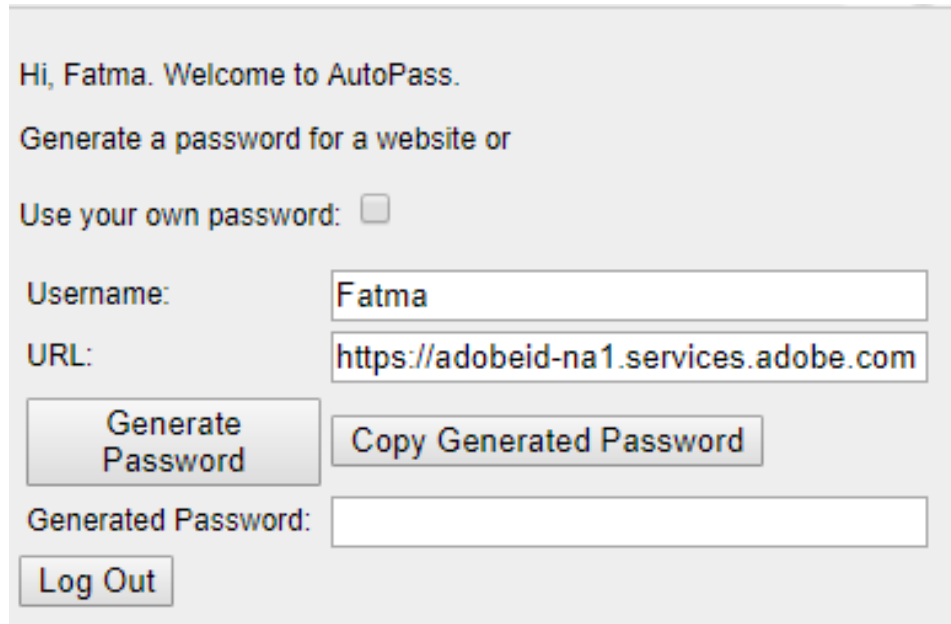
A screenshot of a system dialog box. The title bar of the dialog box is not visible. The main text inside the dialog box reads: "accounts.snapchat.com says" followed by "AutoPass detects a login page at: snapchat.com, would you allow AutoPass to manage generation of a password for this site?". At the bottom right of the dialog box, there are two buttons: a blue "OK" button and a white "Cancel" button with a gray border.

Figure 7.3: AutoPass pop-up for user consent

- (a) If the user clicks *Cancel* then AutoPass will do nothing and will return to

looking for login pages. If the user clicks OK, then the window shown in Figure 7.4 will be displayed. The user will be required to enter his or her *User name* for this website (as opposed to the user name for AutoPass). The site URL will be automatically completed by the AutoPass client. A *Log out* button is provided enabling the user to log out from AutoPass.



Hi, Fatma. Welcome to AutoPass.

Generate a password for a website or

Use your own password:

Username:

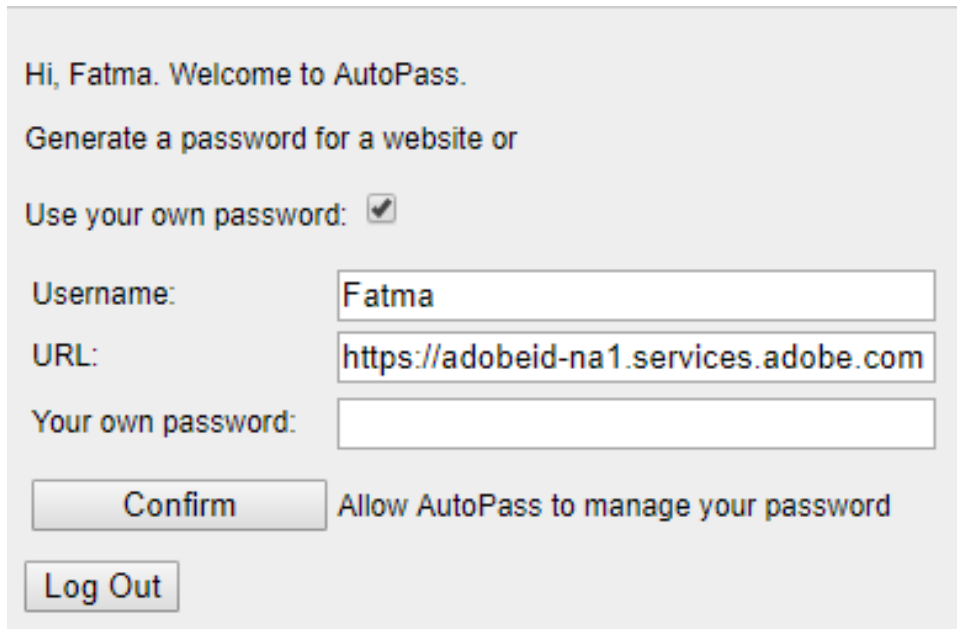
URL:

Generated Password:

Figure 7.4: AutoPass password generation window

- (b) After pressing *Confirm*, the AutoPass client will generate a password for the site which (on this occasion only) the user must manually copy into the appropriate place in the website registration page.

If the user wishes to choose the password for this site then he/she must also click the *Use your own password* box and enter the chosen password into the *Your own password* field as shown in figure 7.5. The site URL will be automatically completed by the AutoPass client. A *Log out* button is provided enabling the user to log out from AutoPass. After pressing *Confirm*, the AutoPass client will regenerate the entered a password for the site and automatically enter it into the field in the login page.



Hi, Fatma. Welcome to AutoPass.

Generate a password for a website or

Use your own password:

Username:

URL:

Your own password:

Allow AutoPass to manage your password

Figure 7.5: Use your own password window

5. Subsequent Use With A Website

If AutoPass detects a login page for a website for which it has already been set up, it will simply fill in the user name and the generated password.

7.3 A User Trial

The prototype described in the previous section was used as the basis of a small-scale informal user trial to provide a preliminary evaluation of the usability and security of AutoPass. The methodology for and the results from this trial are now discussed.

7.3.1 Methodology

Before conducting the trial we completed the Royal Holloway ethical review process. This was necessary since we were conducting a trial with human participants. Details of the ethical review process are available at: <https://intranet.royalholloway.ac.uk/staff/research/research-and-enterprise/research-enterprise/ethics/ethical-approval-process.aspx>.

We used a quantitative approach, and obtained statistics about user behaviour and user attitudes relating to passwords and AutoPass. We were also open to discussions with the participants for feedback on AutoPass. Since only a relatively short time was

available to set up and conduct the trial, and in the absence of funding to support a large study, we decided to recruit volunteers for the trial from fellow students, friends and acquaintances. As a result of our requests we obtained a total of 17 participants for the trial (all of whom were Chrome users).

Each participant was asked to (a) read the provided documentation and install AutoPass in Chrome, (b) use the AutoPass extension in their everyday web activities for at least a week, and then (c) complete a short questionnaire about their experiences using the extension. The goal was to obtain a general overview of AutoPass usability and security from the user perspective. Once signed up, each participant was provided with a link to enable them to install the Chrome extension. The extension was uploaded to Google Play and made available via the link: <https://chrome.google.com/webstore/detail/autopass/oapnffdchomlagblkehioepojkjjcjb/related>.

Each participant was also provided with the following four documents.

- *A Consent form.* The form is included in Appendix A.3. Given that the trial involves working with user passwords, it clearly has privacy implications; it therefore seemed appropriate to both inform the users of any issues and also obtain their informed consent. The form provides very basic information about the nature of the trial and also gives assurances about the handling of any personal data. Participants were asked to return a signed copy of the form before participating in the trial.
- *A User guide.* This is included in Appendix A.4. This provides general information on passwords generators and in particular on AutoPass. It focusses on the nature of the data used and stored by the AutoPass software.
- *The AutoPass manual.* This is included in Appendix A.5. This general manual provides detailed instructions on the installation and use of AutoPass. It is hoped that this will of value to any future users of AutoPass, and not just for participants in the trial.
- *A Questionnaire.* This is included in Appendix A.6. This was set up in Google Forms⁵, and participants were asked to provide their responses via Google after use of the extension for at least a week. Use of Google Forms made it simple to recover statistical information from the responses, and also to generate graphical representations of the response data.

The questionnaire was primarily designed to learn whether the users found AutoPass easy to use, and whether it met their needs for automated password management.

⁵<https://www.google.com/forms/about/>

Given that the trial was only intended as an informal evaluation, we did not design the questionnaire using any formal methodology. The results should therefore be seen as an informal indication of initial reactions to use of the scheme, rather than as a detailed examination of its strengths and weaknesses. A more formal and detailed study is planned future work.

7.3.2 User Base

As noted above, the trial involved a total of 17 participants, a relatively small number compared to previous studies [14, 67, 90]. All the participants were either fellow students at Royal Holloway, University of London or colleagues at the College of Applied Science in Oman. Unfortunately, only 12 of the 17 participants completed the trial by returning the questionnaire, and hence from here on all information is with respect to the 12 who completed the trial. Three of the five who didn't complete the trial were Linux users who were unable to install the extension for compatibility reasons.

Nine of the 12 completing participants were female, and they were all aged between 25 and 64. All participants had a background in Computer Science and/or Mathematics. The age distribution of the 12 who completed the trial is shown in Figure 7.6.

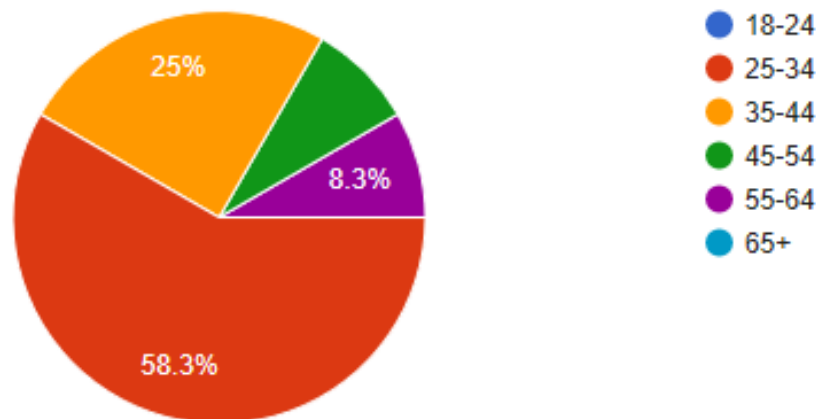


Figure 7.6: Participant age distribution

7.3.3 Summary of Results

The full results from the 12 completed questionnaires are given in Appendix A.7. After two questions about the participants themselves, the next three questions (q3-q5)

in the questionnaire aimed at exploring user attitudes towards passwords, password generation and enabling users to identify and to be aware of these issue before using AutoPass. The questions were derived from the Lyastani et al. analysis [63], shown in Figure 7.7.

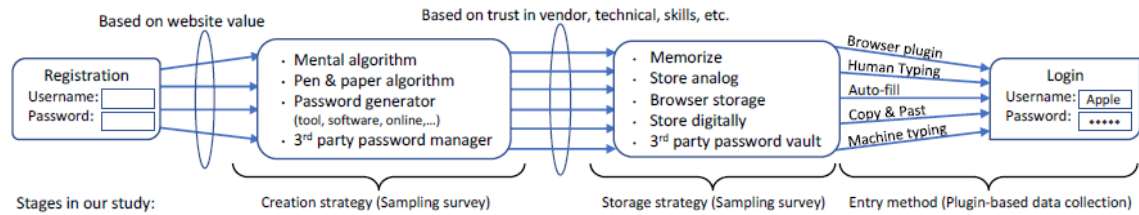


Figure 7.7: password generation strategies

The main conclusions from an analysis of the answers to these questions are as follows.

- As shown in Table 7.1, more than half the participants have problems with both managing large numbers of passwords for multiple accounts and with varying password policies. This result was not unexpected, and is consistent with the literature (see, for example, Lyastani et al. [63]).

Table 7.1: Issues user face with using password

Difficulties user face with passwords	Total
Too many accounts with too many passwords	58.30%
Different websites have different policies on what is permitted as a password e.g. upper case, lower case	58.30%
Some websites require frequent password changes	50%

- Table 7.2 shows that the most common password generation technique is to employ familiar phrases or data, closely followed by use of ‘random’ passwords; it is not clear whether either of these approaches result in genuinely ‘strong’ passwords. Also, a large proportion of the participants admitted to re-using passwords, a clear security risk.

Table 7.2: User password generation techniques

User password generation techniques	Total
Password generator	8.3%
Password generator which is built into a password manager or browser	0%
I use the same password for all accounts with minimal modification	41.7%
I use familiar phrases or dates	50%
I generate a random password from whatever comes to mind	41.7%

- Half of the participants had previously used computer software to generate passwords.

The second main part of the questionnaire (q6-q11) focussed on the usability of AutoPass. 75% of the participants found it easy or very easy to use, which was encouraging. Also, more than half the participants preferred AutoPass to their existing password management method. All but two of the participants valued the ability to continue to use existing passwords.

7.3.4 Analysis

The results described in Section 7.3.3 confirm that users are struggling with the potentially large numbers of passwords they are currently required to manage. The results suggest that many users cope with this problem by risking security through use of weak passwords and/or by re-using passwords. The users are fully aware of such bad practice and the risk associated with it and still accepting the risk. This is quite alarming since most of our participants are from computer science background who are always in contact with the updates in the technology and has better understanding of the technology. Also the age of the participant identify that none of the participants belong to elderly category who could be unfamiliar with the technology and has difficulty to adapt. These issues could be amplified with people have limited knowledge in computer science or if there were in age where adapting with technology become a challenge.

AutoPass was found to be easy to use by the majority of the participants; users also found it easy to create an account. The fact that AutoPass was available in the Google store made it simple to download and install. Users were satisfied with the passwords generated by AutoPass, as shown in Figure 7.8. Users also found it valuable to be able to continue to use existing passwords, as shown in Figure 7.9. These results were encouraging, and suggest that AutoPass offers a useful service and incorporates functionality valued by users.

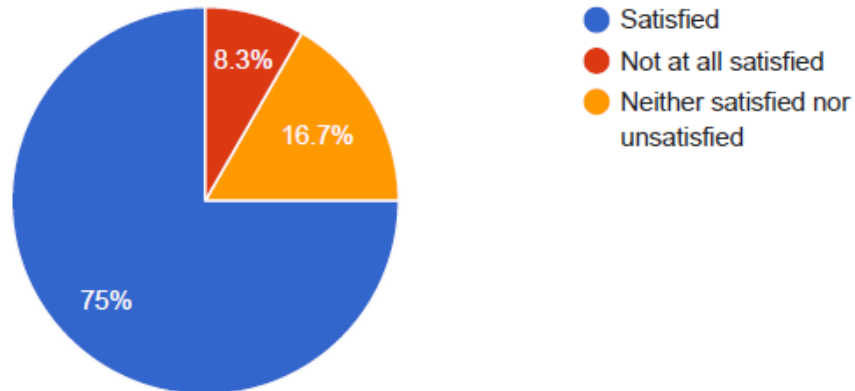


Figure 7.8: Level of satisfaction with passwords generated using AutoPass

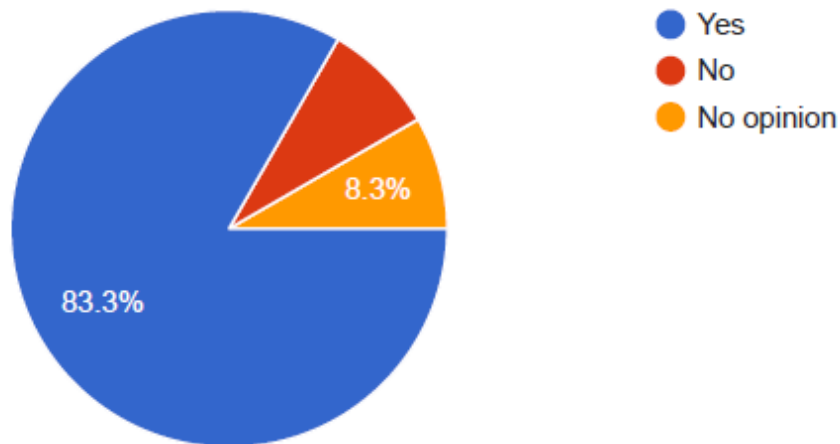


Figure 7.9: Did you find the ability to use existing passwords useful?

When asked to compare AutoPass with other password management schemes, only 17% of the participants said they found it better. However, whilst apparently disappointing, 50% of the participants had not used password management software before, and 33% had no preference. Encouragingly, 58% prefer AutoPass to their current method of password management.

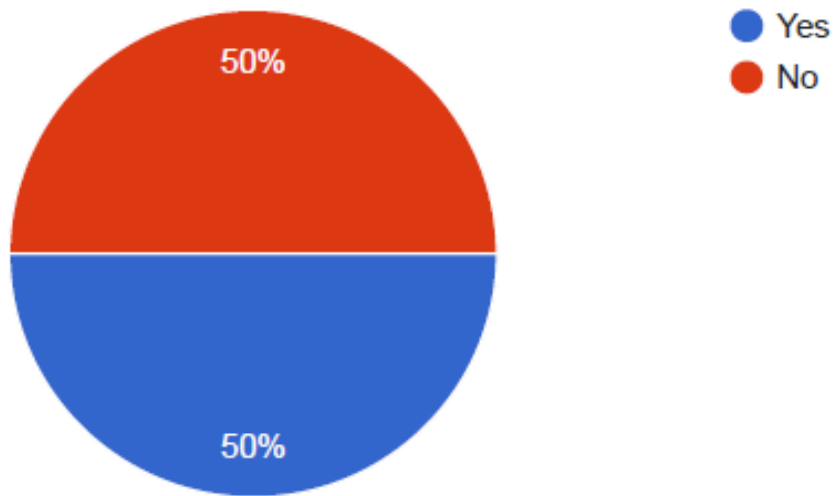


Figure 7.10: Have you ever used a computer program to generate your passwords?

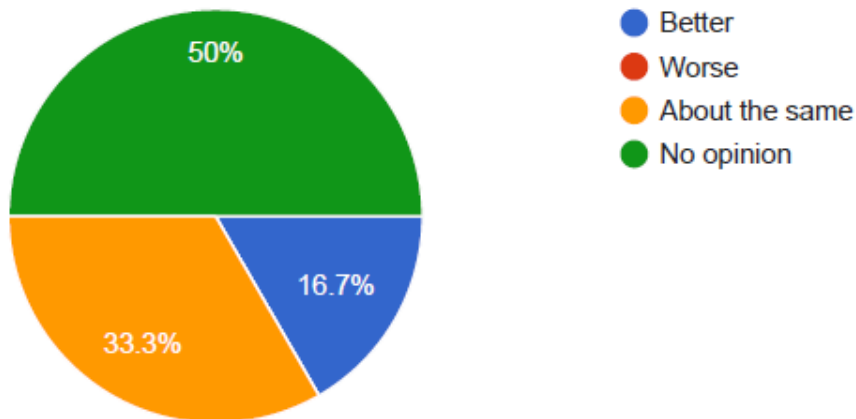


Figure 7.11: If you ever used a computer program to generate your passwords, how does AutoPass compare with methods you have used previously?

The last question was an open question, intended to allow the participants to indicate additional features that might improve the functionality of AutoPass. The responses we received are listed below along with an assessment of the feasibility of implementing the suggested functionality.

- “*Change the random password*”. This feedback is unhelpful since the nature of the suggested change is unclear.

- *“to auto fill the user name as well, not only the URL”*. AutoPass autocompletes the user name and the password automatically from the second visit onwards.
- *“linking password creation with human memorization theories So the program doesn’t generate password only but helps user to remember it”*. This is a valid thought; however the idea of AutoPass is to take the load of memorising away from the user.
- *“There should be specifications regarding creating passwords in AutoPass in order for the password to be solid”*. Unfortunately the author of the comment does not appear to understand that AutoPass generates strong passwords by default.
- *“length of the password should be limited”*. The length of the password is governed by the website requirements. So if a website requires no more than 12 characters then AutoPass will generate a password of that length.
- *“A timer to change your password, to improve security of my password maybe?”*. It would be fairly straightforward for AutoPass to be enhanced to incorporate a password change reminder, e.g. where a user is reminded to change their password after, say, a year has elapsed.

It seems that the most important remaining issue is to convince users to use password generators such as AutoPass, and thereby avoid the need to adopt insecure practices such as password re-use and selection of weak passwords.

7.4 Conclusions

We believe that the small-scale trial described in this chapter suggests that AutoPass is indeed a usable solution for user password management. One important next step will be to enhance AutoPass to add functionality currently missing, including optional use of digital objects and the provision of a default password policy. Another important direction for future work will be to create versions for all the most popular browsers, operating systems and platform types, thereby addressing the compatibility issues identified by three of the original 17 trial participants. Finally, further and more extensive user trials will be required to both test the added features, and obtain more detailed and statistically significant information about user attitudes to password generators such as AutoPass.

Part III

Password Recovery

Chapter 8

Modelling the Password Recovery Process

8.1 Introduction

Passwords are frequently forgotten, and as a result websites typically provide a mechanism to enable a user to re-establish a shared secret. Florêncio and Herley's [28] study reveals that every month over 1.5% of Yahoo users of a typical website forget their password and reset it. Password recovery can be achieved in many ways, including use of a pre-registered email address or mobile phone number, and/or involving other pre-set fall-back means of authentication. However, many of these techniques introduce vulnerabilities which may enable an impostor to falsely change a password, either causing a denial of service or, in the worst case, enabling the impostor to authenticate as the user.

In this chapter, we analyse these issues by first introducing a general model for password recovery, and then examining various existing password recovery options within the context of this model. We also use the model to look at ways of improving the security and/or usability of password recovery. In the next chapter we examine one aspect of password recovery in greater detail, namely the real-world use of password recovery emails.

This chapter is structured as follows. In Section 8.2 we provide a general model for web password recovery, intended to capture the commonly used approaches. Section 8.3 then reviews the various ways in which password recovery is performed in practice, within the context of the model. This then enables us in Section 8.4 to provide a systematic assessment of the security strengths and weaknesses of existing approaches, along with potential usability issues. In Section 8.5 we use the results of this assessment

to provide a series of recommendations on the design of password recovery systems. Section 8.6 concludes the chapter.

8.2 A General Model

We now present a general model for a password recovery system. It provides a framework within which we can examine the security of various options for the recovery process. As discussed above, *password recovery* is a process which enables users to re-establish a password for a website account. Sometimes referred to as *password reset*¹, password recovery is typically performed when a user forgets the password for a website. There are a variety of ways in which password recovery can be performed, some more secure than others, and the general model is intended to capture all the means of password recovery in current use.

8.2.1 Constituent Processes

A password recovery process typically involves three sub-processes, namely:

- **registration**, in which user-specific information to be used during password recovery is captured by the website,
- **password setup**, in which the user chooses and sends the website a new password for that site, and
- **recovery**, where the user interacts with the website with the goal of re-establishing a shared secret password.

Typically, the first of these will be performed just once, the second will be performed infrequently, and the third will be performed whenever necessary. These three sub-processes are next examined in greater detail.

8.2.2 Registration

In establishing an account with the website, a user typically sets up a user name and password, as well as other information for use in the recovery process. This information is used to ensure the security of the recovery process, i.e. to help prevent an unauthorised party from changing the password to cause a denial of service, or, even worse, obtaining a valid password for the account. Existing categories of information of this type are examined in Section 8.3.1.

¹We avoid this terminology since it implies changing the existing password, and not all password recovery schemes involve such a change.

8.2.3 Password Setup

We assume the website uses a password to authenticate the user, and so both initially (probably as part of registration) and whenever the user wishes to change it, the user will need to supply the website with a password. Other information, e.g. a ‘password hint’, may be collected at the same time. Further examples of information that might be gathered in this stage are given in Section 8.3.2.

8.2.4 Recovery

Password recovery is typically invoked by a user when he/she forgets the password for a website. Such a process typically has three stages, as follows.

1. **Recovery request.** This involves the user signalling to the website to request password recovery. The website will typically provide a means for this to occur, e.g. a special button.
2. **Request validation.** The website checks that the request is valid. A website may perform user authentication, obviously using a means other than the password, although by no means all websites do this. In this stage the website may also attempt to verify that the request originates from a human rather than a bot, e.g. using a CAPTCHA.
3. **Password re-establishment.** At the conclusion of this stage, if successful, the user is equipped with a valid password for the website. There are two main implementation approaches.
 - The password recovery system can help the user remember (recover) his/her password; this will typically involve the system keeping a copy of every user password, and password recovery will involve reminding the user what it is (typically after he/she has been authenticated).
 - The password recovery system can help the user set up a new password, often referred to as *password reset*. This may involve the system giving the user a temporary password, which the user must change at first use.

Regardless of the particular implementation approach, the recovery process will typically involve using a secure communications channel to the user, e.g. based on a previously registered email address or phone number. Examples of how each of these steps might be executed are provided in Sections 8.3.3 and 8.3.4.

8.3 Model Components

We next examine how the model components can be instantiated. The various options introduced here are critically analysed later in the chapter.

8.3.1 Registration

Registration for password recovery is typically implemented as part of a general registration process, in which the user establishes a new account (known as *registering* or *signing up*). As well as gathering information, it may also involve security-related steps, e.g. solving a CAPTCHA to prevent automated user account harvesting. It often involves collecting a wide range of information, including matters not relevant to password recovery (e.g. payment information); we focus only on information related to password recovery. Such information can be divided into two main categories:

- *personal information*, i.e. information about the individual that may be used for a range of purposes apart from password recovery;
- *recovery information*, i.e. information used only for password recovery.

A website will also typically require the user to choose a unique user name (or give an email address to function as a user name) as well as a password; these two pieces of information are obviously key to password recovery, although given their special status we do not include them in this classification. We next look at examples of widely used information types of both categories.

Personal information

The following are examples of the types of personal information that may be gathered during registration and subsequently used for password recovery. In each case, examples of websites are given which use such information in a password recovery process.

- name (first name, last name), e.g. as gathered by Instagram²;
- gender, e.g. as collected by Facebook³;
- birth date (day, month, year), e.g. as gathered by Google Mail⁴;
- street address (street name, city, country), e.g. as used by Microsoft email⁵;

²<https://goo.gl/sG4wDv>, accessed:09/04/2018

³<https://goo.gl/wCsLMk>, accessed:09/04/2018

⁴<https://goo.gl/b67119>, accessed:09/04/2018

⁵<https://goo.gl/9hbHNX>, accessed:09/04/2018

- email address, e.g. as collected by Amazon⁶;
- phone number (e.g. mobile number), e.g. as gathered by Twitter⁷.

Recovery information

The types of information established purely for password recovery purposes vary widely, depending on the detailed operation of the recovery process. We can identify the following general categories.

- **Recovery authentication information**, i.e. information that can be used to authenticate the user. Examples include the following.
 - **Answers to security questions**: the website may give a list of security questions (also known as *personal knowledge* or *challenge* questions), for a user-selectable subset of which the user must provide answers. These questions cover topics which the user can easily remember since they relate to the user's personal life, e.g. school name, mother's maiden name, pet name, memorable street address, birthplace, favorite colour, etc. Examples of websites that request answers to security questions during registration include Apple and Google. In addition, some websites, e.g. Alipay.com, allow users to customise the question, i.e. the user provides both the question and the answer.
 - **One-time recovery (backup) codes**: a website may set up one or more one-time backup codes, which the user must securely retain for possible future use for account recovery. These backup codes are typically not used for password recovery as we define it here, but for closely related purposes. In particular, Facebook and Google both support the establishment of such codes to enable users to recover account access if a second authentication factor fails or is unavailable, e.g. if a one-time password sent via SMS or email cannot be accessed by the user. Of course, such backup codes could also be used for password recovery but, since we are not aware of any websites using them in this way, we do not discuss them further in this chapter.
- **Recovery contact details** are special contact details, such as an email address or phone number, that are used for password recovery. Examples of such details include the following.

⁶<https://goo.gl/yrv1fA>, accessed:09/04/2018

⁷<https://goo.gl/uUYysF>, accessed:09/04/2018

- **Contact details for trusted associates (trustees):** i.e. email addresses or phone numbers for one or more individuals trusted by the account holder, e.g. as used by Facebook. During password recovery, a verification code is sent to a trustee, who is trusted to relay it to the correct user.
- **Recovery email address:** i.e. one or more email addresses to which a one-time password or a link to a special recovery page is sent by the website during the recovery process.
- **Recovery phone number:** i.e. a phone number used to send a one-time password for recovery purposes.
- **Recovery preferences** can be gathered at this stage if a website offers more than one option for password recovery. The user preference can be established during registration, or, as is the case for Google, the user is simply offered various options if the password recovery process is invoked.

8.3.2 Password Setup

As part of the process of entering a password, the website may gather and store the following types of information.

- **Password hints:** when a password is entered, some websites also request a hint, intended to help the user remember the password. Alternatively, some websites, e.g. Google (see Figure 8.1), record the time when the password was last changed, and subsequently use this information as a password hint when the user makes a password recovery request.
- **Old passwords:** some websites, e.g. Gmail, retain a means to verify old (superseded) passwords, to enable their use for authentication during password recovery.

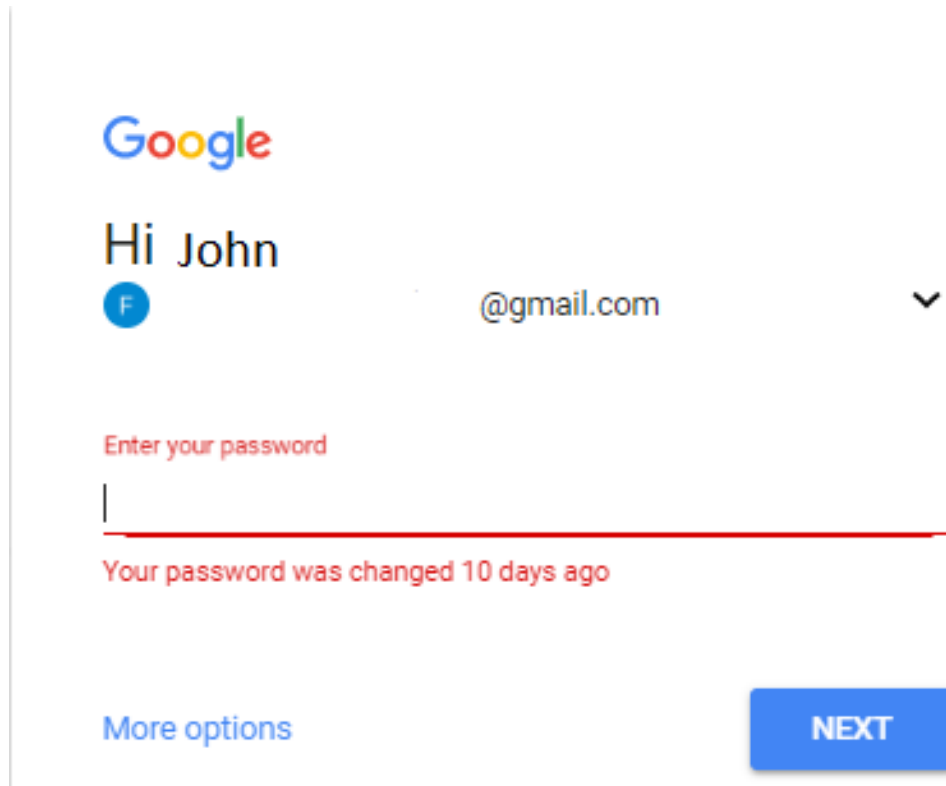


Figure 8.1: Google password change hint

8.3.3 Recovery Request and Validation

We consider the request and validation steps together since, in practice, these steps are often combined. A website will typically provide a simple means for a user to invoke the password recovery process, e.g. by providing a link somewhere close to the password field in the login page. Once invoked the user will typically be asked to perform one or more of the following steps in order to prevent the acceptance of fraudulent requests:

- solve a CAPTCHA — some websites, e.g. PayPal, use a CAPTCHA during password recovery to filter out automated attacks;
- answer one or more of the pre-established security questions, e.g. as performed by Apple;
- select an option for password recovery (if the user registered multiple methods), e.g. as is the case for Amazon;
- enter the last password the user can recall, e.g. as requested by Google.

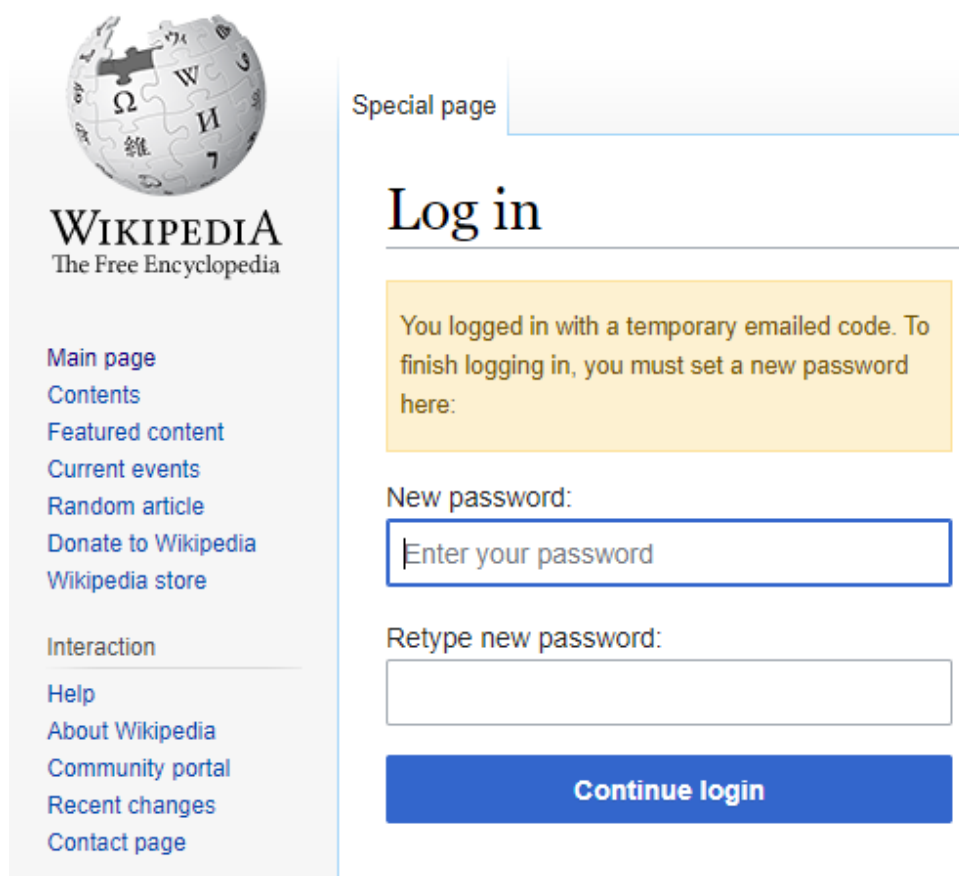
Following initial acceptance of the request, a user may be required to engage in further interactions to validate the request, e.g. as follows.

- An email is sent to the user’s registered address, and the user is asked to perform a task using information contained in the email, e.g. clicking on an embedded link or entering a code value.
- An SMS (text) message containing a secret code is sent to the user’s registered mobile number, and the user is requested to enter this code; alternatively the user may receive an automated call to his/her mobile phone and be asked to engage in a short dialogue.

8.3.4 Password Re-establishment

We next review some commonly used methods for re-establishing a password.

- **Email reset** involves a website sending a message containing secret information to the registered email address of the user requesting recovery. Subsequent use of this information implicitly authenticates the user, assuming that only the user can access emails sent to his/her email address. This approach is very widely used; Bonneau et al. [12] found that 92% of the 138 websites they tested use email-based password recovery. Various types of secret information are commonly sent, e.g. as follows.
 - **Verification codes/temporary passwords** are a temporary means of accessing the user account, purely for the purposes of establishing a new password, and are used, for example, by Amazon and Wikipedia — see Figure 8.2. Note that Wikipedia limits users to one temporary password per 24-hour period, and its temporary passwords expire after a week.
 - **Links** (URLs) are embedded in emails sent to a user-registered address. Clicking on such a link (which typically contains a secret string) redirects the user browser to a page enabling the user to set up a new password. Some websites, e.g. Twitter, limit the validity period of such links.
- **SMS reset** messages are sent to the user’s registered mobile number, and typically contain a secret verification code (analogously to email reset).
- **Use of an old password** is permitted by some websites, e.g. Google, as a means of authenticating a user for password recovery. Typically, it will form only part of the process of authenticating the user.



Special page

WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)
[Featured content](#)
[Current events](#)
[Random article](#)
[Donate to Wikipedia](#)
[Wikipedia store](#)

Interaction

[Help](#)
[About Wikipedia](#)
[Community portal](#)
[Recent changes](#)
[Contact page](#)

Log in

You logged in with a temporary emailed code. To finish logging in, you must set a new password here:

New password:

Retype new password:

Continue login

Figure 8.2: Wikipedia password recovery using a temporary password

- **Use of a trustee** is supported by some websites (e.g. Facebook), where the trustee is used as a secure communications channel to the account holder for sending secret recovery information, such as a temporary (one-time) password.

8.3.5 After Password Re-establishment

After a user has reset his/her password, some websites immediately log out the user and require a fresh log-in with the new password; others allow the user to continue without logging-in again. In parallel with this, some websites notify the account holder via email that password recovery has occurred, with the goal of alerting users if the recovery was not authorised by the account holder.

8.4 Security and Usability Issues

We now consider potential security and usability issues for the recovery validation and password re-establishment techniques given in Sections 8.3.3 and 8.3.4.

8.4.1 Security Questions

As observed in Section 8.3.3, security questions are widely used to help ensure that a recovery request is valid. As discussed in Section 4.2 in Chapter 4, apart from the fact that gathering such personal data potentially endangers user privacy, a range of serious security and usability issues have been identified. Bonneau et al. [10] identify the following problems.

- Answers to some questions are more readily guessed than others since they have a small answer space; such questions offer relatively little protection.
- Some users may provide false answers with the goals of limiting what is revealed about them and making it harder for impostors to guess their responses. However, users may forget their false answers, making the security questions useless in the recovery process.
- Although the questions are designed to cover topics which users will always know, some questions may nevertheless require the user to remember the answer they gave, e.g. which of many pet names they chose, or which colour they said was their favourite. Given such questions are likely to be used very infrequently, and considerable time may elapse between setting them up and using them, a user's recollection of the 'correct' answers is likely to fade.
- Cultural differences can have an impact on the memorability of some questions, e.g. Bonneau et al. [10] found that, for a typical set of questions, French users are most likely to recall their first phone number and are least likely to recall their father's middle name.

Also, the answers to some questions could be obtained via social media, as discussed by Rabkin [74]. Even more seriously, the same questions are used by many sites, making possible the type of MitM attack described by Gelernter et al. [33] (see Section 8.4.5), where a malicious site persuades a user to register with answers to security questions which are then used to impersonate the user to another site. The real issue here is that the authentication information (answers to security questions) is not site-specific; analogous problems arise when users employ the same password with many sites —

this has caused many well-documented security problems, notably when passwords for one site are compromised and can then be used to try to impersonate users to other sites⁸.

The main lessons from this analysis would appear to be that: (a) security questions should be carefully chosen to maximise the level of security offered; (b) security questions only offer a limited level of security and should always be used in combination with other methods; and (c) ideally authentication information should somehow be made site-specific.

8.4.2 Trustee-based Recovery

As discussed by Gong and Wang [34], there are a number of security issues with relying on trusted ‘friends’ of a user to support password recovery.

- The account holder may forget who they chose as trustees, so that an individual may remain a nominated trustee even if they are no longer trusted. That is, the state of trust can change with time and a user could forget to remove a trustee from the list.
- Use of trustees can be prone to ‘forest fire attacks’ [34], where compromise of a trustee account can compromise many other accounts.
- Users are sometimes obliged to nominate several trustees, e.g. Facebook mandates three. This may force the user to select less trusted associates.
- A malicious trustee could take over an account by triggering password recovery and (if necessary) communicating with other trustees (e.g. using social engineering) to obtain all the secret information necessary.

8.4.3 Email Reset

A major vulnerability of this approach is its assumption that emails cannot be intercepted. There are various ways this assumption could be invalidated.

- If emails are retrieved via an unencrypted link, e.g. if the user accesses email via a browser and the website does not use https, or if a mail client employs IMAP over a link not protected using SSL/TLS, then emails might be intercepted.

⁸This analysis suggests a very simple attack on passwords, where a malicious entity sets up a site and persuades users to register and choose an ID and password; the site can then act on the assumption that some users will employ the same user name/password combinations elsewhere, and can try them out with other sites to see if they work. Such an attack could be very effective without even requiring any real-time MitM activity or compromise of existing password databases.

This threat is particularly significant when using public access networks; a MitM attacker operating a ‘fake’ public wireless access network could intercept emails received via SMTP.

- If the reset email is forwarded to a third party, either accidentally or deliberately, then the contents could be compromised.
- Malware in the client device could be used to obtain the reset email. For example, Gelernter et al. [33] describe a malicious android application which, if installed, can covertly read received emails.

The use of email for distributing reset information also has other risks. For example, consider the case where the website for which password recovery is being performed is itself a provider of an email service (e.g. Google). In this case, sending a password reset message by email will not help, as the user will not be able to log in to retrieve it. In such a case it is common practice for the user to be asked to register an alternative email address to be used to distribute recovery information. During password recovery, in some cases the user will be shown a partially obfuscated version of this alternative email address, and the user will be asked to confirm that this is the correct address for use in password recovery. Figures 8.3 and 8.4 show this process as used by Gmail. This is a clear privacy breach, in that it may well be possible for a third party to learn the entire alternative address, e.g. by a web search or by automated harvesting of email addresses [38] (see, for example, Polakis et al. [73]).

Apart from the risk of spam, restricting access to email addresses is clearly desirable given that email addresses are often used as user identifiers. A further possible problem arising with the use of email relates to the non-permanent nature of email addresses. A user may register an email address which may later become re-assigned to someone else, e.g. because they change service providers or employers. In such a case, a password reset message may be received by a third party, who could use it to take control of the user account. Another issue could arise in a work environment in which employee emails are temporarily forwarded to another person, e.g. because the employee is sick or on leave; this could be especially hazardous if the forwarder is a temporary employee or a contractor.

8.4.4 Verification Codes and Reset Links

As discussed in Section 8.3.4, there are two main ways emails can be used for password recovery, namely by sending a verification code or a reset link. As discussed above, both give rise to a risk if the email is compromised. Apart from the direct compromise threat, both have other security and usability issues.

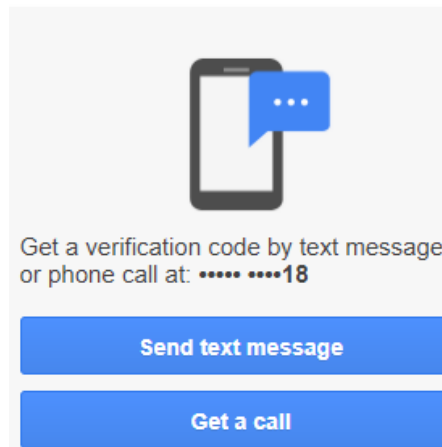


Figure 8.3: Partial leakage of user contact number

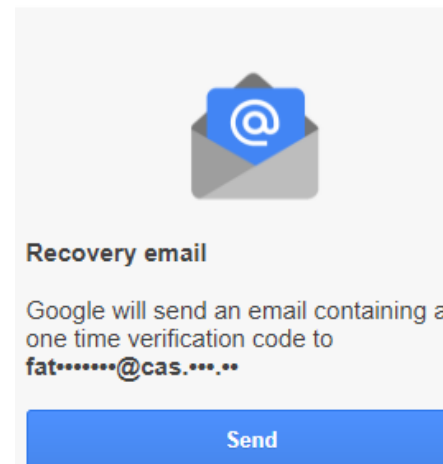


Figure 8.4: Partial leakage of alternative email address

- The following issues are associated with verification codes.
 - Lack of entropy: since verification codes are typically short numeric values, they are potentially vulnerable to brute force searching attacks. That is, if a website does not limit the number of attempts to enter the code, then a simple brute force attack becomes possible, as demonstrated by a successful attack on two Facebook sites [86].
 - As discussed in greater detail in Section 8.4.5 in the context of codes sent via SMS, a user could be misled into revealing a verification code to a malicious site in a type of phishing attack.
- Reset links have a slightly different set of associated issues.
 - Some websites do not expire unused reset links, i.e. they remain valid indefinitely. For example, in an experiment we discovered that a link for rosegal.com remained valid for at least five months. Since it is hard to understand why a link needs to be valid for more than a few hours, this represents an unnecessary risk, since if such a link is ever disclosed it may enable the associated account to be hijacked.
 - Karlof et al. [53] describe a phishing attack on link-based password recovery. The attacker starts password recovery at a website for a target user, knowing this will cause the website to send an email containing a link to the that user. Simultaneously the attacker emails the user asking them to paste the link in

an email they are about to receive into a page on an attacker-controlled website; a similar attack works against verification codes regardless of whether they are in emails. This attack is similar to the Gelernter et al. [33] attack on SMS-based password recovery (see Section 8.4.5).

8.4.5 SMS Reset

There are a variety of issues associated with the use of SMS reset messages, many of which are analogous to issues associated with email reset.

- Since codes sent to mobile phones typically contain only six numeric digits⁹, i.e. there are only 10^6 possibilities, this means that it may be possible to successfully guess a code value. It is therefore imperative that websites limit the number of entry attempts a user is permitted.
- As noted in Section 8.4.1, Gelernter et al. [33] describe an MitM attack on password recovery: the registration process for a malicious website is used to gather the answers to personal security questions in order to successfully complete request validation for a website with which the user already has an account. Of course, this MitM step does not complete the attack, as the attacked website will typically, after receiving the correct answers to security questions, send a verification code (e.g. via SMS) to the genuine user. To complete the attack, the malicious website (as part of its registration procedure) tells the user to expect a message containing a verification code which should be entered into the registration page. If the user enters the verification code received from the attacked website, the malicious website will now have what it needs to complete the process of capturing the user's account. The only defence against this attack is for the message containing the verification code to make it clear which website it is intended for (i.e. the attacked website rather than the malicious website).

This attack will work against verification codes sent via email or SMS, although it is much harder for the sending website to make clear how the code should be used in a 160-character SMS message than in an email. Indeed, Gelernter et al. [33] describe the results of a detailed survey of the use of SMS-based verification codes, with the goal of understanding both how clear the SMS messages are, and how likely users are to mistakenly enter a code intended for one website into a box on a different website. They go on to make a series of recommendations regarding how SMS messages should be designed to minimise the likelihood that

⁹Google and Dropbox both use 6-digit verification codes

a user can be deceived. They also recommend that the verification code should have a short validity period.

- In some countries, network coverage in rural areas is often poor or non-existent, meaning it may not be possible for a user to obtain a verification code sent via SMS. Similarly, if a user is travelling, he/she may not be able to receive the SMS. Also, SMS messages are occasionally delayed, which may cause the user to initiate many requests.
- Just as for emails, the security of SMS reset relies on the assumption that an SMS message is a secure channel to the user. Again, as for email reset, this assumption can be invalidated in a range of ways, e.g. as follows.
 - Guri et al. [33] describe an attack based on an apparently harmless Android app which can monitor and redirect SMS messages to an attacker server. If such an app was installed, then a password recovery SMS might be compromised.
 - If the phone to which the SMS is sent is stolen then the contents of the SMS message could be compromised; even if the phone is password-locked, the message may still be compromised since many phones display the contents of received SMS messages without requiring the phone to be unlocked.
 - SMS messages could be intercepted at the network operator, e.g. by a malicious employee, or when sent over the air interface if the air interface link is not encrypted (as is the case in some countries). The message could also be intercepted by an unauthorised base station (an IMSI catcher or ‘Stingray’) [62] or by exploiting weaknesses in implementations of the SS7 protocol used by telecommunication companies to communicate with each other [91].
 - If the user changes his/her phone number and forgets to notify the website, then the SMS message will be received by the new owner of that number.

8.4.6 Other Issues

We conclude this discussion of existing password recovery techniques by briefly reviewing certain other issues which have a bearing on the security and/or usability of the password recovery process.

- Use of an old password as the only means of user authentication for password recovery is clearly dangerous, since one reason a user might replace a password is because it has been, or is suspected of having been, compromised. It also

seems relatively unlikely that a user will recollect an old password if they cannot remember their current password.

- Many web services use an email address as a user ID, e.g. Amazon, Facebook, Dropbox, Instagram, Twitter and LinkedIn. This has the major advantage of making it easy for the user to remember their account identifier, and it is also convenient for web service providers to deliver account-related functions via email as well as simplifying registration. However, as discussed by Jin et al. [51], there are significant risks associated with such an approach, not least that an attacker automatically knows one of the two parts of a user credential set, thereby facilitating attacks on the password recovery process.
- Password hints may reveal sensitive information, potentially endangering user security or privacy. For example, the Google password hint shown in Figure 8.1 indicates when the user last changed his/her password – this reveals that Google was being used at that time. Whilst this may not seem so significant, for some sites this could be far more revealing.

8.5 Towards Secure Password Recovery

We next provide recommendations on best practice in implementing a password recovery system. These recommendations are based on the analysis given in the previous section. We consider the two most security-significant steps in password recovery, namely request validation and password re-establishment.

8.5.1 Recovery Request Validation

As discussed in Section 8.3.3, two main security-related steps are commonly used during request validation, namely use of a CAPTCHA and security questions. The CAPTCHA prevents automated attacks on password recovery, and appears a reasonable step to include. Security questions are used to prevent malicious triggering of password recovery, which could otherwise be done on a large scale, e.g. using databases of email addresses (since user names are often the same as email addresses). However, the use and effectiveness of security questions is questionable for a variety of reasons. Firstly, as discussed in Section 8.4.1, the answers can sometimes be readily guessed or obtained via social media, and, most seriously, the MitM attacks of Gelernter et al. [33] suggest that they offer relatively little protection. Secondly, there are significant privacy issues, since use of such questions involves gathering personal data, which could be misused and could also cause issues with respect to privacy regulations such as GDPR [16].

Thirdly, there are also usability issues, in that not all answers to questions can always be readily recalled. It is therefore highly questionable whether using security questions as a filtering mechanism is worth the trouble, since while it offers limited security it also has significant negative privacy and usability disadvantages.

8.5.2 Password Re-establishment

Depending on how password re-establishment works, there are a number of important recommendations which emerge from the analysis in Section 8.4. We look separately at three main areas: email reset, SMS reset, and the use of trustees.

Email reset

If email is used to re-establish a shared password, then the website must try to minimise the chance that the email reaches an incorrect recipient. This means, for example, that the email address to which the message is to be sent should ideally have been used (or confirmed by the user) recently. This could be achieved by asking the user to enter the email address to which the email should be sent, and only if this matches with the address stored by the website should it be used.

If verification codes are sent via email, then the discussion in Section 8.4.3 suggests that: (a) they should use as large an alphabet as possible (e.g. including letters, digits and punctuation), and be as long as possible, subject to usability constraints; and (b) they should expire after a short period of time.

URLs sent in an email should: (a) have a short expiry period (just as for codes); (b) contain a random (secret) string which should be sufficiently long to make successful guessing attacks highly improbable; and (c) start with 'https', so that the connection to the server is secure.

SMS reset

If SMS is used to deliver the means to re-establish a password, then very similar requirements to those for email apply. That is, regardless of what information is sent in the SMS, the website should try to ensure that the phone number is current, e.g. by requesting the user to re-enter the number, and only if it matches with the number stored by the website should it be used. Apart from the requirements applying to verification codes and links previously mentioned, Gelernter et al. [33] give a list of recommendations aimed specifically at verification codes sent via SMS. In particular: (a) the SMS message should be designed so that the code is not displayed on the phone screen when locked, i.e. forcing the user to unlock the phone to obtain the code; and

(b) the wording of the SMS should minimise the chance that the user can be deceived into submitting the code to the wrong website, including indicating the identity of the sending website, the purpose of the message, and a warning not to divulge the code to any other person or website.

Trustee-based password recovery

Since significant risks arise from out of date trustee data, as part of the recovery request process the account holder should be asked to verify contact details for the trustee(s) to be used. For example, the user could be asked to re-enter trustee email addresses or phone numbers, with the website only using them if they match already stored values. Websites could also periodically require users to revalidate trustee data. Clear instructions should be sent with the password recovery information sent to a trustee, to minimise the chance that a trustee is misled (e.g. via social engineering) into disclosing the recovery data to the wrong party.

8.6 Conclusions

We have given a general model for the password recovery process, and we have also examined the range of ways in which this model is instantiated by today's websites. We then provided the first comprehensive review of known security and privacy weaknesses in existing approaches to password recovery; we also examined usability issues. This then allowed us to make a series of recommendations regarding how best a password recovery process should be designed.

There is clearly a need for further research in this important area, as well as new, more secure, ways of performing recovery. Almost all the techniques in common use are to some extent flawed, and many also pose privacy risks.

Chapter 9

Real-world Email-Based Password Recovery

9.1 Email-Based Password Recovery

As discussed in the previous chapter, most websites using passwords also implement password recovery to allow users to re-establish a shared secret if the existing value is forgotten; many such systems involve sending a password recovery email to the user, e.g. containing a secret link. The security of password recovery, and hence the entire user-website relationship, depends on the email being acted upon correctly; unfortunately, as we show, such emails are not always designed to maximise security and can introduce vulnerabilities into recovery.

To understand better this serious practical security problem, we surveyed password recovery emails for 50 of the top English language websites. We investigated a range of security and usability issues for these emails, covering their design, structure and content (including the nature of the user instructions), the techniques used to recover the password, and variations in email content from one web service to another. We found that many well-known web services, including Facebook, Dropbox, and Microsoft, suffer from recovery email design, structure and content issues.

This is, to our knowledge, the first study of its type reported in the literature. This study has enabled us to formulate a set of recommendations for the design of such emails.

The remainder of this chapter is structured as follows. Section 9.2 provides the motivation for the study. In Section 9.3 we review the use of password recovery emails; Section 9.4 then reviews the risks associated with email-based password recovery, where this review forms the basis for our subsequent analysis. Section 9.5 gives the scope and

methodology for the study, and Section 9.6 provides the main findings. In Section 9.7 we use these findings to provide recommendations on the design of email-based password recovery. We briefly discuss disclosure of our findings in Section 9.8. Finally, Section 9.9 concludes the chapter.

9.2 Motivation and Methodology

Websites often offer multiple password recovery options, e.g. based on security questions, a registered email address, or a mobile number. Email-based password recovery is widely implemented [12, 32], and is likely to remain in wide use for years to come. In a typical such system, when a user initiates password recovery a *password recovery email* is sent to the user’s registered email address. The recovery email, see e.g. Figure 9.1, will typically contain instructions on how to recover the password; these instructions might include clicking on a link or entering a verification code or temporary password.

Since email-based password recovery is widely used, it is important to understand whether it is secure in practice. Unfortunately, as we show, some recovery emails are not well-designed and can introduce vulnerabilities into the recovery process. In this chapter, we analyse password recovery emails by examining their content and design, and use this analysis to make recommendations regarding their design. To our knowledge there are no previously published studies on the design of recovery emails, despite the potential of poor design to give rise to serious vulnerabilities. This observation has motivated the work described here.

To understand better the size and nature of this serious practical security problem, we manually surveyed password recovery emails for 50 of the top English language websites¹. The study examined the content of recovery emails and evaluated their security and usability, including: their ease of use, their overall design, the clarity of instructions, and the techniques used to recover the password.

9.3 Password Recovery Emails

9.3.1 Use of Recovery Emails

Email-based password recovery (as discussed in the previous chapter) involves an email being sent to the user by the web service; this email contains a secret which, on the assumption that the email can only reach the intended user, is used to authorise the password recovery request. The email may also contain information which helps the user to reset their password at the website. These emails vary in content and design,

¹The list was taken from <https://majestic.com/reports/majestic-million>.

although in all cases it is vital that the email is not available to third parties who could use the secret in the email to hijack the recovery process. Given that correct use of the email by its recipient is security-critical, the content and design of the email are clearly of importance in practice, since they will affect how the email is treated by the user.

Figure 9.1 shows a Dropbox password recovery email. Such emails typically include a user greeting, the email purpose, instructions (possibly including what to do if the user did not request recovery), and contact details. The instructions will include a (one-time-secret) mechanism enabling password recovery, e.g. a URL (link) or code/temporary password. Clicking on such a link redirects the browser to a page enabling the user to set up a new password. Verification codes/temporary passwords typically give temporary access to a user account purely to establish a new password, e.g. as used by Amazon and Wikipedia.

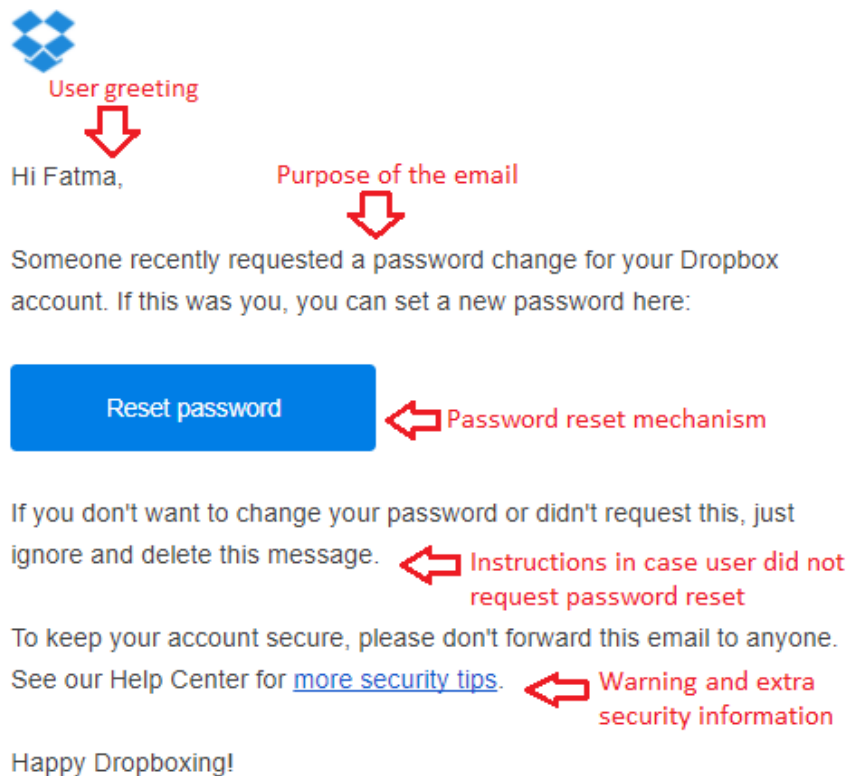


Figure 9.1: Example password recovery email (from dropbox.com)

9.3.2 Email Structure

In general, emails can be divided into *header* and *body* [76]; however, for our purposes we further divide password recovery emails into four components: the email preheader, email header, email body and email signature (where the preheader and signature are parts of the body), as follows.

- **Email preheader (Johnson Box² or Snippet)**. This corresponds to the first few words of an email, the exact length being determined by the email client. It is typically displayed by the email client after the subject header field (see below), to give the recipient an idea of what the email is about³. Figure 9.2 shows an example of a Gmail preheader. Despite its variability across clients, we discuss below the importance of the preheader in ensuring that emails are handled appropriately by their recipients. Given the small number of email clients in widespread use, it is relatively easy for email senders to check the appearance of the preheader for the vast majority of users.

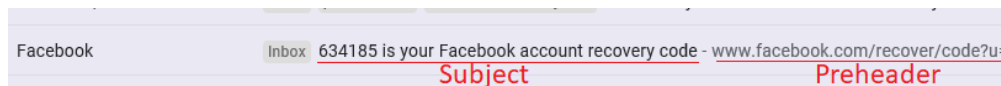


Figure 9.2: Message preheader example

- **Email header**. The header can only contain printable US-ASCII characters (except for carriage return, line feed, and colon), [76], and contains message metadata. It is broken into fields, each starting with a name followed by a colon (:). Examples of key header fields are briefly described below; many of these fields are displayed by the email client, along with the body of the message.
 - The **subject** field is chosen by the sender, and is commonly used to give a summary of the email.
 - The **to** and **from** fields contain the email addresses of the sender and recipient, as specified by the sender.
 - The **orig-date** indicates when the email was sent.
 - The **message-ID** is a globally unique email identifier, [71, 76].

Email clients will typically display the *subject*, *to*, *from* and *orig-date* fields in lists of received emails; other header fields, e.g. the *message-ID*, are typically

²<http://hyperlinkcode.com/blog/html-johnson-box/>

³The term ‘preheader’ is perhaps misleading since it is displayed after the subject header field, and actually occurs in the email after the header.

optionally viewable. For example, the Gmail *show original* button will reveal the entire email header.

- **Email body.** This constitutes the contents of the email. Email bodies were originally simply text strings, but MIME [30] allows bodies in a range of formats and with multiple parts. Today many email bodies contain an HTML-formatted part, which is by default what is displayed; this is thus the most important part for our analysis, and we look in detail at the information included in recovery email bodies below.
- **Email signature.** This is included at the end of the email body in a format determined by the sender’s email client and the user. Its purpose is to provide information about the sender in an accessible form.

9.3.3 Email Client Features

A range of email information can be displayed by an email client, some of which is computed by the client rather than being contained in the email itself. Examples of information of this latter type include the following.

- **Sender Policy Framework (SPF)** [58] is an email-validation system designed to detect and block email spoofing. It allows receiving email exchangers to check that an incoming email comes from a host authorised by the administrators of the originating domain.
- **DomainKeys Identified Mail (DKIM)** [19] aims to prevent email spoofing by allowing the receiver to check that an email was authorised by the owner of the originating domain.
- **Domain-based Message Authentication, Reporting and Conformance (DMARC)** [59] is an email-validation system designed to combat certain techniques often used in phishing and email spam, such as emails with forged sender addresses (in the *from* field) appearing to originate from legitimate organisations.

9.4 Risks from Recovery Emails

We next review threats that can arise from poorly designed recovery emails. Of course, the use of email in password recovery has intrinsic risks, as described in the previous chapter. The list below was compiled by combining the discussion of the design of password recovery SMSs due to Gelernter et al. [33], with observations derived from our study.

- **Poor instructions.** It is important that the user is given clear instructions on how to use the email, and an indication of its sensitivity. Some emails give very brief instructions, e.g. limited to what the user should do to proceed, whereas others give almost too much information and are very hard to follow; this can be challenging, especially for non-technical users. Most seriously, the lack of clear instructions could lead to disclosure of the secret in the email, e.g. by forwarding the email to a third party, thus compromising password recovery.
- **Poor readability.** Some HTML-formatted emails use small fonts and/or small buttons, which can make reading them hard especially for those with impaired vision.
- **Lack of easily recognised source.** Some recovery emails lack clear information as to their source, e.g. a usable email address; in such cases the user may treat it as a phishing email or it may be blocked by a spam filter.
- **Email header or preheader leaking confidential information.** In some cases the temporary password or validation code is included in the header or preheader, which means that it might be available to anyone with temporary access to the phone, even if locked.
- **Lack of contact details.** Some emails lack contact details for use if the recovery process fails. Contact details are often necessary as the source email address may not be monitored for replies.
- **Lack of instructions if recovery not requested.** Recovery emails may lack instructions on what to do if password recovery was not requested, e.g. indicating an attempt to gain unauthorised access to the account.
- **Spam filter issues.** If the email is not constructed carefully, then there is an increased danger that it will be blocked by a spam filter [32], hence preventing the user from performing password recovery.

9.5 A Study of Existing Practice

9.5.1 Scope of Study

We examined 50 widely used English language sites. We looked at only 50 websites because we needed to manually register at each site, trigger a recovery email, and then carefully examine it. The average time to perform the first two steps was 20 minutes, and so it took around 17 hours simply to gather the data. The process could

Table 9.1: The 50 websites tested in the study

Number	Website name	Number	Website name
1	google.com	26	creativecommons.org
2	facebook.com	27	issuu.com
3	twitter.com	28	wix.com
4	microsoft.com	29	oracle.com
5	linkedin.com	30	imdb.com
6	instagram.com	31	slideshare.net
7	adobe.com	32	paypal.com
8	en.wikipedia.org	33	go.com
9	itunes.apple.com	34	myspace.com
10	vimeo.com	35	archive.org
11	pinterest.com	36	www.ncbi.nlm.nih.gov
12	yahoo.com	37	washingtonpost.com
13	amazon.com	38	cpanel.net
14	tumblr.com	39	bloomberg.com
15	github.com	40	ebay.com
16	mozilla.org	41	telegraph.co.uk
17	sourceforge.net	42	ibm.com
18	nytimes.com	43	hp.com
19	soundcloud.com	44	cnet.com
20	bbc.co.uk	45	dailymail.co.uk
21	reddit.com	46	opera.com
22	weebly.com	47	imgur.com
23	dropbox.com	48	debian.org
24	theguardian.com	49	twitch.tv
25	forbes.com	50	surveymonkey.com

not be automated since user registration and password recovery requests require user interactions, e.g. solving a CAPTCHA or validating the email address by clicking on a link or entering a temporary code.

We chose the 50 highest-ranking sites employing email-based password recovery from the list of most-visited websites provided by majestic.com (chosen because it offers information free of charge). In a number of cases, we were not able to study a website on the list; if so we simply moved on to the next site in the list, continuing until we had the results from 50 sites. Sites were omitted from our study if they did not offer the ability to create an account, if creating an account required taking out a paid subscription, or if an organisation (e.g. [flickr.com](https://www.flickr.com) and [yahoo.com](https://www.yahoo.com)) operated multiple sites in which case we skipped second and subsequent related sites. The full list of websites we tested is given in Table 9.1.

9.5.2 Methodology

We performed the following three steps with each of the 50 chosen websites. **Account creation/registration** involved giving a user name, password, and any other information requested. We **initiated password recovery** by making a password recovery request and recording the received recovery email. Finally we **manually analysed** the password recovery email. In this analysis we first tested each email against the design risks listed in Section 9.4, and in each case considered whether the email suffered from that risk. This generated the statistical results given in Section 9.6.5 below. We then looked at each of the four components of the email and noted any examples of particularly good or bad practice; the results of this examination are summarised in Sections 9.6.2–9.6.4.

9.6 Results

9.6.1 Example Recovery Emails

Figures 9.3 and 9.4 provide examples of recovery emails suffering from issues of the type highlighted in Section 9.4.

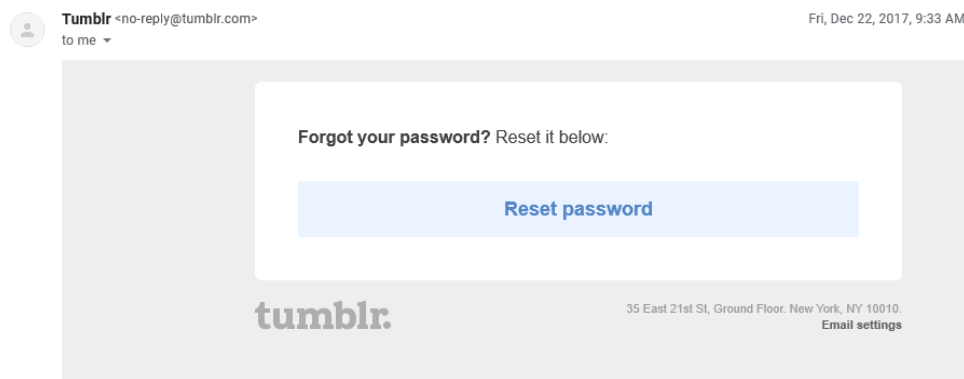


Figure 9.3: Example recovery email: Lacking information

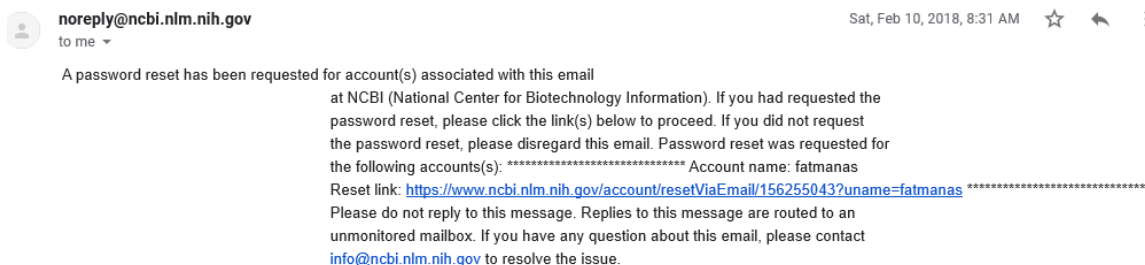


Figure 9.4: Example recovery email: Readability issue

9.6.2 Email Header

We examined the email header for all 50 recovery emails. We focussed on the *subject* and *from* fields.

subject

Among the websites we tested, Facebook (see Figs. 9.2 and 9.5) was the only one which included a verification code in the *subject* field (as opposed to the message body). An email client may display the *subject* fields of received emails even when a phone is locked, increasing the risk that the code will be seen by a third party.

from

This field includes an email address and, optionally, a ‘name’.

- If the sender name is absent then the receiver might doubt its legitimacy. Six of the 50 emails did not state the name of the sender in this field, e.g. the www.ibm.com recovery email had *from* field ‘ibmacct@us.ibm.com’, i.e. without the name of the service. Similar issues arise in SMS-based password recovery [33].
- In some cases, the email address did not identify the type of service, i.e. relating to password recovery, e.g. no-reply@tumblr.com.
- Of the 50 recovery emails we examined, 21 (i.e. 42%) were sent from an email address to which replies are not permitted (e.g. noreply@bloomberg.com). Other websites did not use *noreply* but test emails sent to the address nevertheless bounced, e.g. washingtonpost.com. If a contact email is not provided then this

is very unhelpful for users who encounter difficulties; it could also mean that the recovery email is blocked by an email client spam filter.

- Some emails were not constructed in such a way as to pass the SPF, DKIM and DMARC anti-spoofing tests, e.g. `ibm.com` failed DMARC testing.

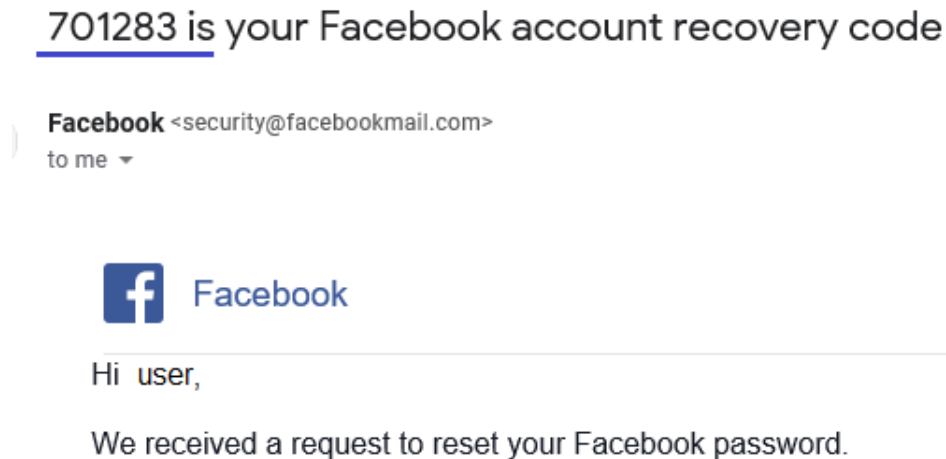


Figure 9.5: Facebook password recovery email

9.6.3 Message Body

Within the message body, 28 of the emails contained a personalised **user greeting** (e.g. a name or email address), whereas the others had a generic greeting or none at all. A personalised greeting is to be preferred since it helps reassure the user that the email is genuine. Information as to when the request was initiated was included in three of the recovery emails we examined; this helps the user verify that the email corresponds to a legitimate request.

46 of the 50 emails included a URL link for password recovery. We have a number of observations.

- 41 (82%) of the emails did not specify how long the URL would remain valid. As discussed in the previous chapter, it is desirable for links to have a short lifetime, since they represent a security risk. Indeed, some tokens were still valid even three months after the experiment, e.g. creativecommons.org/ and bloomberg.com. In other cases the password recovery page was available after token expiry, allowing the user to generate a new token without user authentication, e.g. www.mozilla.org.

- 20 (40%) of the emails did not give a way of disabling the link in the event of accidental initiation of password recovery, increasing the risk of compromise.
- Many emails require the user to copy and paste the URL into a browser, which could expose the user to phishing attacks. For example, an attacker could trigger password recovery for the user, and in parallel instruct the user to copy the URL in a recovery email into an attacker webpage.

Websites give varying advice for the case that the user did not request recovery. Some, e.g. www.dropbox.com and www.wix.com, simply suggested ignoring the email; this may leave the user very concerned as to what is happening. Others suggest contacting the web service, e.g. by filling an online form, contacting a call centre (e.g. www.nytimes.com), replying to the email (e.g. www.vimeo.com), or clicking on a link (e.g. Facebook.com). Other sites, e.g. www.forbes.com, did not address the issue at all.

47 (94%) of the recovery emails failed to advise the user to protect the code or URL. Giving such a warning could help to reduce the risk of code sharing and phishing attacks. The same issue arises when using SMS-based password recovery.

9.6.4 Message Signature

As many as 21 of the emails lacked a signature, e.g. www.wikipedia.org/. Others contained only the URL, e.g. www.microsoft.com and www.dropbox.com.

9.6.5 Summary of Risks

As described above, we conclude by summarising the observed frequency of the seven risk types given in Section 9.4. The results are given in tabular form in Tables 9.2 and 9.3, in which a black spot is used to indicate a website whose recovery email had an issue of the relevant type.

- **Poor instructions.** As many as 22 (44%) of the 50 emails contained no further instructions on the use of the provided link or code, as was the case for www.tumblr.com/.
- **Poor readability.** Three (6%) of the emails used small fonts or condensed blocks of text, giving potential readability issues, e.g. www.ncbi.nlm.nih.gov/.
- **Lack of easily recognised source.** In five (10%) of the cases, the *from* field contained an email address with no obvious link to the originating website, making recognition problematic, e.g. [dailymail.co.uk](mailto:password@and.co.uk) uses `password@and.co.uk`.

- **Email header or preheader leaking confidential information.** Only six of the 50 emails contained a secret code/temporary password; the others used a link or a combination of code and link; of these six, two (4%) leaked the code via the header or preheader field. These two websites are two of the most widely used sites, i.e. [google.com](https://www.google.com) leaked the code via the preheader and [facebook.com](https://www.facebook.com) included the code in the header.
- **Lack of contact details.** As many as 35 (70%) of the recovery emails did not provide any contact details, e.g. for use in the event of problems, although some of these (e.g. [pinterest.com](https://www.pinterest.com)) provided links to websites for further information (e.g. containing FAQs).
- **Lack of instructions if recovery was not requested.** 20 (40%) of the emails, including that from [google.com](https://www.google.com), failed to provide any user guidance for the case where the recovery email was not requested.
- **Spam filter issues.** In our experiment, two (4%) of the emails were routed into a spam folder by the email client, e.g. the email from creativecommons.org. This suggests that the source of these emails could have done more to prevent such an undesirable event.

All 50 of the emails suffered from at least two of the above issues. This suggests that most websites need to improve the design of their recovery emails. Although some the identified issues seem minor, even if they cause a problem in just 1% of recovery attempts this could potentially result in a large number of compromised or unusable accounts.

Table 9.2: The 50 websites with issues identified (Part 1)

Number	Website name	Poor instructions	Poor readability	Lack of easily recognised source	Email header or preheader leaking confidential information	Lack of instructions if recovery not requested.	Spam filter issues.	Lack of contact details.
1	google.com	•			•	•		•
2	facebook.com				•			•
3	twitter.com							•
4	microsoft.com	•				•		•
5	linkedin.com					•		•
6	instagram.com	•						•
7	adobe.com					•		•
8	en.wikipedia.org							•
9	itunes.apple.com							
10	vimeo.com							•
11	pinterest.com							•
12	yahoo.com		•					•
13	amazon.com					•		•
14	tumblr.com	•				•		•
15	github.com					•		•
16	mozilla.org	•				•		•
17	sourceforge.net	•				•		
18	nytimes.com					•		
19	soundcloud.com							•
20	bbc.co.uk							•
21	reddit.com							•
22	weebly.com	•				•		•
23	dropbox.com	•						•
24	theguardian.com	•		•				
25	forbes.com					•		
26	creativecommons.org	•		•		•	•	•
27	issuu.com	•						•
28	wix.com	•						•
29	oracle.com	•						
30	imdb.com	•						•
31	slideshare.net							•
32	paypal.com	•				•		•
33	go.com							

Table 9.3: The 50 websites with issues identified (Part 2)

Number	Website name	Poor instructions	Poor readability	Lack of easily recognised source	Email header or preheader leaking confidential information	Lack of instructions if recovery not requested.	Spam filter issues.	Lack of contact details.
34	myspace.com	•				•	•	•
35	archive.org	•						•
36	www.ncbi.nlm.nih.gov		•					
37	washingtonpost.com	•				•		
38	cpanel.net	•	•					•
39	bloomberg.com	•						
40	ebay.com							
41	telegraph.co.uk					•		
42	ibm.com	•		•		•		•
43	hp.com					•		
44	cnet.com							
45	dailymail.co.uk	•		•		•		•
46	opera.com			•				•
47	imgur.com							•
48	debian.org							•
49	twitch.tv							•
50	surveymonkey.com							

9.7 Recommendations

The issues identified in our survey enable us to make the following recommendations for the design of recovery emails. We divide our recommendations under seven headings corresponding to the risks identified in Section 9.4.

- **Instructions.** A personal greeting is desirable to increase user confidence. Where possible the email should include a clickable URL, and avoid asking the user to copy/paste the URL. Users should be warned against sharing or forwarding password recovery emails. More detailed information should be provided via links to avoid creating congested, unreadable emails. The recovery mechanism should have a limited validity period.
- **Readability.** The message should be easily comprehensible, i.e. the font/font size must be chosen with care.
- **Source recognition.** The name of the sender and the sender email address should match.
- **Email header and preheader design.** The header and preheader must not include any secret recovery information, e.g. a code or link, since preheaders can appear on a mobile phone screen even when locked. Users may also be tempted to perform recovery just by reading the preheader without fully viewing the email, thereby missing any instructions/warnings in the email. This issue can be more serious for users who view emails on a mobile phone, where the preheader can often be a ‘notification’. Similarly, the *subject* field in an email header should state clearly the purpose of the email along with the service name.
- **Contact details.** The email should give a valid email address or a support link for use if the user is unable to recover a password. Ideally the email signature should include the name of the web service along with a logo, an email address, a phone number, a website URL and a physical address.
- **Instructions if recovery not requested.** Users should be given a means to disable the password recovery mechanism and to report unsolicited recovery emails to assure users of their security.
- **Spam filter issues.** The email should be generated and sent in such a way as to minimise the risk that it is blocked by a spam filter.

9.8 Disclosure

We sent a copy of the paper [4] summarising our findings to all 50 of the websites we tested. So far we have received replies from six of the 50 sites: [Vimeo.com](https://www.vimeo.com), [wikipedia.org](https://www.wikipedia.org), [wix.com](https://www.wix.com), [cnet.com](https://www.cnet.com), [dailymail.co.uk](https://www.dailymail.co.uk) and [imdb.com](https://www.imdb.com). All the replies stated that the recipient was pleased to be contacted and that they were willing to make improvements to their email-based password recovery system based on the findings of the paper. Examples of the responses are included in Appendix B.

9.9 Conclusions

We examined the password recovery emails sent by 50 of the most widely visited websites, and found that they all suffer from at least two types of defect. As a result it seems likely that some password recovery processes are unnecessarily prone to compromise. Although email-based password recovery allows the service provider to provide clear and detailed instructions, it would appear that some providers have given relatively little attention to the design of their recovery emails. In some cases the email structure, wording and content clearly give rise to potential vulnerabilities. Our survey has allowed us to make a series of recommendations regarding how best a password recovery email should be designed to maximise security and usability.

Part IV

Conclusions

Chapter 10

Concluding Matters

10.1 Summary of Contributions

We introduced a general model for password generators, and considered all the existing proposals in the context of this model. The model enabled us to analyse the advantages and disadvantages of a range of approaches to building such systems. It also enabled us to propose certain new options to enhance such schemes.

We described in detail the design of the AutoPass password generator. Its use of a server, in particular to store password offsets, PRML specifications and user preferences, helps to remove the shortcomings present in all previously proposed password generators. At the same time, this server is only partly trusted, and does not have the means to recover individual user passwords. A prototype implementation of AutoPass has been developed and used to conduct user trials, to verify that the desired high level of usability can be achieved. The main findings of these trials, which established the usability of the system, were reported.

We also gave a general model for the password recovery process, and we examined the range of ways in which this model is instantiated by today's websites. We provided the first comprehensive review of known security and privacy weaknesses in existing approaches to password recovery; we also examined usability issues. This allowed us to make a series of recommendations regarding how best a password recovery process should be designed.

We examined the password recovery emails sent by 50 of the most widely visited websites, and found that they all suffer from at least two types of defect. As a result it seems likely that some password recovery processes are likely to be unnecessarily prone to compromise. Although email-based password recovery allows the service provider to provide clear and detailed instructions, it would appear that some providers have

given relatively little attention to the design of their recovery emails. In some cases the email structure, wording and content clearly give rise to potential vulnerabilities.

10.2 Directions for Future Work

Whilst AutoPass has been prototyped and a small user trial has been conducted, the trial was very limited in scope and its goal was merely to prove that the system was usable. One obvious next step would be to make any adjustments deemed necessary from a careful analysis of the trial, and to then perform a larger scale trial over a much longer time period. This would probably necessitate paying participants. Of course, in parallel we hope to receive unprompted feedback from users, given that the AutoPass Chrome extension will soon be publicly available for free download from the Google store. Another important next step will be to enhance AutoPass to add functionality currently missing, including optional use of digital objects and the provision of a default password policy. In addition, we plan to create versions for all the most popular browsers, operating systems and platform types, thereby addressing the compatibility issues identified by three of the original 17 trial participants. More generally, whilst password generators have considerable potential, their use is still relatively limited. Even password managers are not used to the extent that their proponents would like. As a result, many users are still vulnerable to simple impersonation attacks through the use of weak passwords, and through use of the same password with many sites. Further research is urgently required on the best approach to try to understand what would enable users to switch to more secure ways of managing their authentication credentials.

There is clearly a need for further research in password recovery, as well as new, more secure, ways of performing recovery. Almost all the techniques in common use are to some extent flawed, and many also pose privacy risks. One direction for future work would be to conduct large scale practical trials to try to understand better how users can interact with password recovery systems both securely and reliably. Password recovery is just one aspect of a larger topic we refer to as *account recovery*. This also covers the case where a user may forget their user name, and/or other information necessary to access an account at a website. This is a topic that has been the focus of very little published research, and a survey of approaches used by major websites would be an obvious way of starting to gain an understanding of the possible strengths and weaknesses of various approaches.

10.3 The Future of Passwords

Passwords have been around for decades, and they are not going to disappear any time soon. The work described in this thesis aimed to explore ways to strength password-based authentication until mature, readily usable and cheap alternatives can be deployed to either replace passwords or provide a secondary means of authentication.

One approach to strengthening password-based authentication is the use of an extra layer of authentication, e.g. involving biometric or activity-based verification. In biometric verification a user is authenticated based on an individual characteristic, e.g. his/her fingerprint details, finger vein pattern, or retina/iris characteristics. Such solutions can be implemented on smart phones, so that the phone is used to help perform authentication (cf. FIDO [26]). In activity-based verification (essentially a different class of biometric authentication), a user is authenticated by his or her behaviour, such as user-mouse interactions, keyboard activity, or software interactions (including game playing).

Another technique that can be used as a secondary authentication method involves the use of contextual information, e.g. IP address, browser information, cookies cached on the browser, the time of the login, and the use of trusted devices. More generally, many modern user authentication methods, including biometric and contextual authentication, involve the use of machine learning techniques, which potentially require gathering significant amounts of data about the user, including the user's behaviour and devices. Whilst access to such data can improve authentication accuracy, it is also a major threat to user privacy as well as being a target for attack.

From the user perspective, password-based authentication can be made more secure through the use of password management solutions, such as password managers and password generators. However, the challenge remains to persuade users to implement such solutions rather than reusing passwords across multiple accounts or using weak passwords.

Bibliography

- [1] C. Z. Acemyan, P. Kortum, J. Xiong, and D. S. Wallach. 2FA might be secure, but it's not usable: A summative usability assessment of googles two-factor authentication (2FA) methods. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 62, pages 1141–1145. SAGE Publications Sage CA: Los Angeles, CA, 2018. [15](#)
- [2] F. Al Maqbali and C. J. Mitchell. Password generators: Old ideas and new. In S. Foresti and J. Lopez, editors, *Information Security Theory and Practice — 10th IFIP WG 11.2 International Conference, WISTP 2016, Heraklion, Crete, Greece, September 26-27, 2016, Proceedings*, volume 9895 of *Lecture Notes in Computer Science*, pages 245–253. Springer, 2016.
- [3] F. Al Maqbali and C. J. Mitchell. Autopass: An automatic password generator. In *International Carnahan Conference on Security Technology, ICCST 2017, Madrid, Spain, October 23-26, 2017*, pages 1–6. IEEE, 2017.
- [4] F. Al Maqbali and C. J. Mitchell. Email-based password recovery — risking or rescuing users? In *2018 International Carnahan Conference on Security Technology, ICCST 2018, Montreal, QC, Canada, October 22-25, 2018*, pages 1–5. IEEE, 2018. [127](#)
- [5] F. Al Maqbali and C. J. Mitchell. Web password recovery: A necessary evil? In K. Arai, R. Bhatia, and S. Kapoor, editors, *Proceedings of the Future Technologies Conference, FTC, Vancouver, Canada, November 15-16, 2018*, pages 324–341. Springer, 2018.
- [6] M. Alsaleh, M. Mannan, and P. C. van Oorschot. Revisiting defenses against large-scale online password guessing attacks. *IEEE Trans. Dependable Sec. Comput.*, 9(1):128–141, 2012. [82](#)
- [7] A. Anne and M. A. Sasse. Users are not the enemy. *Commun. ACM*, 42(12):40–46, 1999. [1](#)

- [8] R. Biddle, M. Mannan, P. C. van Oorschot, and T. Whalen. User study, analysis, and usable security of passwords based on digital objects. *IEEE Transactions on Information Forensics and Security*, 6(3):970–979, 2011. 30
- [9] J. Bonneau. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 538–552, 2012. 15
- [10] J. Bonneau, E. Bursztein, I. Caron, R. Jackson, and M. Williamson. Secrets, lies, and account recovery: Lessons from the use of personal knowledge questions at google. In A. Gangemi, S. Leonardi, and A. Panconesi, editors, *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 141–150. ACM, 2015. 39, 103
- [11] 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. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 553–567. IEEE Computer Society, 2012. 1, 15, 25, 34
- [12] J. Bonneau and S. Preibusch. The password thicket: Technical and market failures in human authentication on the web. In *9th Annual Workshop on the Economics of Information Security, WEIS 2010, Harvard University, Cambridge, MA, USA, June 7-8, 2010*, 2010. http://weis2010.econinfosec.org/papers/session3/weis2010_bonneau.pdf. 101, 113
- [13] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography. Springer, 2003. 9
- [14] S. Chiasson, P. C. van Oorschot, and R. Biddle. A usability study and critique of two password managers. In *Proceedings of the 15th USENIX Security Symposium, Vancouver, BC, Canada, July 31 – August 4, 2006*. USENIX Association, 2006. 27, 51, 53, 87
- [15] L. Chmielewski, J. Hoepman, and P. van Rossum. Client-server password recovery. In R. Meersman, T. S. Dillon, and P. Herrero, editors, *On the Move to Meaningful Internet Systems: OTM 2009, Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009, Vilamoura, Portugal, November 1-6, 2009, Proceedings, Part II*, volume 5871 of *Lecture Notes in Computer Science*, pages 861–878. Springer, 2009. 37

- [16] S. Chunhsien and L. Shangchien. Personal data and identifiers: Some issues regarding general data protection regulations. In *International Wireless Internet Conference*, pages 163–169. Springer, 2018. 109
- [17] G. Cluley. Lastpass vulnerability potentially exposed passwords for internet explorer users. <https://www.grahamcluley.com/2013/08/lastpass-vulnerability/>, August 2013. 22
- [18] Council of the European Union. Regulation (EU) 2016/679 (General Data Protection Regulation), April 2016. <https://publications.europa.eu/en/publication-detail/-/publication/3e485e15-11bd-11e6-ba9a-01aa75ed71a1>. 15
- [19] D. Crocker, T. Hansen, and e. M. Kucherawy. *RFC 6376, DomainKeys Identified Mail (DKIM) Signatures*. Internet Engineering Task Force, September 2011. 116
- [20] M. Dammak, O. R. M. Boudia, M. A. Messous, S. M. Senouci, and C. Gransart. Token-based lightweight authentication to secure iot networks. In *16th IEEE Annual Consumer Communications & Networking Conference, CCNC 2019, Las Vegas, NV, USA, January 11-14, 2019*, pages 1–4. IEEE, 2019. 10
- [21] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang. The tangled web of password reuse. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014. 16
- [22] M. L. Das, A. Saxena, and V. P. Gulati. A dynamic id-based remote user authentication scheme. *IEEE Trans. Consumer Electronics*, 50(2):629–631, 2004. 1
- [23] R. Dhamija and A. Perrig. Deja vu-a user study: Using images for authentication. In S. M. Bellovin and G. Rose, editors, *9th USENIX Security Symposium, Denver, Colorado, USA, August 14-17, 2000*. USENIX Association, 2000. 1
- [24] A. E. Dirik, N. D. Memon, and J. Birget. Modeling user choice in the passpoints graphical password scheme. In L. F. Cranor, editor, *Proceedings of the 3rd Symposium on Usable Privacy and Security, SOUPS 2007, Pittsburgh, Pennsylvania, USA, July 18–20, 2007*, volume 229 of *ACM International Conference Proceeding Series*, pages 20–28. ACM, 2007. 47
- [25] C. Ellison, C. Hall, R. Milbert, and B. Schneier. Protecting secret keys with personal entropy. *Future Generation Comp. Syst.*, 16(4):311–318, 2000. 37

- [26] FIDO Alliance. *FIDO UAF Protocol Specification v1.0: FIDO Alliance Proposed Standard 08*, December 2014. <https://fidoalliance.org/>. 18, 131
- [27] D. Florêncio, C. Herley, and P. C. van Oorschot. Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20–22, 2014*, pages 575–590. USENIX Association, 2014. 1
- [28] D. A. F. Florêncio and C. Herley. A large-scale study of web password habits. In C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, editors, *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8–12, 2007*, pages 657–666. ACM, 2007. 2, 18, 94
- [29] D. A. F. Florêncio and C. Herley. Where do security policies come from? In L. F. Cranor, editor, *Proceedings of the Sixth Symposium on Usable Privacy and Security, SOUPS 2010, Redmond, Washington, USA, July 14–16, 2010*, volume 485 of *ACM International Conference Proceeding Series*. ACM, 2010. 16
- [30] N. Freed and N. Borenstein. *RFC 2045, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. Internet Engineering Task Force, November 1996. 116
- [31] N. Frykholm and A. Juels. Error-tolerant password recovery. In M. K. Reiter and P. Samarati, editors, *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6–8, 2001*, pages 1–9. ACM, 2001. 37
- [32] S. L. Garfinkel. Email-based identification and authentication: An alternative to PKI? *IEEE Security & Privacy*, 1(6):20–26, 2003. 113, 117
- [33] N. Gelernter, S. Kalma, B. Magnezi, and H. Porcilan. The password reset MitM attack. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22–26, 2017*, pages 251–267. IEEE Computer Society, 2017. x, 40, 103, 105, 107, 108, 109, 110, 116, 120
- [34] N. Z. Gong and D. Wang. On the security of trustee-based social authentications. *IEEE Trans. Information Forensics and Security*, 9(8):1251–1263, 2014. 104
- [35] V. Goyal, V. Kumar, M. Singh, A. Abraham, and S. Sanyal. Compchall: Addressing password guessing attacks. *CoRR*, abs/1111.3753, 2011. 17

- [36] P. A. Grassi, J. L. Fenton, N. B. Lefkovitz, J. M. Danker, Y.-Y. Choong, K. Greene, and M. F. Theofanos. Digital identity guidelines: Enrollment and identity proofing requirements. Technical report, 2017. <https://www.nist.gov/publications/digital-identity-guidelines-enrollment-and-identity-proofing-requirements>. 12
- [37] P. A. Grassi, M. E. Garcia, and J. L. Fenton. Digital identity guidelines. *NIST Special Publication*, 800:63–3, 2017. x, 11, 12
- [38] M. Guri, E. Shemer, D. Shirtz, and Y. Elovici. Personal information leakage during password recovery of internet services. In J. Brynielsson and F. Johansson, editors, *2016 European Intelligence and Security Informatics Conference, EISIC 2016, Uppsala, Sweden, August 17-19, 2016*, pages 136–139. IEEE Computer Society, 2016. xii, 39, 40, 105
- [39] J. A. Halderman, B. Waters, and E. W. Felten. A convenient method for securely managing passwords. In A. Ellis and T. Hagino, editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*, pages 471–479. ACM, 2005. x, 28, 29, 48, 49, 51, 53
- [40] M. Hammad, Y. Liu, and K. Wang. Multimodal biometric authentication systems using convolution neural network based on different level fusion of ECG and fingerprint. *IEEE Access*, 7:26527–26542, 2019. 10
- [41] D. Hardt (editor). *The OAuth 2.0 Authorization Framework*. Internet Engineering Task Force (IETF), October 2012. 18
- [42] C. Herley and P. C. van Oorschot. A research agenda acknowledging the persistence of passwords. *IEEE Security & Privacy*, 10(1):28–36, 2012. 1
- [43] C. Herley, P. C. van Oorschot, and A. S. Patrick. Passwords: If we’re so smart, why are we still using them? In R. Dingledine and P. Golle, editors, *Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*, volume 5628 of *Lecture Notes in Computer Science*, pages 230–237. Springer, 2009. 2
- [44] M. Horsch. *Generating and Managing Secure Passwords for Online Accounts*. PhD thesis, Darmstadt University of Technology, Germany, 2018. <http://tuprints.ulb.tu-darmstadt.de/7003/>. x, 13, 14, 16, 17, 35
- [45] M. Horsch, A. Hülsing, and J. A. Buchmann. PALPAS — passwordless password synchronization. In *10th International Conference on Availability, Reliability and*

- Security, ARES 2015, Toulouse, France, August 24-27, 2015*, pages 30–39. IEEE Computer Society, 2015. [x](#), [32](#), [33](#), [35](#), [46](#), [48](#), [50](#), [55](#), [58](#)
- [46] M. Horsch, M. Schlipf, J. Braun, and J. A. Buchmann. Password requirements markup language. In J. K. Liu and R. Steinfeld, editors, *Information Security and Privacy — 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4–6, 2016, Proceedings, Part I*, volume 9722 of *Lecture Notes in Computer Science*, pages 426–439. Springer-Verlag, 2016. [35](#), [48](#), [54](#), [55](#), [58](#), [60](#), [64](#)
- [47] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 10118–3, Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions*, 3rd edition, 2004. [61](#)
- [48] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 18033–3:2010, Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers*, 2nd edition, 2010. [67](#)
- [49] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 29100, Information technology — Security techniques — Privacy framework*, 1st edition, 2011. [15](#)
- [50] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 27000, Security techniques — Information security management systems — Overview and vocabulary*, 5th edition, 2018. [9](#)
- [51] L. Jin, H. Takabi, and J. B. D. Joshi. Security and privacy risks of using e-mail address as an identity. In A. K. Elmagarmid and D. Agrawal, editors, *Proceedings of the 2010 IEEE Second International Conference on Social Computing, SocialCom / IEEE International Conference on Privacy, Security, Risk and Trust, PASSAT 2010, Minneapolis, Minnesota, USA, August 20-22, 2010*, pages 906–913. IEEE Computer Society, 2010. [109](#)
- [52] M. Just and D. Aspinall. Personal choice and challenge questions: a security and usability assessment. In L. F. Cranor, editor, *Proceedings of the 5th Symposium on Usable Privacy and Security, SOUPS 2009, Mountain View, California, USA, July 15-17, 2009*, ACM International Conference Proceeding Series. ACM, 2009. [39](#)
- [53] C. Karlof, J. D. Tygar, and D. A. Wagner. Conditioned-safe ceremonies and a user study of an application to web authentication. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2009, San Diego, California, USA, 8th February – 11th February 2009*. The Internet Society, 2009. [106](#)

- [54] A. H. Karp. Site-specific passwords. Technical Report HPL-2002-39 (R.1), HP Laboratories, Palo Alto, May 2003. Available at [https://www.labs.hp.com/publications/HPL-2002-39\(R.1\)](https://www.labs.hp.com/publications/HPL-2002-39(R.1)). x, 25, 26, 48, 49, 51
- [55] S. M. Kelly. Lastpass passwords exposed for some internet explorer users. MashableUK, <http://mashable.com/2013/08/19/lastpass-password-bug/>, August 2013. 22
- [56] J. Kelsey, B. Schneier, C. Hall, and D. Wagner. Secure applications of low-entropy keys. In E. Okamoto, G. I. Davida, and M. Mambo, editors, *Information Security, First International Workshop, ISW '97, Tatsunokuchi, Japan, September 17-19, 1997, Proceedings*, volume 1396 of *Lecture Notes in Computer Science*, pages 121–134. Springer, 1997. 61
- [57] W. M. Kharudin, N. F. M. Din, and M. Z. Jali. Password recovery using graphical method. In A. Abraham, A. K. Muda, and Y.-H. Choso, editors, *Pattern Analysis, Intelligent Security and the Internet of Things*, pages 11–20. Springer, 2015. 37
- [58] S. Kitterman. *RFC 7208, Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1*. Internet Engineering Task Force, April 2014. 116
- [59] M. Kucherawy and e. E. Zwicky. *RFC 7489, Domain-based Message Authentication, Reporting, and Conformance (DMARC)*. Internet Engineering Task Force, March 2015. 116
- [60] Y. Li, H. Wang, and K. Sun. Email as a master key: Analyzing account recovery in the wild. In *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*, pages 1646–1654. IEEE, 2018. 40
- [61] Z. Li, W. He, D. Akhawe, and D. Song. The emperor’s new password manager: Security analysis of web-based password managers. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 465–479, San Diego, CA, 2014. USENIX Association. 22
- [62] A. Lilly. IMSI catchers: hacking mobile communications. *Network Security*, 2017(2):5–7, 2017. 108
- [63] S. G. Lyastani, M. Schilling, S. Fahl, M. Backes, and S. Bugiel. Better managed than memorized? studying the impact of managers on password strength and reuse. In W. Enck and A. P. Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, pages 203–220. USENIX Association, 2018. 16, 88

- [64] M. Mannan, D. Barrera, C. D. Brown, D. Lie, and P. C. van Oorschot. Mercury: Recovering forgotten passwords using personal devices. In G. Danezis, editor, *Financial Cryptography and Data Security — 15th International Conference, FC 2011, Gros Islet, St. Lucia, February 28 – March 4, 2011, Revised Selected Papers*, volume 7035 of *Lecture Notes in Computer Science*, pages 315–330. Springer, 2011. 37
- [65] M. Mannan and P. C. van Oorschot. Digital objects as passwords. In N. Provos, editor, *3rd USENIX Workshop on Hot Topics in Security, HotSec’08, San Jose, CA, USA, July 29, 2008, Proceedings*. USENIX Association, 2008. 30
- [66] M. Mannan and P. C. van Oorschot. Passwords for both mobile and desktop computers: ObPwd for Firefox and Android. *USENIX ;login*, 37(4):28–37, August 2012. x, 30, 31, 48, 49, 51
- [67] M. Mannan, T. Whalen, R. Biddle, and P. C. van Oorschot. The usable security of passwords based on digital objects: From design and analysis to user study. Technical Report TR-10-02, School of Computer Science, Carleton University, February 2010. <https://www.scs.carleton.ca/sites/default/files/tr/TR-10-02.pdf>. 30, 32, 51, 87
- [68] D. McCarney. Password managers: Comparative evaluation, design, implementation and empirical analysis. Master’s thesis, Carleton University, August 2013. Available at <https://danielmccarney.ca/assets/pubs/McCarney.MCS.Archive.pdf>. x, 3, 19, 22, 24, 25, 33, 34
- [69] R. Morris and K. Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, 1979. 1, 15, 18
- [70] L. O’Gorman. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*, 91(12):2021–2040, 2003. x, 10, 11
- [71] S. Pasupatheeswaran. Email ‘message-ids’ helpful for forensic analysis? In *ADF 2008 — 6th Australian Digital Forensics Conference, Perth, Australia, 3 December 2008*. School of Computer and Information Science, Edith Cowan University, Perth, Western Australia, 2008. <https://ro.ecu.edu.au/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1048&context=adf>. 115
- [72] D. Pauli. KeePass looter: Password plunderer rinses pwned sysadmins. *The Register*, November 2015. http://www.theregister.co.uk/2015/11/03/keepass_looter_the_password_plunderer_to_hose_pwned_sys_admins/. 22

- [73] I. Polakis, G. Kontaxis, S. Antonatos, E. Gessiou, T. Petsas, and E. P. Markatos. Using social networks to harvest email addresses. In E. Al-Shaer and K. B. Frikken, editors, *Proceedings of the 2010 ACM Workshop on Privacy in the Electronic Society, WPES 2010, Chicago, Illinois, USA, October 4, 2010*, pages 11–20. ACM, 2010. <http://www.syssec-project.eu/m/page-media/3/social-harvest-wpes10.pdf>. 105
- [74] A. Rabkin. Personal knowledge questions for fallback authentication: security questions in the era of facebook. In L. F. Cranor, editor, *Proceedings of the 4th Symposium on Usable Privacy and Security, SOUPS 2008, Pittsburgh, Pennsylvania, USA, July 23-25, 2008*, ACM International Conference Proceeding Series, pages 13–23. ACM, 2008. 1, 39, 103
- [75] S. Ragan. Lastpass compromise: Here’s what you need to know and what you can do. CSO, <http://www.csoonline.com/article/2936254/data-protection/lastpass-compromise-heres-what-you-need-to-know-and-what-you-can-do.html>, June 2015. 22
- [76] P. Resnick (editor). *RFC 5322, Internet Message Format*. Internet Engineering Task Force, October 2008. 115
- [77] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In P. McDaniel, editor, *Proceedings of the 14th USENIX Security Symposium, Baltimore, MD, USA, July 31 – August 5, 2005*, pages 17–32. USENIX Association, 2005. x, 26, 27, 48, 49, 50, 51
- [78] J. H. Saltzer. Protection and the control of information sharing in multics. *Communications of the ACM*, 17(7):388–402, 1974. 15
- [79] H. Sanchez and J. T. Murray. Putting your passwords on self-destruct mode: Beating password fatigue. In *Workshop on Security Fatigue, WSP@SOUPS 2016, Denver, CO, USA, June 22, 2016*. USENIX Association, 2016. 2
- [80] S. E. Schechter, A. J. B. Brush, and S. Egelman. It’s no secret: Measuring the security and reliability of authentication via “secret” questions. In L. F. Cranor, editor, *Proceedings of the 5th Symposium on Usable Privacy and Security, SOUPS 2009, Mountain View, California, USA, July 15-17, 2009*, ACM International Conference Proceeding Series. ACM, 2009. 39
- [81] D. Silver, S. Jana, D. Boneh, E. Y. Chen, and C. Jackson. Password managers: Attacks and defenses. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, pages 449–464, 2014. 3, 63

- [82] R. E. Smith. Authentication: From passwords to public keys. In *booktitle*. Addison-Wesley Professional, 2001. x, 10
- [83] W. Stallings, L. Brown, and Bauer. *Computer security: principles and practice*. Pearson Education, 2012. 10
- [84] V. Stavova, V. Matyas, and M. Just. Codes v. people: A comparative usability study of two password recovery mechanisms. In S. Foresti and J. Lopez, editors, *Information Security Theory and Practice — 10th IFIP WG 11.2 International Conference, WISTP 2016, Heraklion, Crete, Greece, September 26-27, 2016, Proceedings*, volume 9895 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2016. x, 38, 39
- [85] T. Takada and H. Koike. Awase-e: Image-based authentication for mobile phones using user’s favorite images. In L. Chittaro, editor, *Human-Computer Interaction with Mobile Devices and Services, 5th International Symposium, Mobile HCI 2003, Udine, Italy, September 8–11, 2003, Proceedings*, volume 2795 of *Lecture Notes in Computer Science*, pages 347–351. Springer, 2003. 47
- [86] J. Titcomb. Hacker reveals how he could take over any facebook account and change its password. *The Telegraph*, March 2016. <http://www.telegraph.co.uk/technology/2016/03/08/hacker-reveals-how-he-could-log-in-to-any-facebook-account-and-c/>. 106
- [87] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: using hard AI problems for security. In E. Biham, editor, *Advances in Cryptology — EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2003. 82
- [88] D. Wang and P. Wang. Two birds with one stone: Two-factor authentication with security beyond conventional bound. *IEEE Trans. Dependable Sec. Comput.*, 15(4):708–722, 2018. 15
- [89] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang. Targeted online password guessing: An underestimated threat. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1242–1254. ACM, 2016. 17

- [90] L. Wang, Y. Li, and K. Sun. Amnesia: A bilateral generative password manager. In *36th IEEE International Conference on Distributed Computing Systems, ICDCS 2016, Nara, Japan, June 27-30, 2016*, pages 313–322. IEEE Computer Society, 2016. [87](#)
- [91] B. Welch. Exploiting the weaknesses of SS7. *Network Security*, 2017(1):17–19, 2017. [108](#)
- [92] M. V. Wilkes. *Time-sharing computer systems*. Macdonald and Jane’s ; American Elsevier, 1968. [15](#)
- [93] R. Wolf and M. Schneider. The passwordsitter. Technical report, Fraunhofer Institute for Secure Information Technology (SIT), May 2006. [x](#), [29](#), [48](#), [49](#), [52](#)
- [94] K.-P. Yee and K. Sitaker. Passpet: Convenient password management and phishing protection. In L. F. Cranor, editor, *Proceedings of the 2nd Symposium on Usable Privacy and Security, SOUPS 2006, Pittsburgh, Pennsylvania, USA, July 12-14, 2006*, volume 149 of *ACM International Conference Proceeding Series*, pages 32–43. ACM, 2006. [x](#), [30](#), [48](#), [49](#), [51](#)

Appendices

A AutoPass Prototype and User Trial

A.1 AutoPass client – code extracts

Key extracts from the AutoPass client software (i.e. JavaScript from the Chrome extension), used in the trials described in [Chapter 7](#), are given below.

```

/*
 * utils.js contains common functions used by AutoPass
 * author: Wanpeng Li
 */

/**
 * generate a strong password for user
 * @param {{string}} mPasswd [a long-term strong password selected by user]
 * @param {{string}} hostname [the host name of the target website]
 * @param {{type}} pwdPolicy [a password policy specified by the website]
 * @param {{type}} digitalObj [a digital file, e.g. a text fragment, picture, or audio sample]
 * @return {{string}} [return a deterministic password for the user]
 */
function generatePassword(mPasswd, hostname, pwdPolicy, userAccountName,digitalObj) {
    // hash the master password
    var mPasswdHash = sha256(mPasswd);
    // hash hash(master password) + hostname
    var outHash = sha256(mPasswdHash + hostname + userAccountName);
    // generate
    var password = mapStringToPRML(pwdPolicy, outHash);
    return password
}

/**
 * generate a password offset for user
 * @param {{string}} mPasswd [a long-term strong password selected by user]
 * @param {{string}} hostname [the host name of the target website]
 * @param {{type}} pwdPolicy [a password policy specified by the website]

```

```

* @param {[type]} digitalObj [a digital file, e.g. a text fragment, picture, or audio sample]
* @return {[string]}      [return a deterministic password for the user]
*/

function generatePasswordOffset(mPasswd, hostname, pwdPolicy,
userAccountName,userPwd,digitalObj){

    var mPasswdHash = sha256(mPasswd);

    // hash hash(master password) + hostname

    var outHash = sha256(mPasswdHash + hostname + userAccountName);

    // generate

    var outNum = bigInt(outHash, 16);

    var userPwdHex = ascii_to_hexa(userPwd);

    var userPwdNum = bigInt(userPwdHex, 16);

    var pwdOffset = userPwdNum.subtract(outNum);

    return pwdOffset.toString(16);

}

/**
* check the login status
* @return {Boolean} true if the user has logged in
*/

function hasLoggedIn() {

    var data = localStorage.getItem("masterPassword");

    if (data){

        return true;

    }else{

        return false;

    }

}

/**
* generate master passwords for the user

```

```

* @return {string} a 128-bit string (16 chars)
*/
function generateMasterkey() {
    return random(16);
}

/**
* change ascii to hexadecimal
* @param {str} str the asicc string
* @return {str} the hex string
*/
function ascii_to_hexa(str)
{
    var arr1 = [];
    for (var n = 0, l = str.length; n < l; n ++)
    {
        var hex = Number(str.charCodeAt(n)).toString(16);
        arr1.push(hex);
    }
    return arr1.join("");
}

/**
* convert hex decimal to ascii
* @param {str} str1 the hex string to convert
* @return {str} the ascii string
*/
function hex_to_ascii(str1)
{

```



```

var hex = str1.toString();
var str = "";
for (var n = 0; n < hex.length; n += 2) {
    str += String.fromCharCode(parseInt(hex.substr(n, 2), 16));
}
return str;
}

/**
 * calculate sha256 of a give string
 * @param {string} str [the input string]
 * @return {string} [the sha256 of the input string]
 */

function sha256(ascii) {
    function rightRotate(value, amount) {
        return (value>>>amount) | (value<<(32 - amount));
    };

    var mathPow = Math.pow;
    var maxWord = mathPow(2, 32);
    var lengthProperty = 'length'
    var i, j; // Used as a counter across the whole file
    var result = ""

    var words = [];
    var asciiBitLength = ascii[lengthProperty]*8;

    /** caching results is optional - remove/add slash from front of this line to toggle
    // Initial hash value: first 32 bits of the fractional parts of the square roots of the first 8 primes
    // (we actually calculate the first 64, but extra values are just ignored)

```



```

for (j = 0; j < words[lengthProperty];) {
  var w = words.slice(j, j += 16); // The message is expanded into 64 words as part of the iteration
  var oldHash = hash;
  // This is now the undefinedworking hash", often labelled as variables a...g
  // (we have to truncate as well, otherwise extra entries at the end accumulate
  hash = hash.slice(0, 8);

  for (i = 0; i < 64; i++) {
    var i2 = i + j;
    // Expand the message into 64 words
    // Used below if
    var w15 = w[i - 15], w2 = w[i - 2];

    // Iterate
    var a = hash[0], e = hash[4];
    var temp1 = hash[7]
      + (rightRotate(e, 6) ^ rightRotate(e, 11) ^ rightRotate(e, 25)) // S1
      + ((e&hash[5])^(~e)&hash[6])) // ch
      + k[i]
    // Expand the message schedule if needed
    + (w[i] = (i < 16) ? w[i] : (
      w[i - 16]
      + (rightRotate(w15, 7) ^ rightRotate(w15, 18) ^ (w15>>>3)) // s0
      + w[i - 7]
      + (rightRotate(w2, 17) ^ rightRotate(w2, 19) ^ (w2>>>10)) // s1
    )|0
    );

    // This is only used once, so *could* be moved below, but it only saves 4 bytes and makes things
    unreadable
    var temp2 = (rightRotate(a, 2) ^ rightRotate(a, 13) ^ rightRotate(a, 22)) // S0
      + ((a&hash[1])^(a&hash[2])^(hash[1]&hash[2])); // maj

```

```
    hash = [(temp1 + temp2)|0].concat(hash); // We don't bother trimming off the extra ones,
they're harmless as long as we're truncating when we do the slice()
```

```
    hash[4] = (hash[4] + temp1)|0;
}
```

```
for (i = 0; i < 8; i++) {
    hash[i] = (hash[i] + oldHash[i])|0;
}
}
```

```
for (i = 0; i < 8; i++) {
    for (j = 3; j + 1; j--) {
        var b = (hash[i]>>(j*8))&255;
        result += ((b < 16) ? 0 : '') + b.toString(16);
    }
}
return result;
};
```

```
/**
```

```
 * randomly generate a string
```

```
 * @param {int} howMany the length of the output string
```

```
 * @param {string} chars the chars used to generate the random string
```

```
 * @return {string} the random string
```

```
 */
```

```
function random(howMany, chars) {
```

```
    var chars = chars
```

```
        || "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_=@+";
```

```
    var numbers = new Uint8Array(howMany);
```

```

var rnd = crypto.getRandomValues(numbers);
var value = new Array(howMany);
var len = chars.length;
for (var i = 0; i < howMany; i++) {
    value[i] = chars[rnd[i] % len]
};
return value.join("");
}

```

```

// chrome.webRequest.onBeforeSendHeaders.addListener(
//     function(details) {
//         for (var i = 0; i < details.requestHeaders.length; ++i) {
//             if (details.requestHeaders[i].name === '11') {
//                 details.requestHeaders.splice(i, 1);
//                 break;
//             }
//         }
//         console.log(details.requestHeaders);
//         return {requestHeaders: details.requestHeaders};
//     },
//     {urls: ["<all_urls>"]},
//     ["blocking", "requestHeaders"]);

```

```

/**
 * retrieve PPD of a website
 * @param {string} domain the domain name of the target website
 * @param {integer} count the number of search results to be retrieved
 * @return {xml} an xml file
 */

```

```

function cashPPD(domain, count) {
    // search for PPD
    var xmlhttp = new XMLHttpRequest();
    var searchURL = "https://api.ppdds.passwordassistance.info/v1/search"
    xmlhttp.open("POST", searchURL);
    xmlhttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
    xmlhttp.setRequestHeader("Accept", "application/json;charset=UTF-8");

    xmlhttp.onload = function () {
        var result = JSON.parse(xmlhttp.responseText);
        console.log(result);
        var results = result.result;
        // retrieve PPD file
        var foundPPD = false;
        if (results && results.length > 0) {
            for (var i = 0; i < results.length; i++) {
                var rawURL = results[i].url;
                var url = new URL(rawURL);
                var domainRetrieved = extractDomain(url.host);
                // console.log(domainRetrieved)

                // make sure domain names of retrieved PPD same as the search domain
                if (domainRetrieved === domain){
                    foundPPD = true;
                    var handler = results[i].ppdHandle;
                    var baseURL = "https://api.ppdds.passwordassistance.info/v1/ppds/";
                    var xml = new XMLHttpRequest();
                    xml.open("GET", baseURL + handler);
                    xml.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
                    xml.setRequestHeader("Accept", "application/json;charset=UTF-8");
                    // console.log("I am here")
                }
            }
        }
    }
}

```

```

xml.onload = function(){
    var res = JSON.parse(xml.responseText);
    // console.log(res);
    storePPD(domain, res);
    console.log("Store PPD for domain: " + domain);
};
xml.send();
}
}
if (!foundPPD) {
    console.log("No PPD is found on domain: " + domain);
}
}else{
    console.log("No PPD is found on domain: " + domain);
}

};
xmlhttp.send(JSON.stringify({ query: domain, count: count}));
}

/**
 * store ppd to local storage
 * @param {string} domain the domain name of the website
 * @param {json} ppd a pdd json file
 * @return {none}
 */
function storePPD(domain, ppd) {
    var pddStore = localStorage.getItem(domain);
    if (pddStore) {
        console.log(domain + " pdd has been cashed earlier!");
    } else {

```

```

    localStorage.setItem(domain, JSON.stringify(ppd));
  }
}

/*
 * Domain name extractor. Turns host names into domain names
 * Adapted from Chris Zarate's public domain genpass tool:
 * http://labs.zarate.org/passwd/
 */

function extractDomain(host) {
  var s; // the final result
  // Begin Chris Zarate's code
  var host=host.split('.');

  if(host[2]!=null) {
    s=host[host.length-2]+'.'+host[host.length-1];

domains='ab.ca|ac.ac|ac.at|ac.be|ac.cn|ac.il|ac.in|ac.jp|ac.kr|ac.nz|ac.th|ac.uk|ac.za|adm.br|adv.br|agro.pl|ah.cn|aid.pl|alt.za|am.br|arq.br|art.br|arts.ro|asn.au|asso.fr|asso.mc|atm.pl|auto.pl|bbs.tr|bc.ca|bio.br|biz.pl|bj.cn|br.com|cn.com|cng.br|cnt.br|co.ac|co.at|co.il|co.in|co.jp|co.kr|co.nz|co.th|co.uk|co.za|com.au|com.br|com.cn|com.ec|com.fr|com.hk|com.mm|com.mx|com.ph|com.pl|com.ro|com.ru|com.sg|com.tr|com.tw|cq.cn|cri.nz|de.com|ecn.br|edu.au|edu.cn|edu.hk|edu.mm|edu.mx|edu.pl|edu.tr|edu.za|eng.br|ernet.in|esp.br|etc.br|eti.br|eu.com|eu.lv|fin.ec|firm.ro|fm.br|fot.br|fst.br|g12.br|gb.com|gb.net|gd.cn|gen.nz|gmina.pl|go.jp|go.kr|go.th|gob.mx|gov.br|gov.cn|gov.ec|gov.il|gov.in|gov.mm|gov.mx|gov.sg|gov.tr|gov.za|govt.nz|gs.cn|gsm.pl|gv.ac|gv.at|gx.cn|gz.cn|hb.cn|he.cn|hi.cn|hk.cn|hl.cn|hn.cn|hu.com|idv.tw|ind.br|inf.br|info.pl|info.ro|iwi.nz|jl.cn|jor.br|jpn.com|js.cn|k12.il|k12.tr|lel.br|ln.cn|ltd.uk|mail.pl|maori.nz|mb.ca|me.uk|med.br|med.ec|media.pl|mi.th|miasta.pl|mil.br|mil.ec|mil.nz|mil.pl|mil.tr|mil.za|mo.cn|muni.il|nb.ca|ne.jp|ne.kr|net.au|net.br|net.cn|net.ec|net.hk|net.il|net.in|net.mm|net.mx|net.nz|net.pl|net.ru|net.sg|net.th|net.tr|net.tw|net.za|nf.ca|ngo.za|nm.cn|nm.kr|no.com|nom.br|nom.pl|nom.ro|nom.za|ns.ca|nt.ca|nt.ro|ntr.br|nx.cn|odo.br|on.ca|or.ac|or.at|or.jp|or.kr|or.th|org.au|org.br|org.cn|org.ec|org.hk|org.il|org.mm|org.mx|org.nz|org.pl|org.ro|org.ru|org.sg|org.tr|org.tw|org.uk|org.za|pc.pl|pe.ca|plc.uk|ppg.br|presse.fr|priv.pl|pro.br|psc.br|psi.br|qc.ca|qc.com|qh.cn|re.kr|realestate.pl|rec.br|rec.ro|rel.pl|res.in|ru.com|sa.com|sc.cn|school.nz|school.za|se.com|se.net|sh.cn|shop.pl|sk.ca|sklep.pl|slg.br|sn.cn|sos.pl|store.ro|targi.pl|tj.cn|tm.fr|tm.mc|tm.pl|tm.ro|tm.za|tmp.br|tourism.pl|travel.pl|tur.br|turystyka.pl|tv.br|tw.cn|uk.co|uk.com|uk.net|us.com|uy.com|vet.br|web.za|web.com|www.ro|xj.cn|xz.cn|yk.ca|yn.cn|za.com';

```



```

domains=domains.split('|');
for(var i=0;i<domains.length;i++) {
    if(s==domains[i]) {
        s=host[host.length-3]+'.'+s;
        break;
    }
}
}else{
    s=host.join('.');
}
// End Chris Zarate's code
return s;
}

```

```

/**
 * generate C-ary representation of a string
 * @param {integer} c the integer use to generate the representation
 * @param {string} the hex string e.g. 0-f
 * @return {string} a C-ary representation of the string
 */
function generateCaryString(c,str) {
    var stacks = [];
    var bigNum = bigInt(str, 16);
    // console.log(bigNum);
    while (bigNum.compare(0)){
        var rem = bigNum.mod(c);
        stacks.push(rem);
        bigNum = bigNum.divide(c);
    }
    return stacks.reverse();
}

```

```

}

/**
 * map string representation to characters in PRML
 * @param {JSON} jsonPRML json file of the PRML file of the website
 * @param {integer} the hex string, e.g. 0-f
 * @return {string} a string that match the PRML
 */
function mapStringToPRML(jsonPRML, str) {
  console.log(jsonPRML)
  var characterSet = jsonPRML.ppd[0].characterSets;
  var characters = "";
  // console.log(characterSet)
  // retrieve all characters from PRML
  for (var i = 0; i < characterSet.length; i++) {
    var character = characterSet[i].characters;
    characters += character;
  }

  // build a map between C-ary and strings.
  var dict = {};
  for (var i = 0; i < characters.length; i++) {
    dict[i] = characters[i];
  }
  var c = characters.length;
  var cary = generateCaryString(c, str);
  console.log(cary);
  console.log(dict);
  // transfer C-ary to strings
  var str = "";
  for (var i = 0; i < cary.length; i++) {

```

```

    var c = cary[i];
    str += dict[c.toString()];
}
return str;
}

/**
 * verify a password whether it satisfy the PRML requirements
 * @param {JSON} jsonPRML the PRML object for a domain
 * @param {string} password the generated password
 * @return {password or false}    return true if the generated password is OK
 */
function verifyPassword(jsonPRML, password) {
    var valid = false;
    var tmpPwd = ""
    // retrieve requirements from PRML
    var properties = jsonPRML.ppds[0].properties;
    var maxLength = properties.maxLength;
    var minLength = properties.minLength;
    // check the minimum password length
    if (minLength && password.length < minLength) {
        console.log("Password: " + password + " is not valid, Reason: not enough length")
        return false;
    }
    // check maximum password length and trim generated password
    if (maxLength && password.length > maxLength) {
        console.log("trim Password: " + password + " to meet max length requirement")
        tmpPwd = password.substring(0, maxLength);
        console.log(tmpPwd)
    }
    // check the consecutive occurrence of a character in password

```

```

var maxConsecutive = properties.maxConsecutive;
var char = tmpPwd[0];
var counter = 1;
for (var i = 1; i < tmpPwd.length; i++) {
    var tempChar = tmpPwd[i];
    if (char === tempChar) {
        counter++;
        if (maxConsecutive && counter > maxConsecutive) {
            // console.log(counter);
            console.log("Password: " + tmpPwd + " is not valid, Reason: max consecutive occurrence")
            return false;
        }
    }else{
        counter = 1;
        char = tempChar;
    }
}

// check character set
var characterSet = jsonPRML.ppds[0].characterSets;
for (var i = 0; i < characterSet.length; i++) {
    var charOccursSet = jsonPRML.ppds[0].properties.characterSettings.characterSetSettings;
    var minCharOccurs = charOccursSet[i].minOccurs;
    var maxCharOccurs = charOccursSet[i].maxOccurs;
    if (minCharOccurs) {
        // check min occurs of a charset
        var charSet = new Set(characterSet[i].characters);
        var interSet = occurrence(charSet, tmpPwd);
        var count = interSet.length;
        if (count < minCharOccurs) {
            console.log("Password: " + tmpPwd + " is not valid, Reason: min char occurrence")
        }
    }
}

```

```

    return false;
  }
}

if (maxCharOccurs) {
  // check max occurs of a charset
  var charSet = new Set(characterSet[i].characters);
  var interSet = occurrence(charSet, tmpPwd);
  var count = interSet.length;
  if (count > maxCharOccurs) {
    console.log("Password: " + tmpPwd + " is not valid, Reason: max char occurrence")
    return false;
  }
}

return tmpPwd || password;
}

// count occurrence of a set characters appeared in password
function occurrence(setChar, password) {
  var _intersection = [];
  for (var elem of password) {
    if (setChar.has(elem)) {
      _intersection.push(elem);
    }
  }
  return _intersection;
}

```

A.2 AutoPass Testing Documents

As stated in Chapter 7, participants in the user trial of AutoPass were provided with four documents. These are reproduced in [A.3-A.6](#) below.

A.3 Consent Form

All user trial participants were asked to read and sign the consent form before starting use of AutoPass.

CONSENT FORM

April 1, 2019

Title of project: Testing AutoPass

Name of researcher: Fatma AL Maqbali

I confirm that I have read and understand the information sheet for the above study and have had the opportunity to ask questions. I understand that my participation is voluntary and that I am free to withdraw at any time, without giving any reason. I agree to take part in the above study. I understand that I can withdraw my consent at any time by contacting the researcher.

I understand that for the duration of the project my contribution will be kept safely and securely with access only to those with permission from the researcher. I give my permission for the information I have given to be used for research or educational purposes only (including research publications and reports). At no time will any user password ever be held (short or long term) externally to a user device. Any information included in publications or reports will be anonymised, and no information about individual accounts or passwords will ever be released.

Signature:

Name :

Date:

Thank you for taking part in this study

A.4 Testing AutoPass: A user guide

This document is intended to give an overview of the AutoPass password generator, and to provide the user with guidance on its use for testing purposes. It also sketches how AutoPass operates and indicates precisely what user information is stored and where.

Testing AutoPass: A user guide

Fatma Al Maqbali
Information Security Group, Royal Holloway, University of London
`fatmaa.soh@cas.edu.om`

April 1, 2019

1 Introduction

This document is intended to give an overview of the AutoPass password generator, and to provide the user with guidance on its use for testing purposes. It also sketches how AutoPass operates and indicates precisely what user information is stored and where.

1.1 Background

This experiment is concerned with *password generators*, i.e. schemes designed to simplify password management for end users by generating site-specific passwords on demand from a small set of readily-memorable inputs. Note that the term has also been used to describe much simpler schemes for generating random or pseudorandom passwords which the user is then expected to remember; however, we use the term to describe a system intended to be used whenever a user logs in and that can generate the necessary passwords on demand and in a repeatable way.

1.2 Password Generators

For the purposes of my research, a password generator has the following components.

- A set of *input values* is used to determine the password for a particular site. Some values must be site-specific so that the generated password is site-specific. The values could be stored locally or online, based on the characteristics of the authenticating site, or user-entered when required. In practice, systems can, and often do, combine these types of input.
- A *password generation function* combines the input values to generate an appropriate password. This function could operate in a range of

ways depending on the requirements of the web site performing the authentication. For example, one web site might forbid the inclusion of non-alphanumeric characters in a password, whereas another might insist that a password contains at least one such character. To be broadly applicable, a password generation function must therefore be customisable.

- A *password output method* enables the generated password to be transferred to the authenticating site. This could, for example, involve displaying the generated password to the user, who must then type (or copy and paste) it into the appropriate place.

All this functionality needs to be implemented on the user platform. There are various possibilities for such an implementation, including as a stand-alone application or as a browser plug-in.

1.3 AutoPass

AutoPass is an experimental password generator, developed as part of my PhD research. It incorporates both features from existing schemes as well as a number of novel features. The system design is described in detail in a published paper [1].

Wanpeng Li has kindly developed a prototype implementation of my design, and the purpose of the experiment in which you are being asked to participate is to test the usability and deployability of this prototype of AutoPass as a way of validating my work. The results will form an important part of my PhD thesis.

AutoPass has two main components: the AutoPass server and the AutoPass client software. The AutoPass server is used to store relatively non-sensitive user data, e.g. the user name and website-specific password policies (i.e. specifications of the types of password a particular site will accept). The AutoPass client software is implemented as a browser plug-in; it provides a user interface, and automatically generates site-specific user passwords as a combination of the specified set of inputs. Some of the inputs are stored locally and some are stored in the AutoPass server, with which the client software interacts as necessary. Where possible, the generated password is automatically inserted into login forms.

Figure 1 depicts the AutoPass architecture, showing the main components of the scheme.

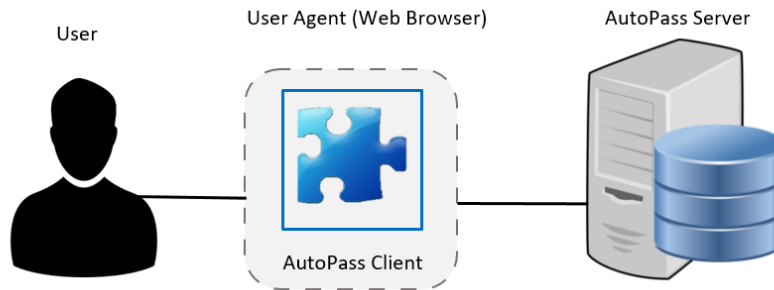


Figure 1: AutoPass System Architecture

2 Data Collection and Storage

As I have described, AutoPass needs access to a variety of configuration data to enable it to operate automatically, and this configuration data clearly needs to be stored somewhere. There are two possible locations for data storage, namely the AutoPass server and the AutoPass client, and AutoPass uses both. The configuration data stored at the server is held long-term, i.e. for the lifetime of the user account at the server; the data held on the client may be held either short-term, typically for the life of a session, or long-term, i.e. while the software remains installed on the client platform. We summarise below the various types of stored data.

2.1 Server-stored data

The following user-specific configuration data is held at the AutoPass server:

- the user account name;
- an email address for the user;
- the (encrypted) master password;
- a hash of the master password;
- a (salted) hash of the session password
- for each website for which a password has been generated for this user:
 - the (first part) of the URL of the website;
 - the types of input used to generate the password for this site;

- the password offset for this site (this relatively non-sensitive information cannot be used to compute the user password without information that is not held on the server)

The following site-specific configuration data is held at the AutoPass server:

- the (first part) of the URL of the website;
- the password policy of the site, encoded in PRML (an XML-based scheme for specifying rules used by a site that contain password choices);

Note that the site-specific configuration data could be maintained by a server separate from that used to store the user-specific configuration data. Indeed, since this data is completely non-confidential, it could be provided by a service independent of AutoPass, e.g. the Password Requirements Description Distribution Service (PRDDS) [2], which provides an online interface to meet requests for PRML-based Password Requirements Descriptors (PRDs) for websites identified by their URL.

2.2 Client-stored data

The following data is held long-term by the AutoPass client:

- cached copies of password policies for recently visited websites.

The following data is held short-term by the AutoPass client:

- the session password;
- a multiply-iterated hash of the master password.

3 Contact details

For questions about the experiment please contact:

Fatma AL Maqbali
Email: pbva132@live.rhul.ac.uk

References

- [1] Fatma Al Maqbali and Chris J Mitchell. Autopass: An automatic password generator. In *Security Technology (ICCST), 2017 International Carnahan Conference on*, pages 1–6. IEEE, 2017. <https://pure.royalholloway.ac.uk/portal/files/28337270/aaapg.pdf>.

- [2] Moritz Horsch, Mario Schlipf, Johannes Braun, and Johannes A. Buchmann. Password requirements markup language. In Joseph K. Liu and Ron Steinfeld, editors, *Information Security and Privacy — 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4–6, 2016, Proceedings, Part I*, volume 9722 of *Lecture Notes in Computer Science*, pages 426–439. Springer-Verlag, 2016.

A.5 AutoPass: An Introductory Guide

This document describes how to install and use AutoPass.

AutoPass: An Introductory Guide

Fatma Al Maqbali
Information Security Group, Royal Holloway, University of London
`fatmaa.soh@cas.edu.om`

April 1, 2019

1 Introduction

This document describes how to install and use AutoPass.

2 What is AutoPass?

AutoPass is a password generator designed to simplify password management for end users by generating website-specific passwords on demand from a small set of inputs. It incorporates features from existing schemes as well as a number of novel techniques. Its operation relies on the installation of a Chrome extension available from the Google webstore.

AutoPass has two main components: the AutoPass server and the AutoPass client software. The AutoPass server is used to store relatively non-sensitive user data, e.g. the user name and website-specific password policies (i.e. specifications of the types of password a particular site will accept). The AutoPass client software is implemented as a Chrome extension; it provides a user interface, and automatically generates site-specific user passwords as a combination of the specified set of inputs. Some of the inputs are stored locally and some are stored in the AutoPass server, with which the client software interacts as necessary.

3 AutoPass Installation

The AutoPass extension icon is shown in Figure 1. The AutoPass extension can be installed by clicking on the following link: <https://chrome.google.com/webstore/detail/autopass/oapnffdchomlagblkehieopjkkjcb/related>

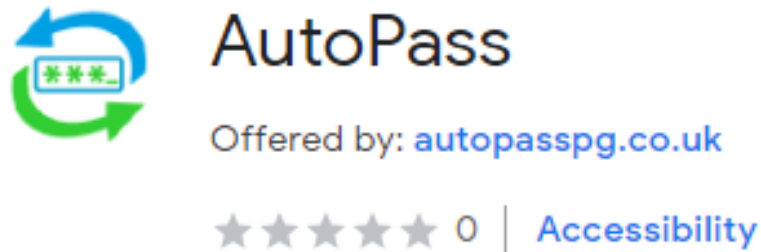


Figure 1: The AutoPass extension icon

After installation, when you next open Chrome, the extension will be displayed next to the search bar.

4 AutoPass Registration

Before using AutoPass to generate passwords, it is necessary to register with the AutoPass server and create an account. This involves the following steps.

1. Use your browser to visit: <http://www.autopasspg.co.uk/login/>
2. Click on the *Sign Up* button. This will lead you to the registration page, shown in Figure 2. The various items of information required to complete the registration process are marked with numbers in figure 2, and each item is now explained.

The screenshot shows a web browser at the URL www.autopasspg.co.uk/register/. The page title is "Create your AutoPass Account". The form contains the following fields and buttons:

- Account name:** Input field with a "Check account availability" button (1).
- First name:** Input field (2).
- Last name:** Input field (3).
- Email:** Input field containing "me@example.com" (4).
- Password:** Input field (5).
- Password confirmation:** Input field (6).
- Master Password:** Input field with a "Generate Master Key" button (7).
- reCAPTCHA:** A box containing "I'm not a robot" (checkbox), a reCAPTCHA logo, and "reCAPTCHA Privacy - Terms" (8).
- Registration:** A button (9).

Below the form, there is a warning: "Please keep a hard copy this master key, cause AutoPass is not able to recover this key if you forget your AutoPass login password." At the bottom left, there is a "Print Registration" button (10).

Figure 2: AutoPass registration page

1. *Account name*: please provide a user name; you can check its availability by pressing on *check account availability*. If the account name you choose is unavailable, i.e. it is already in use, then you will need to provide another user name.
2. *First name*: please provide your first name.
3. *Last name*: please provide your last name.
4. *Email*: please provide an email address for account recovery purposes.
5. *Password*: please provide a login password which must be at least eight characters long — you will need to remember this password, which you will be required to enter every time you start the extension (when you start Chrome).
6. *Password confirmation*: please re-enter the login password you provided in the previous field.
7. *Master Password*: when you click *Generate Master Key*, AutoPass will generate a master password for you, which will be used to generate passwords for individual websites. It is strongly recommended that you keep a hard copy of the master password, because AutoPass is not able to recover it if you forget your AutoPass login password. It is stored encrypted using a key generated from the login password.

8. *Solve CAPTCHA*: please tick the box next to *I am not a robot*, and then solve the supplied CAPTCHA (if appropriate).
9. *Registration*: please press this button to complete the registration process.
10. *Print registration*: please press this button to make a printed copy of the registration credentials (including the master password).

5 Signing In

Once registration is complete, you will need to *sign in* to the extension whenever you start Chrome. To *sign in*, please follow the steps listed below.

1. In Chrome, click on the AutoPass extension and then provide your account name and the password, as shown in Figure 3.

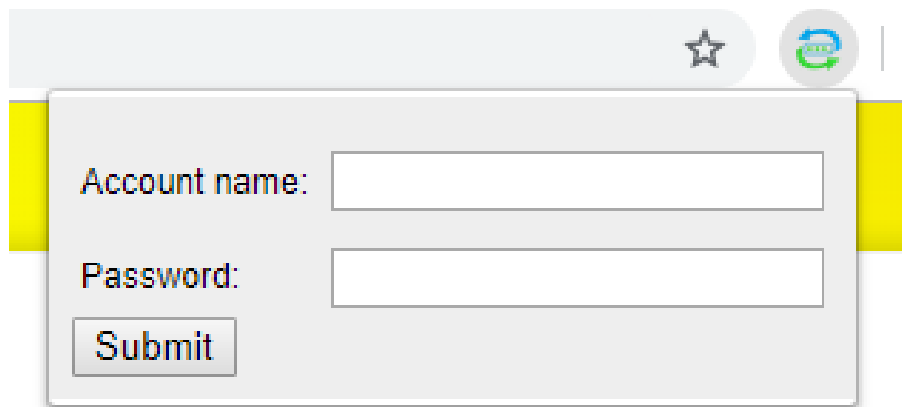


Figure 3: AutoPass login window

2. *Account Name*: provide the user name you registered with AutoPass.
3. *Password*: provide the login password you registered with AutoPass.

After signing in, AutoPass will be available to assist in providing passwords for websites.

6 Using AutoPass

6.1 First time use with a website

When you first navigate to a website (using Chrome), AutoPass will detect the login page and display a pop-up to ask if you wish AutoPass to generate a password for this website — for example as shown in Figure 4.

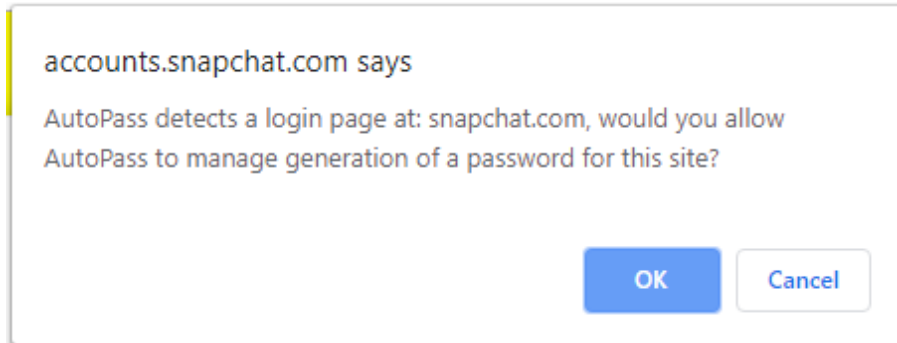


Figure 4: AutoPass pop-up for user consent

If you click on OK, AutoPass will display a pop-up window, as shown in Figure 5. To use AutoPass with this website, please provide the *User name* that you wish to use with this website. AutoPass will automatically fill in the URL for the website.

Hi, Test. Welcome to AutoPass.

Generate a password for a website or

Use your own password:

Username:

URL:

Generated Password:

Figure 5: AutoPass password generation window

You now have two choices.

- If you leave the *User your own password* unchecked, when you click on *Generate Password* AutoPass will generate a new password for this site. This new password will displayed in the *Password Generated* field. Clicking on the *Copy Generated Password* button will cause the generated password to be copied to the clipboard, enabling you to paste it in the appropriate place in the website account set up page.
- If you have already established a password with this website, and you wish to keep using it, then select the *Use your own password* option. Figure 6 shows the window that results if you select this option. In this case please enter your chosen password in the *Your Own Password* field, and click on *Confirm*.

Hi, Test. Welcome to AutoPass.

Generate a password for a website or

Use your own password:

Username:

URL:

Your own password:

Allow AutoPass to manage your password

Figure 6: Use your own password window

Finally note that the *Log Out* button allows you to log out of AutoPass.

6.2 Everyday use

AutoPass will automatically fill in a previously established user name and password if a login page detected and AutoPass has already been set up for this website.

6.3 Notes on use

If you choose to use AutoPass to generate a new password for a site, and you wish to keep using this site after you have stopped using AutoPass, then you need to keep a record of the password AutoPass generates.

Some websites may give the error message *No password policy is found on domain: ****. In such a case AutoPass is unable to generate a password for this site.

7 Contact details

If you have any questions about the operation and security properties of AutoPass, please contact Fatma Al Maqbali at the following email address:

pbva132@live.rhul.ac.uk

A.6 AutoPass evaluation questionnaire

This survey is intended to evaluate the usability and deployability of AutoPass.

AutoPass evaluation questionnaire

This survey is intended to evaluate the usability and deployability of AutoPass.

About you

1. Gender: Male Female

2. Age:

20-29 30-39 40-49 50-59 60-69

Your use of passwords

3. What difficulties do you face with using passwords? [Please select all that apply]

- Too many accounts with too many passwords.
 - Different websites have different policies on what is permitted as a password e.g. upper case, lower case.
 - Some websites require frequent password changes.
 - Other issues: _____
-

4. How do you currently generate your passwords? [Please select all that apply]

- Password generator.
 - Password generator which is built into a password manager or browser.
 - I use the same password for all accounts with minimal modification.
 - I use familiar phrases or dates.
 - I generate a random password from whatever comes to mind.
 - Other methods: _____
-

5. Have you ever used a computer program to generate your passwords?

- Yes
- No

AutoPass Usability

6. Overall, how easy did you find it to use AutoPass?

- Very easy Easy Neutral Difficult Very difficult

7. How easy was it to create an account with AutoPass?

- Very easy Easy Neutral Difficult Very difficult

8. Knowing that AutoPass can generate the correct strong password on demand without the need to store it made you feel:

- More secure
- Less secure
- No different

9. Level of satisfaction with passwords generated using AutoPass

- Satisfied
- Not at all satisfied

Neither satisfied nor unsatisfied

10. Do you prefer AutoPass over your current password management method?

Yes

No

11. Overall, what additional features do you think might improve the functionality of AutoPass?

Answer: _____

A.7 AutoPass Survey Outcome

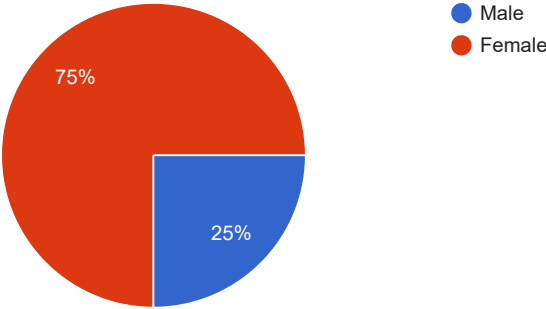
The complete set of questionnaire responses obtained from the user trial participants is given below.

AutoPass evaluation questionnaire

12 responses

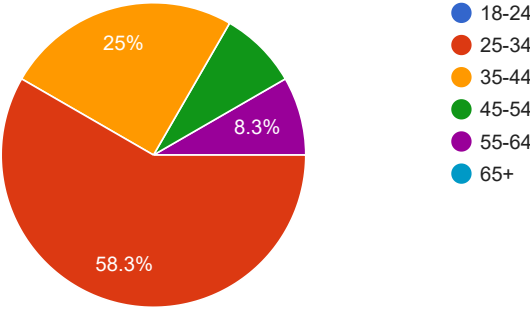
Gender?

12 responses



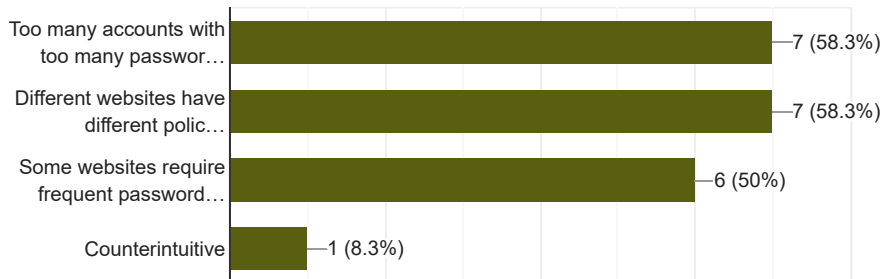
Age?

12 responses



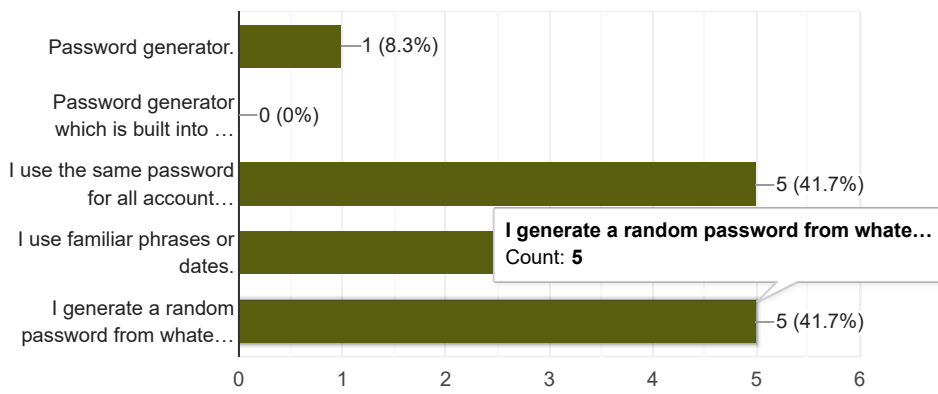
What difficulties do you face with using passwords? [Please select all that apply]

12 responses



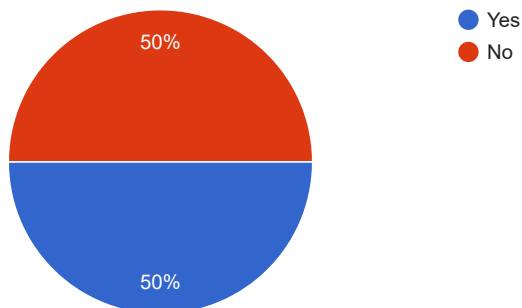
How do you currently generate your passwords? [Please select all that apply]

12 responses



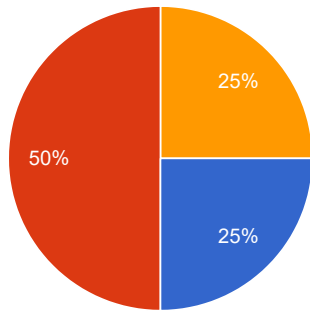
Have you ever used a computer program to generate your passwords?

12 responses



Overall, how easy did you find it to use AutoPass?

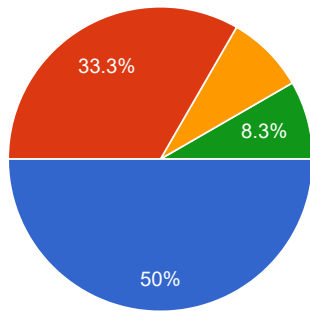
12 responses



- Very Easy
- Easy
- Neutral
- Difficult
- Very Difficult

How easy was it to create an account with AutoPass?

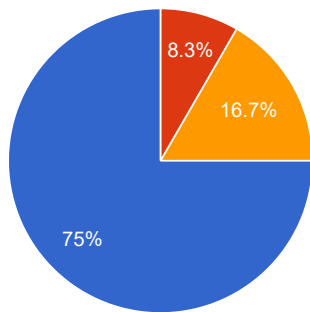
12 responses



- Very Easy
- Easy
- Neutral
- Difficult
- Very Difficult

Level of satisfaction with passwords generated using AutoPass

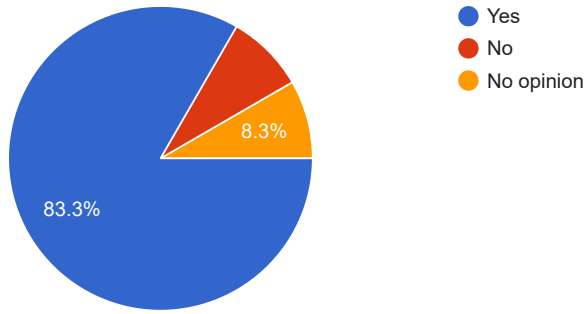
12 responses



- Satisfied
- Not at all satisfied
- Neither satisfied nor unsatisfied

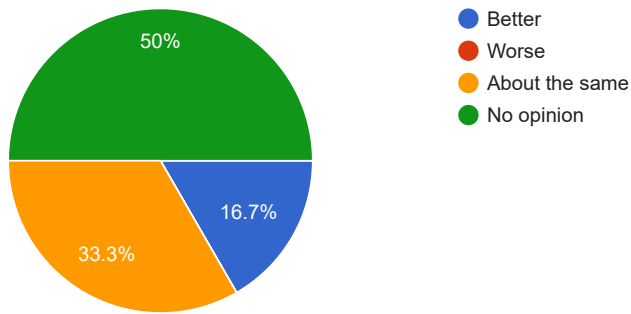
Did you find the ability to use existing passwords useful?

12 responses



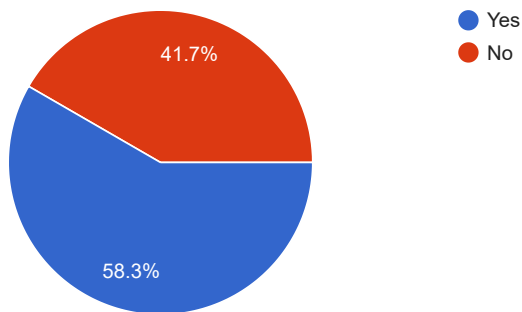
If you ever used a computer program to generate your passwords, how does AutoPass compare with methods you have used previously?

12 responses



Do you prefer AutoPass over your current password management method?

12 responses



What additional features do you think might improve the functionality of AutoPass?

6 responses

Change the random password

to auto fill the username as well, not only the URL. Great work!

linking password creation with human memorization theories So the program doesn't generate password only but helps user to remember it.

There should be specifications regarding creating passwords in AutoPass in order for the password to be solid

length of the password should be limited

A timer to change your password, to improve security of my password maybe?

This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#)

Google Forms

B Password Recovery Emails – Disclosure and Responses

As described in Chapter 9, details of the study of password recovery emails were sent to all 50 of the websites included in the study. The email sent to the 50 websites is given in B.1 and examples of the responses are provided in B.2-B.7.

B.1 Email Sent to the surveyed websites

Given below is the email sent to the surveyed websites.

Subject: Your website was surveyed to evaluate the security of email based password recovery.

Dear (webservice) team,

I am currently a PhD student at Royal Holloway, University of London working under the supervision of Professor Chris Mitchell. I am performing research on the security of password management.

As part of my research, I have conducted a survey of 50 major websites to analyse and evaluate email-based password recovery, of which your website was one. Every password recovery email we examined possessed at least two issues which could compromise the security of the password recovery process.

I have also formulated a set of recommendations to mitigate the identified risks. A paper describing the results of the study is attached for your reference.

<https://pure.royalholloway.ac.uk/portal/files/30955357/ebprro.pdf>

Regards,

Fatma AL Maqbali

Information Security Group

Royal Holloway

University of London

B.2 vimeo.com

Jason R. (Vimeo Trust & Safety)

Apr 29, 12:24 PM EDT

Hi Fatma,

My name is Jason and I work on Vimeo's Trust and Safety team.

Thanks for sending along your research into account recovery processes. This is something we deal with very frequently and are always looking for ways to improve. We will certainly review your findings and pass along any suggestions to our security and product teams.

Best of luck with your research! Thanks for helping us to keep Vimeo secure.

Sincerely,

Jason R.

Trust & Safety Analyst

Figure 1: Response from vimeo.com

B.3 wikipedia.org

Re: Your website was surveyed to evaluate the security of email based password recovery.

nwilson@wikimedia.org

on behalf of

Wikimedia Answers <answers@wikimedia.org>

Thu 25/04/2019 09:39

To: Al Moqball, Fatma (2015) <Fatma.AIMoqball.2015@live.rhul.ac.uk>

Hi Fatma

Thank you for your work on this important topic, and for letting us know. I've passed this along to the security team for their perusal.

Best wishes,

Nick Wilson

Wikimedia Foundation

Figure 2: Response from wikipedia.org

B.4 Wix.com



Jenee (Wix Support)

April 30, 2019 1:57 PM (UTC-04:00)

Hi,

Thank you for contacting us about sending a security vulnerability report. Please send us details on **how to recreate the security vulnerability**, and **how it affects Wix or our customers**.

The details you provide will help us determine the severity of the issue and if this issue is eligible for a security bug bounty.

If it seems within scope, we will invite you to our HackerOne program for additional investigation into the issue.

Looking forward to your reply.

Warm Regards,
Jenee W. - Product Specialist

Figure 3: Response from Wix.com

B.5 cnet.com



Dear Fatma,

Thank you for contacting CNET Customer Support! Your feedback is an important factor in providing the best customer experience possible.

Your comments have been forwarded to our editorial staff. To be sure that your thoughts are heard, please feel free to reach out to them directly with the following link:

- <https://www.cnet.com/contact/>

We are constantly looking to hear your feedback and suggestions on how we can elevate the CNET experience.

If further questions or concerns shall arise, please feel free to contact us again or simply reply to this email.

Sincerely,

Tim

CNET Customer Support

Figure 4: Response from cnet.com

B.6 dailymail.co.uk

FW: [CONTACT] Technical Support Inbox x

DM Mailonline Technical <technical@mailonline.co.uk>
to me ▾

Thu, May 9, 2:56 PM (18 hours ago)

Dear Fatma,

Many thanks for your email.

We just read your research and found some areas where we could improve our password recovery emails.

Best Regards,
Support team

Figure 5: Response from dailymail.co.uk

B.7 IMDb.com

Your IMDb Inquiry Inbox x



IMDb.com <support-imdb@imdb.com>
to me ▾

May 13, 2019, 8:27 PM (13 hours ago) ★ ↩



[Help](#) | [IMDb.com](#)

Message from Customer Service

Hi Fatma,

Thank you for contacting us to report this issue. We appreciate you working with us to better protect our customers.

Please review our Help Page for further details : <https://help.imdb.com/article/imdb/general-information/how-to-report-security-is-..>

They will email you back and contact you to further investigate the issue.

Best regards,
Taylor
IMDb Customer Service

Figure 6: Response from IMDb.com

C Other Issues Identified in Email-based Password Recovery

Tables 1 and 2 below supplement Tables 9.2 and 9.3 of Chapter 9. They summarise other issues identified in the 50 surveyed password recovery emails.

Table 1: Other issues identified in email-based password recovery (Part 1)

Number	Website name	Password reset mechanism	Personalised greeting (name, email)	No replies (in email)	Request initiation(when)	Email spoofing	Email address does not relate to password recovery	Email signature absent
1	google.com	email /link	•	•	•		•	•
2	facebook.com	email /link			•			•
3	twitter.com	email /link			•			•
4	microsoft.com	email/temporary code		•	•			
5	linkedin.com	email /link		•	•			•
6	instagram.com	email/link			•			•
7	adobe.com	email /link			•		•	•
8	en.wikipedia.org	email/temporary code			•	•	•	
9	itunes.apple.com	email/ link			•			•
10	vimeo.com	email/link	•		•		•	•
11	pinterest.com	email/link	•		•			•
12	yahoo.com	email/code		•			•	•
13	amazon.com	email/code	•		•			
14	tumblr.com	email/link	•	•	•		•	•
15	github.com	email/link	•	•	•		•	
16	mozilla.org	email/link	•					•
17	sourceforge.net	email/link	•	•	•		•	
18	nytimes.com	email/link	•		•		•	•
19	soundcloud.com	email/link			•		•	•
20	bbc.co.uk	email/link	•		•			•
21	reddit.com	email/link			•		•	
22	weebly.com	email/link	•	•	•		•	
23	dropbox.com	email/link		•	•		•	
24	theguardian.com	email/link			•			
25	forbes.com	email/link			•		•	•
26	creativecommons.org	email/link	•		•		•	
27	issuu.com	email/link		•	•		•	•
28	wix.com	email/link	•	•	•		•	•
29	oracle.com	email/link			•			•
30	imdb.com	email/link	•	•	•		•	
31	slideshare.net	email/link		•	•		•	•
32	paypal.com	email/code			•			•
33	go.com	email/link			•			•
34	myspace.com	email/link			•		•	
35	archive.org	email/link		•	•		•	

Table 2: Other issues identified in email-based password recovery (Part 2)

Number	Website name	Password reset mechanism	Personalised greeting (name, email)	No replies (in email)	Request initiation(when)	Email spoofing	Email address does not relate to password recovery	Email signature absent
36	www.ncbi.nlm.nih.gov	email/link		•	•		•	
37	washingtonpost.com	email/link	•		•		•	•
38	cpanel.net	email/link			•		•	
39	bloomberg.com	email/link	•	•	•		•	•
40	ebay.com	email/link or code			•		•	•
41	telegraph.co.uk	email/link	•		•		•	•
42	ibm.com	email/link	•	•	•	•	•	
43	hp.com	email/link		•	•		•	
44	cnet.com	email/link	•	•				•
45	dailymail.co.uk	email/link	•		•			•
46	opera.com	email/link	•	•	•		•	
47	imgur.com	email/link	•		•			
48	debian.org	email/link			•		•	
49	twitch.tv	email/link		•	•			
50	surveymonkey.com	email/link			•		•	•