Security Ceremonies

Including Humans in Cryptographic Protocols

by

Kenneth J. Radke

Bachelor of Engineering (Electrical) (Queensland University of Technology) – 2000
Bachelor of Information Technology (Queensland University of Technology) – 2000
Graduate Certificate in Engineering (University of Technology Sydney) – 2002
Masters of Engineering Management (University of Technology Sydney) – 2003
Graduate Certificate in Information Technology (Queensland University of Technology) – 2009

Thesis submitted in accordance with the regulations for the Degree of Doctor of Philosophy

Queensland University of Technology

2013
Keywords

Security ceremonies; TLS; HTTPS; provable security; protocols; human encryption; secure designs; trust; freshness; CHURNs; and POPS+. 
Abstract

In 1993, Bellare and Rogaway applied the mathematical reductionist proof technique to cryptographic authentication protocols, at a level which was more practice-focused than previous theory-focused cryptographic reductionist proofs. These proofs are powerful. The reductionist proofs mean that as long as the problem that a protocol reduces to remains hard, particularly “impossibly hard”, then the protocol remains secure. That means that any technique found to break the protocol can be used to solve the previously thought to be impossibly hard problem. Thus, security should be assured (until a way to solve the impossibly hard problem is found). However, in reality reductionist security proofs have failed for protocols which involve humans. This is typically because the approximation of the real-world problem, the model, used to construct the security proof in, does not capture enough real world considerations. The security proof only holds for what is covered by the model. In particular, when considering protocols which involve humans, is the question of how to model human behaviour.

We focus on protocols which include humans, which have previously been labelled security ceremonies. We research human usage of protocols such as HTTPS. We expand the understanding of what a security ceremony is and what ceremony analysis allows. From this base of understanding, we go on to develop three main tools which will aid cryptographers to create security proofs for protocols which involve humans. The three tools are: we present a generic model for capturing the human recognition capability, since a human being able to recognise an object seems critical to any protocol where an entity authenticates itself to the human; we provide a tool for generating human-assisted random values which are recognisable to the human who created them called Computer-HUman Recognisable Nonces (CHURNs); and we provide a model for capturing the concept of human perceptible freshness, thus allowing cryptographers to create protocols resilient to replay attacks for the human.
# Contents

**Front Matter**
- Keywords ........................................... i
- Abstract .......................................... iii
- Table of Contents ................................... v
- List of Figures ..................................... xi
- List of Tables ...................................... xiii
- Declaration ......................................... xv
- Previously Published Material ..................... xvii
- Acknowledgements ................................... xix

**1 Introduction**
1.1 Motivation: Provable Security that Stands-Up to Human Usage . 3
  1.1.1 Secure Protocols ................................. 3
  1.1.2 Provable Security ................................. 3
  1.1.3 Modelling Humans ................................. 4
1.2 Contribution and Outline ............................. 4

**2 Background**
2.1 Security Ceremonies ................................. 8
2.2 Provable Security in Protocols Involving Humans .............. 9
  2.2.1 Provable Security in the Presence of Human Factors ....... 10
  2.2.2 Human Perceptible Authenticators .................. 11
  2.2.3 HTTPS Usage .................................. 12
2.3 HCI and Sociotechnical Considerations .................. 13
2.4 The Need for Freshness in Cryptographic Protocols ........... 14
2.5 Notation and Nomenclature ........................... 17
3 Learning about Human Users

3.1 Introduction .................................................. 20
3.2 Background ................................................. 21
  3.2.1 Trust ..................................................... 22
  3.2.2 Privacy ............................................... 24
  3.2.3 HTTPS and Extended Validation Certificates .......... 24
3.3 Methodology .................................................. 24
  3.3.1 Participants ............................................ 25
  3.3.2 Procedure ............................................. 25
    Web Logs / Diaries ................................... 25
    In-depth Contextual Interviews ....................... 26
  3.3.3 Analytic Method ....................................... 27
  3.3.4 Ethics ............................................... 27
3.4 Findings .................................................... 27
  3.4.1 Identified Problem: Users Need to Protect Themselves on
    the Internet ........................................... 28
    Lack of Delineation Leading to Obscured Security ..... 28
    Web Concerns Continuum ............................... 29
    Users are Responsible for Computer Security .......... 31
  3.4.2 Users’ Actions and Motivation .......................... 31
    Individual Mitigation of the Absence of Rules ........ 31
    What Users Wish to Protect ............................. 34
    Personal Levels of Security ............................ 35
3.5 Discussion and Impact to Web-Based HCI Design .......... 36
3.6 Limitations .................................................. 39
3.7 Summary .................................................... 39

4 Security Ceremonies ............................................. 41
4.1 Introduction .................................................. 41
4.2 Analysis of Flawed Ceremonies .............................. 42
  4.2.1 Ceremonies Example: HTTPS with MITM Attack .......... 43
  4.2.2 EMV Smart Card Security Ceremony .................. 45
  4.2.3 Opera Mini Ceremony ................................. 48
    Opera Mini Design ................................... 48
  4.2.4 Opera Mini Ceremony Analysis ....................... 49
  4.2.5 Lessons Learnt ....................................... 53
4.3 Investigation of a Provably Secure Protocol . . . . . . . . . . . . . . 55
   HCI Issues . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
   Cryptographic Issues . . . . . . . . . . . . . . . . . . . . . 58
4.4 Application to Future Protocol Design and Security Proofs . . . . 58
   4.4.1 Rushing User Resistance . . . . . . . . . . . . . . . . . . . 59
   4.4.2 Security Proofs at a Level Above the Cryptographic Level:
       $POPS+$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
4.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61

5 Formalising Human Recognition . . . . . . . . . . . . . . . . . . . . . 63
   5.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
   5.2 Security Model for Human-Based Recognition . . . . . . . . . . . 65
       5.2.1 Formalisation . . . . . . . . . . . . . . . . . . . . . . . . 65
       HPA Scheme . . . . . . . . . . . . . . . . . . . . . . . . . . 66
       Human Indistinguishability . . . . . . . . . . . . . . . . . . 67
       Security and Usability . . . . . . . . . . . . . . . . . . . . . 68
       Probabilistic versus Deterministic . . . . . . . . . . . . . . 69
       5.2.2 Security Definition for Human Recognition . . . . . . . . 70
       Upper Bound on $\mathcal{A}$’s Probability of Success . . . . . . 70
       5.2.3 Analysis and Discussion . . . . . . . . . . . . . . . . . . . 72
   5.3 Human-Specific $HPAGen$ . . . . . . . . . . . . . . . . . . . . . . 73
       5.3.1 Gajek et al. Browser Based Mutual Authentication over TLS 73
       5.3.2 Human-based Recognition with Non-human Controlled Au-
           thenticators . . . . . . . . . . . . . . . . . . . . . . . . . 75
   5.4 Non-Human-Specific $HPAGen$ . . . . . . . . . . . . . . . . . . . . 76
       5.4.1 Human-based Recognition with Two Devices . . . . . . . . 77
       5.4.2 Proof of human-assisted device pairing protocol . . . . . . 82
           Adversarial model . . . . . . . . . . . . . . . . . . . . . . . 82
   5.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84

6 CHURNs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
   6.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
       6.1.1 Example Use Case . . . . . . . . . . . . . . . . . . . . . . 87
       6.1.2 CHURN Overview . . . . . . . . . . . . . . . . . . . . . . 87
   6.2 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
       6.2.1 Power, Control and Trust . . . . . . . . . . . . . . . . . . . 88
6.2.2 Randomness, Uncertainty and Test Methods . . . . . . . . 89
  Histogram . . . . . . . . . . . . . . . . . . . . . . . . . . . 90
  Entropy . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
  Compressibility . . . . . . . . . . . . . . . . . . . . . . . . 91
  Hamming Distance . . . . . . . . . . . . . . . . . . . . . . 92
6.3 Developed Program . . . . . . . . . . . . . . . . . . . . . . . . . 93
  6.3.1 Symbol Alphabets Used . . . . . . . . . . . . . . . . . . . 93
  6.3.2 Displaying Choices . . . . . . . . . . . . . . . . . . . . . . 94
  6.3.3 Random Choice Locations . . . . . . . . . . . . . . . . . . 95
  6.3.4 Constraining the User . . . . . . . . . . . . . . . . . . . . . 95
6.4 Test Setup and Hypotheses . . . . . . . . . . . . . . . . . . . . . . 98
  6.4.1 Keyboard Entry . . . . . . . . . . . . . . . . . . . . . . . . 99
  6.4.2 Mouse Entry Unconstrained . . . . . . . . . . . . . . . . . 99
  6.4.3 Mouse Entry Constrained . . . . . . . . . . . . . . . . . . 102
  6.4.4 Questionnaire . . . . . . . . . . . . . . . . . . . . . . . . . 103
  6.4.5 Ethics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 104
6.5 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 104
  6.5.1 Participant Analysis . . . . . . . . . . . . . . . . . . . . . . 104
    Age, Gender, and Qualifications . . . . . . . . . . . . . . . . 104
    Logins per week . . . . . . . . . . . . . . . . . . . . . . . . 104
    Feeling in Control . . . . . . . . . . . . . . . . . . . . . . . 105
  6.5.2 The Human Side to Randomness . . . . . . . . . . . . . . . 106
  6.5.3 Histograms . . . . . . . . . . . . . . . . . . . . . . . . . . . 107
    Single character histograms . . . . . . . . . . . . . . . . . . 107
    N-gram histograms . . . . . . . . . . . . . . . . . . . . . . . 109
  6.5.4 Compression Tools . . . . . . . . . . . . . . . . . . . . . . . 109
  6.5.5 Hamming Distance . . . . . . . . . . . . . . . . . . . . . . . 112
  6.5.6 “Who Decides?” . . . . . . . . . . . . . . . . . . . . . . . . 114
6.6 Limitations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 120
6.7 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 121

7 Human Perceptible Freshness . . . . . . . . . . . . . . . . . . . . . . 123
  7.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123
  7.2 Human Perceptible Freshness (HPF) . . . . . . . . . . . . . . . . 127
  7.3 Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129
    7.3.1 Existing Work . . . . . . . . . . . . . . . . . . . . . . . . . 130
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3.2</td>
<td>Updated Model</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td>Attack</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td>H-ACCE Protocol Definition</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td>Updated Model</td>
<td>134</td>
</tr>
<tr>
<td></td>
<td>H-ACCE Security</td>
<td>135</td>
</tr>
<tr>
<td>7.4</td>
<td>Compiler</td>
<td>135</td>
</tr>
<tr>
<td>7.5</td>
<td>Human-Perceptible Freshness for H-ACCE</td>
<td>138</td>
</tr>
<tr>
<td>7.5.1</td>
<td>Session Identifiers</td>
<td>138</td>
</tr>
<tr>
<td>7.5.2</td>
<td>Human Perceptible Freshness for a H-ACCE protocol H-ACCE-HPF</td>
<td>139</td>
</tr>
<tr>
<td>7.6</td>
<td>Summary</td>
<td>140</td>
</tr>
</tbody>
</table>

8 Conclusion and Future Work  

8.1 Conclusion  
8.1.1 Enhanced Understanding of Security Ceremonies  
8.1.2 Formalising Human Recognition  
8.1.3 CHURNS  
8.1.4 Human-Perceptible Freshness  

8.2 Future Work  
8.2.1 CHURNs  
8.2.2 Formalising Human Recognition  
8.2.3 Human-Followable Security  

Bibliography
# List of Figures

2.1 1978 Protocol by Needham and Schroeder (as shown by Boyd and Mathuria [13]). ........................................ 15
2.2 Protocol from Figure 2.1 with Random Nonces for both Parties (as shown by Boyd and Mathuria [13]). ................. 16
3.1 Example web log (fonts adjusted for readability). .............. 26
3.2 Findings overview. .................................................. 38
4.1 HTTPS Ceremony, of Ellison [41]. ................................. 44
4.2 Murdoch et al.’s EMV Ceremony [83]. ............................ 46
4.3 Opera Mini Secure Connection - viewing NAB’s secure logon page. 50
4.4 Opera Mini Ceremony. ................................................ 51
5.1 How $W_{H,HPA}$ relates to $HPASpace$ and $HPASpace_{H}$. .............. 68
5.2 How $W'_{H,HPA}$ and $W''_{H,HPA}$ relates to $HPASpace$ and $HPASpace_{H}$. ... 69
5.3 Pasini and Vaudenay’s SAS Protocol [86] .......................... 77
5.4 Prasad and Saxena’s human-assisted device pairing protocol based on SAS protocol [91] ................................. 80
5.5 Prasad and Saxena’s human-assisted device pairing protocol considered using our formalism ............................. 81
6.1 Example twenty sequence constrained output ..................... 97
6.2 Participant keyboard input form. .................................. 100
6.3 Unconstrained mouse entry form. .................................. 101
6.4 Constrained mouse entry form. .................................... 103
6.5 Histogram of participant ages. ..................................... 105
6.6 Histogram of logins per week. ..................................... 105
6.7 Histogram of single character usage per participant via a keyboard and CHURN-generator. .............................. 108
6.8 Arithmetic compression of participant’s keyboard, unconstrained mouse, and CHURN generation output. .......................... 110
6.9 gzip compression of participant’s keyboard, unconstrained mouse, and CHURN generation output. .......................... 111
6.10 bzip2 compression of participant’s keyboard, unconstrained mouse, and CHURN generation output. .......................... 111
6.11 Observed Hamming Distances for participant pairs for each of the twenty sequences. ............................................. 115
6.12 Observed Hamming Distances for participant pairs for Sequence 8 versus the ideal p=0.75 n=1225 Binomial Distribution. ... 117
6.13 Observed Hamming Distances for participant pairs for Sequence 8 with matched Binomial Distribution of p=0.6271 and n=1225. 118

7.1 Example HTTPS webpage with a HTTP form viewed using Internet Explorer 10. .................................................. 125
7.2 Compiled H-ACCE protocol based on TLS protocol proven to be a secure ACCE protocol ................................. 136
7.3 Compiled H-ACCE protocol using a CHURN and the TLS protocol 141
# List of Tables

2.1 Notation and Nomenclature for use in thesis. .......................... 17

6.1 Random characters in fixed positions in sequences. .................. 96
6.2 Random characters in the same-across-sequence random positions. 97
6.3 Hamming Distance example from first sequence. ...................... 102
6.4 Binomial distribution of Hamming Distances based on 1225 pairs and probability of 0.75. ..................................................... 113
6.5 Fitted binomial probabilities for human-randomness contribution to each sequence. ......................................................... 119
6.6 Minimum, maximum, and average values for human contribution to randomness. ................................................................. 120
Declaration

The work contained in this thesis has not been previously submitted for a degree or diploma at any higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signed: .................................................. Date: .........................
Previously Published Material

The following papers have been published or presented, and contain material based on the content of this thesis.


Acknowledgements

Any acknowledgement section for a project of significance such as a PhD must begin with the supervisors, mentors and inspirational figures. In my case the two gentlemen who fit into each of those categories are Prof. Colin Boyd and Dr. Juan Gonzalez Nieto.

As I was finishing my Masters degree more than ten years ago, I looked around for the next challenge, the next way to grow myself academically, while I continued to work full-time. It seemed as if the next logical step was a PhD. A person I learnt a lot from in my Masters, Prof. Vernon Ireland, gave me two pieces of advice regarding a PhD. They were:

1. Find a research topic you are passionate about (because you will need this to enable you to finish); and

2. Run your PhD as if it is a project.

Since, at that time, I could not find a topic I was passionate about, I put aside the idea of taking on a PhD. I thought for “forever”.

Working as a consultant for ten years taught me that, at least for me, I could ignore Prof. Ireland’s first piece of advice. As a consultant, we never knew what our next task would be: if it would be challenging and fulfilling; or mundane to the point of being soul-destroying. My time as a consultant taught me that it is far more important to have good people around me, such that I would enjoy the journey no matter what the task was, than to have a topic I was passionate about.

Prof. Ireland’s second piece of advice has been critical to the success of my PhD. I recommend running a PhD as a project to anyone who has project management experience when they start their PhD.

In Prof. Boyd and Dr. Gonzalez Nieto I do not just have “good” people around me; I have great people. The very best. To be clear, for many years after
shelving the idea of a PhD in 2002, I did not contemplate a PhD before meeting and learning from Prof. Boyd and Dr. Gonzalez Nieto. I was not looking to find a topic or to start a PhD when I met them. Seemingly by luck I took a subject on cryptology in 2009 as part of a Graduate Certificate, at which point I met Prof. Boyd and Dr. Gonzalez Nieto and eight months later I started my research for this thesis.

When I met with Prof. Boyd and Dr. Gonzalez Nieto to discuss the possibility of a PhD with them as my supervisors, they had some ideas for potential topics. I read the first topic on the list and said, “I’ll take that one,” to which Prof. Boyd spluttered, “While I agree that sometimes our decisions are fairly arbitrary…” However, I knew I had the right people, and that the topic was far less important. A source of amusement was that even in our fifth meeting, when in my mind my research had begun, they still spoke in terms of “If you pick us as your supervisors…” They seemingly did not realise then, though hopefully they realise now, the high regard I had for them and that I simply would not have considered a PhD without them.

So I thank Prof. Boyd and Dr. Gonzalez Nieto for firstly taking me from “I will never do a PhD” to “This is an ideal situation to do a PhD in.” I thank them for then taking me on as a PhD candidate. I thank them for their supervision and support through the years. If the quality of the work and the company, and the green tea and Paellas, could continue for the next 30 years, my life would be very happy indeed.

There are so many others to thank.

I think it may be impossible to do a PhD without a supportive family. Certainly looking around me at the other PhD candidates I see doing well, each of them is well supported by their family. I thank my family, especially my wife and my immediate family, and my extended family, for their ongoing support. As far as my immediate family goes, one of the reasons why a PhD was attractive was because I could see them a lot more compared with commercial work; but definitely the PhD has resulted in times when I see them less than I would like. The creation of this document has been one of those times.

Certain friends, both new and old, have aided me substantially along the way. Probably the first point is that I’m glad they still remember my name - for some of them it has been years since I have seen them regularly. Then, their capacity to step in and take me out for a meal, or come over and help mow the lawn, was
greatly appreciated.

My thanks to David Long, who each year has called me and asked if I was interested in some extra work. Those short bursts of work, at realistic hourly rates not PhD scholarship rates (which are about 1/8th), have made the financial burden of the PhD in my household far more bearable. His encouragement and friendship are greatly appreciated also.

The Information Security Institute at the Queensland University of Technology was a wonderful place to do a PhD. I remember my first reactions when I started. They were: everyone was driven, and no one was being “carried” (as opposed to the commercial world where for a team of people, frequently someone is coasting and being carried by the team); and . . . “they get my jokes!” I had found Nirvana. This institute has now moved into the Institute for Future Environments, of which we are the section which is Secure and Resilient Infrastructure. I won’t name individual names here. Everyone contributed to me as a person, my knowledge of topics such as cryptography, information security, and human studies, and my PhD. Whether it be the people I had lunch with on the ground floor of Margaret St, at Nudos, or on level 8 P Block; or the people who sat closest to me and that I interact with almost every day; or the people I ran with, or swam with; or the people who came to my seminars and asked questions or made observations; or the crypto reading group that has run for my entire PhD; or the team from the Capture the Flag competitions I’ve enjoyed being part of for three years; or the 60ish distinct people who have been participants in my human-research studies . . . Whether student or Professor, a clear contribution has been made in all cases. Attempting to list such a number of people, many of whom have already moved on, would simply mean I mistakenly leave someone out. To single out two people from the institute whose contribution was not strictly research related, Christine Kincaid and Gleb Sechenov have made my PhD both easier and more enjoyable and I thank them very much for that.

I thank Prof. Margot Brereton and the human-computer interaction and participatory design reading group I was part of for a year and a half. Prof. Brereton’s supervisory contribution to my understanding of human-computer interaction concepts was greatly appreciated. The other members of the reading group are all wonderful researchers who have helped my research and my understanding, and they have all turned up repeatedly to be part of my research. Thank you all.
Prof. Laurie Buys taught me my interview technique, was my first interviewee, and arranged for a training session with a journalist. This was critical in my second human study in which I had in-depth interviews with each of my participants. Now I have a new skill for life, which is greatly appreciated.

Prof. Helen Partridge aided each of my three ethics approvals. It may be only a slight exaggeration to suggest that such is the ethics approval process at the Queensland University of Technology, that I would still be working on my first ethics approval, rather than finalising my thesis, if it wasn’t for Prof. Partridge’s assistance. Since neither of my supervisors had been through ethics approval previously, we were very grateful of Prof. Partridge’s willing assistance.

Dr. James McGree made time on at least five occasions to freely provide his expertise on statistics, which was very much appreciated. Dr. McGree’s assistance was critical in analysing the data for the CHURNs presented in Chapter 6.

During my PhD I’ve been lucky to have made relationships with three universities which I have visited, two of them three times. After each time I have visited these universities I have left them feeling that when we host researchers at our university we don’t do enough for our guests - that is how well these hosts have looked after me, contributed to my research, and created a wonderful life-experience for me. Therefore I thank Dr. Lizzie Coles-Kemp and the people at Royal Holloway, University of London; Prof. Dr. Mark Manulis and the people at the Center for Advanced Security Research Darmstadt; and Prof. Dr. Jörg Schwenk and the people from Ruhr-Universität Bochum.

We have had many visitors to our institute while I have been a PhD student. Some stayed for a matter of days or weeks, others for six months or more. Some have been PhD candidates, others Professors. Most made an impact on my knowledge and research, and several I have reached out to since they returned have home for clarification and advice. It is amazing the impact that a one hour lunch can have, with the right person. I thank each of these visitors.

I thank all the participants in my three human trials. I believe that these studies are what give my research meaning, compared with research for human designs and protocols not based in such trials. As such, the contribution of each of the participants has been critical.

Of particular thanks in the final steps of this journey are firstly the four panellists for my PhD Defence, and secondly to the two external examiners. It is wonderful to have the review and input of experts, and them taking the time
to examine and review a document this size on top of their other commitments is greatly appreciated.

Finally, there is one person who is the only person who I am confident has read all of my work from my PhD. She has read each of the versions of my published papers, my unpublished papers, my milestone documents, my ethics approvals, my grant proposals, and this thesis, providing feedback each time. For this, and many other things, I am eternally grateful. She is, of course, my mother...
For my beautiful wife and children.
Chapter 1

Introduction

There has been an increasing trend to meld information security with the social sciences, as indicated by conferences both in the U.S.A and in Europe such as WEIS (economics), SHB (human behaviour) and SOUPS (usability). This multi-disciplinary approach brings into context the human usage of information security systems. As Shostack and Stewart stated in 2008, “...our approach to information security is flawed” and “the way forward cannot be found solely in mathematics or technology” [111].

The most comprehensive method currently known for ensuring security of a protocol is by using provable security, specifically practice oriented provable security (POPS). In 1993 Bellare and Rogaway introduced POPS by defining a model for reductionist security proofs for cryptographic key exchange protocols [7]. Since this time, many cryptographic protocols have been accompanied by a reductionist security proof.

A reductionist security proof means that the security of the protocol is reduced to a known hard mathematical problem, such that if an adversarial advantage is achieved over the protocol, then there will be some significant advantage over the known hard problem. If a protocol is proven secure using a reductionist proof, as long as the “hard” mathematical problem remains sufficiently hard that it is not solved, the protocol is unbreakable within the defined security model.

Unfortunately, many protocols so proven to be secure in theory, have been found to be insecure in practice, when deployed in the real world. This inequality between the theoretical security and the actual security can be traced back to a
Chapter 1. Introduction

deficiency in the security proof model. The mathematical security models while useful, especially for examining the security of a protocol in isolation, do not take into account the wide range of side channel attacks, social engineering, and interfaces to other protocols and the environment, which occur in the real world.

Jesse Walker coined a term security ceremony which Ellison wrote was a more robust method for examining the security of a protocol than by examining a protocol in isolation [41]. Security ceremonies may be described as protocols in their context of use. For example, the protocol HTTPS [97] provides a connection secure from eavesdroppers between two nodes on a network. However a security ceremony would explicitly also encompass the human user, who is viewing a website on their computer, and who is using HTTPS via their web browser running on their computer to securely connect to another computer on the network.

In this thesis we move the understanding of security ceremonies forwards in three main steps. The first step is that a security ceremony is a protocol in its context of use. The second step is the realisation that adding steps to an underlying protocol simply makes another protocol. A fundamental realisation here is that just as the security of a particular cryptographic building block, such as a hash function, would not be proven via its usage in an arbitrary protocol; the security of the underlying protocol cannot be proven via its usage in the higher level protocol. The third step forward in the understanding of security ceremonies is that protocols involving humans need to be human followable. As a step towards achieving this and being able to prove that protocols have this human-followable quality, we define human-perceptible freshness.

In general we focus on mutual authentication, particularly providing the human with assurance that they are communicating with the party they intend to be communicating with. We blend concepts from the provable security community, network security community, human computer interface (HCI) design community, and sociotechnical community.

To be able to create proofs of security for protocols which include humans, we research human security and privacy concerns [47,92,95], their usage of protocols such as HTTP and HTTPS [92,93], and research the ways that current security ceremonies fail [93,94]. Based on the knowledge gained through this process, we develop three tools which will aid cryptographers create and prove the security of protocols which involve humans. Those tools are a formalisation of human recognition, CHURNs which can be used with humans to guarantee freshness,
and a formalisation of human perceptible freshness which is a critical stepping stone towards human perceptible security.

1.1 Motivation: Provable Security that Stands-Up to Human Usage

There are a number of sub-topics which ideally should be highlighted so that an understanding of our path may be understood. We shall go through a set of such topics here to provide the motivation for this research.

1.1.1 Secure Protocols

The most obvious sub-topic is that security protocols which involve humans need to be secure. When communicating with a bank, a user may expect that the protocols used provide:

Confidentiality No one except for the user and the bank can see the information being passed between the user and the bank;

Integrity The messages between the user and the bank cannot be manipulated without the user and the bank knowing a change has occurred; and

Authentication The user should have confidence that they are communicating with the intended bank, and the bank should have confidence they are communicating with the intended user.

Contrary to user expectations, the security services outlined above are not always provided. Most login protocols involving financial institutions only involve the human authenticating themselves to the bank, and not the bank authenticating itself to the human. Therefore, this process needs to be understood and protocols that more correctly meet user expectations need to be developed.

1.1.2 Provable Security

Having decided that secure protocols need to be developed, the next topic is “How do we know the protocol is secure?” Different researchers and different practitioners use different techniques. One method may be to create a list of all
known attacks, and to examine the protocol to see whether each attack works. While this is useful, of course it is only able to detect previously known attacks.

Today there are two main types of approach to analysis of security protocols. One is the formal methods approach which treats cryptographic primitives in a black-box manner. The approach we have used is the concept of mathematically provable security used by cryptographers. Reducing the protocol down to a known mathematical problem has significant security implications. If the mathematical problem that the protocol reduces to is known to be what is called “hard”, and by this a lay explanation may be “impossible to solve using all the computers on earth in the next billion years”, then suddenly we do not need to have pre-conceived and coded attacks. Because of the reduction from protocol to hard problem, we now know that any successful attack on the protocol can be used to break a problem thought to be impossibly hard.

Both techniques have their advantages and disadvantages. Both have their place. We have used provable security because it gives the greater security assurance.

1.1.3 Modelling Humans

To be able to create a mathematical proof of security, we need to have a model to base the proof on. Since we are focusing on humans and including human users in cryptographic protocols, significant research must take place regarding humans and their usage of protocols. In this we differ from many computer scientist researchers, where they conceive of and create something, and then they do a human trial to see how it performs. We did the human studies first and built our creations using that knowledge. While we still tested our creation in a human trial, see Chapter 6, and further human trials could be conducted, by conducting generic quantitative studies we reduced our need for assumptions and expanded our knowledge-base beyond our own experiences before creating.

1.2 Contribution and Outline

The thesis may be seen to be in two parts. The first part, Chapters 3 and 4, may be seen as researching and expanding the understanding of the problem space. The contributions from these chapters were:
• We contributed a greater understanding of human security and privacy issues in HCI designs [47,95].

• We contributed a greater understanding of how humans use the Internet, what the security decisions they identify are, and how they make their decisions [92].

• We contributed a greater understanding of the methods humans use to protect themselves on the web, and what the users believe are the items they need protected (see Chapter 3).

• We further developed the understanding of what a security ceremony is, and what ceremony analysis can encompass [93].

• We provided a greater understanding of what is required for the creation of a secure human protocol [94].

The second part of this thesis, Chapters 5, 6 and 7, are the contributions we made having explored and expanded the understanding of security ceremonies, human security decision making, and protocols that involve humans. The goal of the thesis was to provide cryptographers with tools that would allow them to make secure protocols which involve humans, without having to re-learn their craft or to take on a second research area. As such, our contributions from these chapters were:

• We contributed a general model based on an expanded understanding of human security issues, for use in a cryptographic security proof for the ability of a human to recognise. Modelling the human recognition capability seems fundamental for any protocol where an entity is to be authenticated to a human (see Chapter 5).

• Based on the knowledge we had gained about human protocol use and flaws in security ceremonies which involve humans, presented in Chapters 3 and 4, we contributed a method, supported by a human trial and analysis of the data collected, for creating a cryptographically sound random challenge and response for a human. We called this creation a Computer-Human Recognisable Nonce (CHURN) (see Chapter 6).

• As our final contribution, we took a significant step towards human-followable security in that we define Human-Perceptible Freshness (HPF). We
provide a model for proving the existence of HPF in a protocol. Finally, we provide a compiler to allow cryptographers to turn a base protocol which has certain security properties into a secure protocol which has HPF (see Chapter 7).

We present what we believe are meaningful expansions of our work and our conclusions in Chapter 8.
Chapter 2

Background

To create a mutual authentication protocol between a human and a computer, which is secure with respect to the common understanding of confidentiality and integrity [60, 79], a number of fields of research need to be examined. An adversary having the capabilities of a computer, and one of the parties in a protocol being a computer, means that lessons learned in the non-computer world between humans cannot be directly applied. For example, an attack in a physical environment (such as a robbery) may need a success rate of, at worst, one in ten to be worthwhile for the perpetrator; whereas in the cyberworld attacks that work one time in a million can be seen as successful [111]. So this suggests cryptography with enough security to withstand a computer attack is required, and yet humans are known to have neither the patience, nor the capacity, to compute the necessarily large numerical values required for modern cryptography. Further, if modern cryptography is used, then the human loses visibility, the process becomes non-transparent, and hence, for the general populace, blind trust is required that the data is secure.

Further, cryptography is no longer required only by nation states, the military, or secret lovers as was the case in the past [81]. Today, the general populace, in developing and first world countries, have huge amounts of data and communications they would like protected, and there are many real-world settings, such as smart phones, RFID tags, and e-commerce, that require protected communication. The ubiquitous usage of cryptography throughout a human’s day means that, even if we could somehow remove the advantage that com-
Computers provide the cryptanalyst over human capabilities, for example by using CAPTCHAs [39,120] or POSHs [27], the amount of encrypting and decrypting required makes anything more than human involvement in the cryptosystem at critical authentication steps unrealistic.

In this background section, with our goal being a secure authentication protocol which is not only usable by humans, but also understandable by humans in such a way that blind trust is not required, we will cover security ceremonies (a term coined by Jesse Walker) [41], and the ideals of provable security. We will examine some human-computer interaction (HCI) design and sociotechnical considerations, and close with the need for freshness in cryptographic protocols.

### 2.1 Security Ceremonies

The concept of a ceremony was developed earlier than 2007 [42], before a paper on ceremony design and analysis by Elison [41]. Although little progress has been made regarding ceremonies since 2007, a number of researchers in different areas have agreed that ceremony analysis is a promising research direction. These research areas include formal methods, network security, and applied cryptography.

In the formal methods’ security community, there has been a call to include parts of ceremony analysis in the formal methods’ analysis of protocols [76]. This work has been further developed in Martina et al’s more recent work in the PKI context [77]. Martina et al used the verification method outlined by Ruksenas et al. [102,103], adapted using Bella’s goal availability principles [4], to address the open question that Ellison posed as to how to model human behaviour. Recently, progress has been made towards a ceremonies threat model for the formal methods’ community [20].

In the network security community, the concept of a ceremony has been used to describe protocols which include humans, and thus to create more robust security ceremonies [64]. Karllof et al. describe a concept of conditioned-safe ceremonies, based on a defence-in-depth approach adapted from the human reliability community. Central to their approach is the use of forcing functions whose property is to prevent a user from proceeding, until a critical step is completed.

In the applied cryptography community, Ellison’s ceremonies have been used as a basis for modelling authentication ceremonies involving humans [14]. In the
authentication ceremony described by Brainard et al., a human who still has their primary authentication details intact, the helper, vouches for another personally known human who has lost their authentication details (the asker). This vouching process, an extra factor in identification of the asker, allows emergency authentication details to be provided.

There is a large body of work on such topics as phishing on the internet, and social engineering in general [35, 57]. This reflects the common understanding that many security decisions are based on trust, such as trust in a brand, rather than the mathematical assurances of a correctly executed protocol. For this reason, ceremony analysis provides a more complete understanding of the issues surrounding the use of a protocol by a human, than protocol analysis alone.

2.2 Provable Security in Protocols Involving Humans

In 1993, Bellare and Rogaway responded to a need to add more rigour to authentication protocol analysis [7]. They applied reduction techniques for proving algorithms\(^1\) to authentication and key distribution protocols. These techniques had been previously used by Goldwasser, Micali, Rivest, Blum and Yao in other cryptographic primitive settings [10, 49, 50, 124]. The critical concept of a reductionist proof of security is that, if an adversary can break the protocol, then the adversary can also break the underlying cryptographic primitive.

Perhaps a more significant contribution of Bellare and Rogaway’s 1993 work was the concept of practice-oriented provable security (POPS). Provable security research prior to this had been based on only theoretical primitives [5]. This mean that, at the time of Bellare and Rogaway’s 1993 papers, provably secure cryptographic primitives tended to be much less efficient than primitives used in practice [7, 8]. Since there was no intersection between provably secure cryptographic primitives and the primitives used in practice, provable security pre-1993 was just theory. With the addition of an idealised model, the random oracle, protocols using the primitives used in practice could have security proofs developed.

Unfortunately, the concept behind POPS has not extended as far as required into protocol design, particularly in the area of protocols which involve humans.

\(^1\)These reductionist proof techniques were collectively called provable security.
A fundamental ideal of POPS is that at no point should a protocol be able to be broken without breaking the underlying cryptographic primitive, and hence the protocol should not be weaker than the underlying primitive. In reality, particularly with respect to humans, this is not the case. Beyond human involvement and potential social attacks, information is leaked concerning otherwise secure protocols via means such as observing computation time and power consumption, collectively known as side-channel attacks. For example, humans have shown themselves to be susceptible to many social engineering attacks, which allow the theoretically secure protocols (which have a reductionist proof) to be broken in practice. Further, humans do not execute a protocol as the protocol designer thought they would. The reasons protocols, proven secure mathematically, are broken when humans use them, can be summarised to the model used for the security proof was insufficient. Such a statement masks a variety of sources of deficiency, some of which include:

- modelling a human is too difficult, and hence humans are either left out of the model and security proof, or else humans are given unrealistic powers such as being expected to follow the protocol 100% correctly, 100% of the time; or they are expected to completely forget previous actions.

- The model, and hence the security proof of the protocol, does not include critical out-of-band (OOB) communication and necessary setup steps prior to the protocol running.

- The protocol definition, and hence the security proof based on the model, does not include the complete design (for one example, see Section 4.3). Most particularly, decisions that affect security, particularly HCI decisions, are left out of the protocol definition and are hence being made by non-security-aware practitioners.

### 2.2.1 Provable Security in the Presence of Human Factors

Two of the main benefits of practice oriented provable security are firstly, the focus on cryptographic primitives actually being used; and secondly, the provision of a concrete level of security defined around security parameters usable by the implementers of protocols. For example, the probability of success of an adversary may depend on the number of hash function calls made [7, 8].
Two promising directions in the provable security community have been made by Hopper and Blum [59], and by Gajek et al. [44]. Hopper and Blum’s contribution was to provide a goal of creating \((\alpha, \beta, t)\) protocols for use by humans, in which at least \((1 - \alpha)\) of the human population can do what they need to do, in at most \(t\) seconds, with probability of correct execution of the protocol greater than \((1 - \beta)\). This data could be collected empirically, and their idea was to create light-weight cryptographic protocols that would have a mathematical proof of security, with ideally 90% of the population executing the protocol correctly inside 10 seconds, 90% of the time [59]. Unfortunately, the protocol they suggested resulted in 10% of the population executing the protocol correctly inside 300 seconds, 80% of the time, and has gone on to become the basis of light-weight protocols for constrained devices, such as RFID, rather than human executable protocols [18, 53, 63]. However, the concept of combining empirical evidence of usability with a security proof is a promising direction.

### 2.2.2 Human Perceptible Authenticators

Gajek et al. presented a protocol for mutual authentication between a human and an online entity, via the web [44]. This work was extended, with a different proof, in the journal version [45]. This protocol, discussed in depth in Sections 4.3 and 5.3.1 and the basis of our proposal, has a number of innovative and useful features. Firstly, for the purposes of the security proof, the human is separated from their computer and web browser, so that the authentication between the human and a server has three parties, being the human, the human’s computer with a web-browser, and the server. Secondly, the human and the human’s computer are given specific functions in the security proof model. These functions were that the browser on the human’s computer renders a webpage (based on browser state), and the human must be able to recognise what Gajek et al. called a human perceptible authenticator (HPA) [44]. The HPA can be anything, but in the protocol [44] the HPA was an image previously selected by the user and sent to the server, for the server to use in subsequent protocol runs as their authenticator to the human. By adding these functions, the human’s involvement is partitioned from the non-human protocol messages, and a formal proof of security is created.

This technique of creating a protocol proof with the human assumptions being included but partitioned in such a way that a human trial will inform how secure the protocol is, is a significant step forward in the quest to prove proto-
cols secure for human use. However, a complete design, informed by iterative cumbersome protocol-specific human studies following each new protocol design and developed proof, would potentially take years with no guarantee of success. Human protocols do need to be verified via human trial post-theoretical proof, however simply writing a security proof in terms of the human is not sufficient and a method is required for arriving at a design more likely to succeed.

2.2.3 HTTPS Usage

A protocol used daily by many internet users is HTTPS which is the secure version of the basic HyperText Transfer Protocol (HTTP) used to view webpages over the internet. Any banking transaction, online purchase, accessing on online email or social media, will typically be conducted using HTTPS. The security is accomplished by the Transport Layer Security (TLS) protocol suite. As such, HTTPS and TLS are used interchangeably.

To give a feel for how often this HTTPS protocol is used each day, in February 2012 a certificate authority claimed to provide “over 3.5 billion OCSP lookups every day” [98]. To explain what this means, one usage of the HTTPS suite of protocols is that security certificates may be used by parties to identify themselves. The certificates are provided by a certificate authority and have public information mathematically related to private information which is given only to the owner of the certificate. The certificate means that only the holder of the certificate, who therefore has the private information, can decrypt anything sent to them by any other party who uses the public information (or cryptographic key) from the certificate to encrypt the message.

There needs to be a method of revoking certificates, for example if they have been found to have been issued to the wrong entity. Therefore, as part of the HTTPS protocol which is using the certificates, a check should be made by Internet browsers to ensure the certificate has not been revoked. The method of checking if a certificate has been revoked is via the Online Certificate Status Protocol (OCSP). Another factor is that in the majority of HTTPS sessions, only one party, the server, holds a certificate. Both parties or neither party could hold a certificate, but the majority of HTTPS sessions has only one party holding a certificate and this is what would be checked using OCSP. Therefore, while other protocols may use security certificates, such as secure email or Virtual Private Networks (VPN), a single certificate authority having over 3.5 billion OCSP
lookups every day gives an idea of the number of HTTPS connections that are made each day, as HTTPS is the most common usage of security certificates. Many of the examples from our thesis focus on the use of HTTPS in security ceremonies.

### 2.3 HCI and Sociotechnical Considerations

HCI research on browser-based authentication protocols has revealed much concerning what humans can, what humans will not, and what humans cannot, do, drawing over the years from what Harrison et al. have identified as three broad paradigms of HCI research – a-theoretic, cognitive and situated [54]. Lessons can be learned from initial work by Simon [113, 114], which showed us the boundaries of human short term recall, and cognitive load issues, through to specific controlled studies on decision making in use of security systems. An example of such research is by Schechter et al. who created a study in which bank websites were progressively changed, to become less and less secure, and the researchers determined whether the participants continued to enter their password into the website (which they did) [108]. Our research has indicated that a recent security improvement, which attempts to provide users with the necessary authentication information via the use of Extended Validation Certificates [43], and the associated inbuilt functionality in current browsers to colour code and present typically real world company name information to the user, is not being used by web-users in their web security decision making (see Chapter 3) [92].

Dourish has provided a bridge between social science and HCI design, contributing significantly in areas such as defining and using context [36]. Of specific concern, when defining context, was the impression (still common seven years later) that context is fixed, explicit and can be adequately captured by explicitly measurable information rather than something that is “...being continually renegotiated and defined in the course of action” [36]. One simple application of the concept of context is the case of the **rushing user**. A **rushing user** is used by Kumar et al. to describe a user who, in a rush, takes the shortest path through a protocol, skipping steps which are not required for subsequent steps to work [68]. As Dhamija et al. describe, security is typically not the primary task and hence users may not notice security indicators or read warning messages [35]. There is also a body of work which focuses on achieving security by aligning what a system
does with the user’s mental models of that system [104,115,126]. As far back as 1975, there were design principles created for securing computer systems, all of which are still current and two of which should perhaps be reminded today: *keep the design as simple and as small as possible, and it is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly* [104]. As Smith states, “Repeatedly, I ended up with problems because what computers are doing with cryptography doesn’t match the mental model that humans have - end users as well as system programmers [115].” More recent work includes Chiasson et al.’s research into constructing a set of design principles for security management systems [23].

The concept of aligning the actual system to the user’s mental model of the system (or vice versa) is useful at a guiding level along the lines of “the user must understand what the system is doing, and what the response to her actions will be.” However, the concept of the human cognitive model that exists prior to the situation is a contentious one. There is significant evidence that people co-construct meaning using embodied competencies and situational circumstances [117]. Suchman argues understanding conversations and interactions, as dynamic co-constructions, could prove more useful for designers of human-machine interactions. The lesson we take from this body of work is the necessity for the user to be in control and to have visibility of (and to understand and actively participate in), ideally, the cryptographic authentication processes. This is in keeping with the central concept of Norman’s popular design book, which is “when people have trouble with something, it isn’t their fault - it is the fault of the design [85].” The human user being in control is a fundamental driver for the design of the CHURNs presented in Chapter 6.

Finally, Sasse et al. have argued that existing HCI techniques are sufficient to address security issues in the design of systems [105]. While this may be true, we argue that it is also necessary to understand the security requirements and establish a consistent security framework.

### 2.4 The Need for Freshness in Cryptographic Protocols

In general terms, a *replay* attack is an attack where messages sent between participants in a protocol are captured by the adversary and stored, to be re-
sent (replayed) at a later time to the participants. A replay attack is defined as “The adversary records information seen in the protocol and then sends it to the same, or a different, principal, possibly during a later protocol run [13].” It should not be possible to replay messages from previous protocol runs as valid messages in the current run of the protocol. To ensure that replaying does not occur, some assurance of the freshness of messages, meaning the message is new and has not been used before, is required. Three traditional methods used to ensure freshness are random nonces, timestamps and counters. A random nonce is defined as “a random value generated by one party and returned to that party to show that a message is newly generated,” and critically the value can be used only once [13].

![Diagram of the 1978 Protocol by Needham and Schroeder](image)

Figure 2.1: 1978 Protocol by Needham and Schroeder (as shown by Boyd and Mathuria [13]).

As an example of a replay attack, Figure 2.1 displays the famous protocol by Needham and Schroeder [84] as shown in the protocol design book by Boyd and Mathuria [13]. The symbols used in Figure 2.1 and Figure 2.2 are defined as:

- **S** is a trusted third party, which holds long term keys for communication between **S** and **A**, denoted **K**\textsubscript{AS}, and between **S** and **B**, denoted **K**\textsubscript{BS}.

- **A** and **B** are entities, possibly with no prior relationship, who wish to use their existing relationships with **S** to establish a new key **K**\textsubscript{AB}, which **A** and **B** will use for future communication with each other.

- **N** is a random nonce, that is, a random number used only once. Therefore, **N**\textsubscript{A}
is a random nonce generated by \( A \), and \( N_B \) is a random nonce generated by \( B \).

\( K_{XY} \) is a secret key between parties \( X \) and \( Y \).

The Needham and Schroeder protocol shown in Figure 2.1 already made use of a random nonce to ensure freshness, but only for the messages to party \( A \) via the use of random nonce \( N_A \). The lack of a random nonce from party \( B \) led to an attack being discovered by Denning and Sacco [31]. Message 3 from Figure 2.1 has no assurance of freshness, that is no random nonce, timestamp nor counter, and hence an adversary could replace message 3 with a previously recorded message 3. This would mean that party \( B \) would communicate with an old key \( K'_{AB} \) which the adversary may have knowledge of. The solution was to include a nonce from \( B \) in the messages, as shown in Figure 2.2 [13].

![Figure 2.2: Protocol from Figure 2.1 with Random Nonces for both Parties (as shown by Boyd and Mathuria [13]).](image)

timestamps and counters have a common flaw which requires that protocol participants remain in synchronization with each other. To achieve synchronization, usually both a prior and a continuing relationship are required. Therefore, in Internet-based applications where no prior nor continuing relationship is mandatory, random nonces are most commonly used to provide freshness assurances to messages.
2.5 Notation and Nomenclature

Throughout the thesis, a number of symbols, abbreviations and names are used. Cryptography blends together algebra, set theory, discrete mathematics, logic and computer science. As such, some explanations are provided in Table 2.1 for reference when reading the thesis.

Table 2.1: Notation and Nomenclature for use in thesis.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>∈</td>
<td>Means “is an element of.”</td>
</tr>
<tr>
<td>⊆</td>
<td>Subset. ( HPASpace_H \subseteq HPASpace ) may be read as “Every element in the set ( HPASpace_H ) is in the set ( HPASpace ).”</td>
</tr>
<tr>
<td>∪</td>
<td>This is the union of two sets. The union of two sets is the set of all elements from each set. Since there are no duplicates in a set, an element is either in the set or not in the set, any element that is contained in both sets is shown in the union set only once.</td>
</tr>
<tr>
<td>\</td>
<td>This is the set-minus symbol. ( { HPASpace \setminus HPASpace_H } ) may be read as “what is the set of all elements of ( HPASpace ) which are not in ( HPASpace_H ).”</td>
</tr>
<tr>
<td>{x|f(x)}</td>
<td>This is a predicate notation that defines a set. The “|” symbol may be read “such that” or “where”, as the values to the left of “|” are the values returned when the boolean condition to the right of the “|” returns true. Therefore ( {x|f(x)} ) may be read as “what is the set of all ( x ) where ( f(x) ) returns true.”</td>
</tr>
</tbody>
</table>

Continued on next page...
Table 2.1 – continued from previous page

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>←</td>
<td>This means “an algorithm outputs.” Other words that may be used are “a function outputs” or “an oracle outputs.” That is, a computer program outputs a value based on a sampling distribution or some sort of logic. The symbol is critical because it captures the concept that the value output each time the algorithm runs may not be the same, even though the input is the same. So $HPA_H \leftarrow GenHPA(H, HPASpace)$ may be read “the algorithm $GenHPA$ takes as input $H$ and $HPASpace$, and outputs $HPA_H$.”</td>
</tr>
<tr>
<td>·</td>
<td>This is used as a wildcard, meaning it can take any value. Therefore, for a function $GenHPA$ which takes two inputs, a human $H$ as the first input and a space as the second input, writing $GenHPA(\cdot, HPASpace)$ means that any human $H$ can be used as the first input to $GenHPA$.</td>
</tr>
<tr>
<td>⊕</td>
<td>This is the exclusive OR symbol, meaning that for two values there can be either one or the other, but not both. When considering values of 0 and 1, this means that (0,0) and (1,1) return 0 while (0,1) or (1,0) return 1.</td>
</tr>
<tr>
<td>∑</td>
<td>A symbol denoting the addition of a sequence of numbers. $\sum_{i=1}^{n} \text{ may be read as “the addition of all values in a sequence from value in position 1 to the value in position } n.”$</td>
</tr>
<tr>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>${0,1}^\kappa$</td>
<td>This is short hand for writing “a $\kappa$ length sequence of 0s and 1s.”</td>
</tr>
</tbody>
</table>
Chapter 3

Learning about Human Users

To discover essential aspects that need to be considered when designing cryptographic protocols that involve humans, we focused on internet and web browser usage as an area where humans use information security every day. Even though web security protocols are designed to make computer communication secure, it is widely known that there is potential for security breakdowns at the human-machine interface. Our analysis of protocols which involve humans where the security of the protocol is known to be broken has confirmed that one source of security flaw may be attributed to the designers of systems and software. As described in Section 4.2.1, the design of the human-computer interface in that web-based security ceremony is such that the necessary information for making an informed security decision is hidden from the user.

Since the research community does not have a good understanding of how people make security decisions, we conducted two studies with human participants at the start of our research. In the first of the studies we left our participants in their own context and they kept a log of their decision situations, their thought processes to make their decision, and the resulting decision. In the second of the studies we conducted in-depth interviews with our participants. These exploratory qualitative studies provide the basis for the research, analysis and decision making in our future work. As such, these two early qualitative studies were framed in an open fashion which gained information from participants and themes were distilled from the data collected. This is in contrast to the third and final human study we did, presented in Chapter 6 which was a quantitative
study conducted after a design was created to validate a design.

3.1 Introduction

The human-machine interface is acknowledged as one of the primary challenges in designing secure human-computer security systems [89]. As discussed in Chapter 4, the combination of the human users, (potentially multiple) cryptographic protocols, and the various systems which are used to interact with the protocols, may be described as a “security ceremony” [41].

The central figure in Human-Computer Interaction (HCI) is “the user” [106]. Gaining an understanding of how users make security decisions when using a web browser is very complex. Whether they are aware of it or not, when browsing the web, people make many security related decisions. However, little research has been conducted to explore the range of security decisions made and perception of those actions.

Our analysis of security ceremonies known to be broken [41,83], presented in Chapter 4, has revealed that one source of security flaws may be attributed to the designers of the systems and software. From our case study work on what users wish to keep private, and the tensions that the need for privacy creates versus the information a device or technology needs to function correctly, we have seen how groups of users will build up a collective information practice [47, 95]. This collective information practice is the way a group of people will collectively develop a method for using a design, which is overlaid with the concept that the way a design is used is rarely the way the designer expected. Since, from the human perspective with respect to web usage, we do not have a good understanding of how users make security decisions, it became critical to explore the ways that humans were actually making security decisions and using their web browsers, in their own environments. The purpose of our studies was to examine the security system in its entirety, from the perspective of interaction design, examining the users in their context of use. By understanding such decision making processes, our aim is to improve future designs of interfaces to better protect user security.

Our exploratory studies used qualitative approaches to investigate users’ web usage in their natural environment. In our first study we used week-long diary entries for 20 participants. In our second study we used in-depth individual par-
participant interviews with 18 participants from the initial group (two participants were no longer available) exploring their security decisions made in their web usage. From this data we distilled common themes about users’ security decisions concerning web usage.

3.2 Background

Our work is focused mainly on design aspects of security of online programs and entities, particularly web browsers and websites. As such, the prior work which most closely shares the ideals and approach of our work is by Dourish et al. [38]. Their research, which focused more on mobile and wireless ubiquitous computing, similarly used semi-structured interviews to gather data, and grounded theory to analyse the data. This is appropriate when searching for the right questions to ask rather than the answers to specific questions. Our work adds a further piece to the landscape that Dourish et al. began to describe. While they focused on a set definition of security and to what degree existing approaches met that definition, we, using both diary and semi-structured interview data collection techniques, explored what is important to users, what their definition of security is, and what systems individual users put in place to ensure their security goals are met.

There have been many controlled studies in decision making regarding security and privacy in the use of systems, for example, studies into MySpace and Facebook security and privacy settings [69,88]. Lampe, Ellison and Steinfield, in their study of 1085 Facebook users which explored users’ expectations of privacy, found that 90% of participants believed that no one from outside their university would read their Facebook page. 97% of participants believed that no law enforcement agency would look at their Facebook page [69]. These types of studies, while useful, examine what has already been implemented in the systems to protect users’ privacy and security, rather than what the users of the systems believe should be protected. Qualitative research is required to gain an understanding of both what the users wish protected, and the methods of protection which will be intuitive to them.

Various studies concerning trust on the internet, such as that done by Lee et al. [71], have been conducted using surveys. However, while these works provide overview data, the survey methodology employed tends to generalize across
contexts and frame questions from the researchers’ perspective. This means that little insight is gained into each individual’s priorities, decision making processes and practices. This was also the conclusion of Connelly et al., when they stated that surveys were not able to reflect participants’ privacy concerns accurately, and that the discrepancy depended on the context [25].

Several studies have been conducted into the effectiveness of web-based security warnings [16,108,116]. In their study, Sotirakopoulos et al.’s primary finding was that laboratory experiments concerning security decisions were fundamentally flawed. They observed that the most common reason the participants gave for ignoring security warnings was that they trusted the study in which they were participating. Secondly, they stated that the security conscious were reluctant to take part in such studies, so we have ensured we have included members from the computer security community [116].

Sotirakopoulos et al. and Connelly et al. raise questions about the ability to draw security practices and user behaviour conclusions for the general population from such non-contextual studies. This has led us to take a different approach centred on people and their own contexts of use.

We examine a more fundamental question of, “What is it that the users are trying to protect?” rather than first presuming what needs to be protected and secondly attempting to create more useful warnings to protect those presumed areas.

There is a body of work which focuses on achieving security by aligning what a system does with the user’s mental models of that system [23,115,126]. The concept of aligning the actual system to the user’s mental model of the system (or vice versa) is useful at a guiding level along the lines of “the user must understand what the system is doing, and what the response to their actions will be.” However, the concept of the human cognitive model that exists prior to the situation is a contentious one. There is significant evidence that people co-construct meaning using embodied competencies and situational circumstances [117].

### 3.2.1 Trust

Literature from the past 15 years is replete with papers concerning trust on the web. The reality of a world wide web necessitates investigating trust issues in a number of categories. Areas of trust include what makes users trust a website,
the role of trust in customer loyalty, and how to address trust across cultural boundaries [26, 40, 55].

Attempts to establish trust with people the first time they visit a website are usually aimed at a range of triggers upon which users have been “trained” to base their decision to proceed. These triggers include the list of measures taken to ensure data is transferred, processed and stored securely, and displaying seals of independent trusted third party auditors [40].

Notably, there appears no consensus regarding whether ease-of-use is a factor which strongly relates to the user trusting a website. Some studies have shown there is little correlation [46, 90] while other studies have shown a strong relationship [99]. In addition, usability has been shown to be a security issue, with poor usability directly impacting the security of a system [122].

Spelling and grammar have been found to be a factor in establishing a user’s trust in a website. Poor spelling, grammar and syntax create doubts about the party’s identity and thus impede what is termed calculative and knowledge based trust [66].

Koehn divides trust into four categories:

1. Goal-based trust: the trust that arises when two parties have a similar objective.

2. Calculative trust: trust that is created based on evidence. Is there a good reputation and a history of keeping promises?

3. Knowledge-based trust: trust based on knowledge of the other party, their character, and having worked with them previously. This is not distinct from calculative trust.

4. Respect-based trust: the parties respect each other, do not wish to exploit each other, and are open to constructive criticism [66].

Most efforts to establish trust between vendors and buyers online, such as security certificates, are targeted at calculative and knowledge-based trust. Respect-based trust is more lasting than either calculative or knowledge-based trust [66]. Finally, trust is significant with regard to customer loyalty. Specifically in the context of e-commerce, significant supportive evidence suggests that customers who feel a higher level of trust in the store will revisit the site more often [71].
3.2.2 Privacy

Privacy is generally approached as a social consideration, whereas security is seen as a technical concern, though they are closely related [37]. We argue that technical security decisions have such an impact on privacy, that privacy needs to be considered from a technical perspective, in order to ensure that the privacy expected from the social perspective is protected.

3.2.3 HTTPS and Extended Validation Certificates

Most users have visited websites with addresses starting with “HTTPS”. Addresses starting with HTTPS should mean that a secure connection has occurred between the user’s web browser and the viewed web site, accomplishing the security goals of confidentiality and integrity.

There is nothing in HTTPS which guarantees who the owner of the certificate is, and thus there is no guarantee as to the identity of the other party. We know that only the owner of the corresponding private key can read messages sent, but not who the owner of the private key is. All that the user can be assured of is that they are securely connected to someone, and unfortunately that someone may not be the entity the user believes they are connected to [41]. To help combat this, Extended Validation Certificates have been introduced.

The process of acquiring an extended validation certificate enforces that the holder of a certificate, required for HTTPS communication, is who they claim to be [43]. This allows web browsers to display the name of the company who owns the website, as well as the company’s web address. Recent enhancements include colour coding of the address bars in web browsers which is now standard, and more effective visual warning techniques regarding security certificates as described in Maurer et al.’s research [78].

3.3 Methodology

Although there were two trials at two different times, the participants in each trial were identical, except for two participants who left Australia before the second study started. The first study we call Web Logs and the second study we call In-depth Contextual Interviews.
3.3. Methodology

3.3.1 Participants

Twenty participants (18 for the second study) were recruited in Australia via two research groups (human-computer interaction (HCI) design and information security) and from personal contacts of the researchers. Seven participants were from the general community (aged between 25-64), six were HCI design researchers who build prototypes and systems for humans to use (aged between 25-54), and seven participants were computer security researchers (aged between 25-74). 45% were female and 55% were male, with all having tertiary qualifications. Nine participants were recently from a country other than Australia (within the last two years), while 11 participants have been in Australia for more than 10 years. There were eight nationalities, from four continents, represented. All participants owned their own computer (at least one).

3.3.2 Procedure

Participation involved a week long diary study of web-based security decisions, typically accompanied by pictures of their screen at the time of decision, followed by in-depth qualitative interviews.

Web Logs / Diaries

Participants were asked to keep a one week log of their security decisions made while using the web on their computer. A template for the log file was provided to each participant. The template, a Microsoft Word document, consisted of a table with three columns. The columns were titled:

- Screen image (of the web page)
- Thoughts about the security decision
- Your security decision

An example web log entry from one of the participants is shown in Figure 3.1.

Web logs were encouraged wherever the participant used the Internet, be that home or work, with some participants having a web log for home and for work. Participants’ web logs ranged in size from one participant having one entry through to two participants having more than 15 entries. Typically web logs had four to seven entries.
In-depth Contextual Interviews

Participants’ perceptions of computer security, their decision making processes and their subsequent actions were elicited via semi-structured in-depth interviews. The interview process was conducted in the tradition of Beyer and Holtzblatt who recommend a Master-Apprentice technique, where the interviewer is the apprentice learning from the participant who is the master [9].

These 30-70 minute interviews covered four broad topics, including: a) participant’s experience using computers, b) the meaning of computer security, c) the range of activities and settings used on the internet and d) responsibility for computer security. In this way, the interviews explored the participant’s background to better understand their experiences and reasons for their methods and actions, without resorting to subjective “assess your computing knowledge on a numerical scale” techniques.

Throughout the study, in all communications the term security was used, with no further delineation provided. As Yee states, “It is impossible to even define what security means without addressing user expectations [125].” To this end, in their interview, conducted after completion and submission of their web log, each participant was asked to complete the sentence “Computer security means to me...”
This approach of blending interviews and web logs was chosen as a useful way to achieve an understanding of decisions made in the context of use. Diary studies have the shortcomings of any self-reporting mechanism, but the diary studies provide significant insight into a participant’s decision making process in their own environment, and the self-reporting issues were partially mitigated via the subsequent in-depth interviews. This methodology provided us with rich information about how each individual uses the web in their normal environments.

### 3.3.3 Analytic Method

The data collected from the web logs and interviews was compared and analysed, with a thematic analysis conducted to identify themes and patterns [48, 72, 73]. Through reading and rereading the data, web log entries and interviews were investigated and classified. Coding was done with common and contrasting concepts identified. Finally, themes were identified to create a comprehensive picture of how and what users protect on the web.

### 3.3.4 Ethics

Permission was gained from our university’s Human Ethics committee for each of these trials. Risks were minimal compared with every day computer usage. The greatest risk was the possibility that a participant may include some private information they did not wish to share in a screen-shot for the web-logs. To mitigate this risk, this risk was explained to all participants and the participants had the option of writing about their security decision without including a screenshot. All participants were able to cease being part of the trials at any time.

### 3.4 Findings

There were two broad themes presented in our data. Firstly, a problem was identified, which was that there is a lack of delineation between what should be trusted and what could not be trusted. This problem has led to our participants individually realizing that their security and privacy are insufficiently protected. Secondly, we outline what behaviours and processes the participants have exhibited, on an individual basis, to protect themselves and mitigate against the
risks created by this absence of rules. This leads to our discussion regarding the impact of our findings to HCI design.

3.4.1 Identified Problem: Users Need to Protect Themselves on the Internet

Particularly the participants from the general population and the participants who design human-computer interfaces identified that there was a lack of delineation in the web interface regarding what could be trusted and what could not be trusted. This led to an unknown status regarding their privacy and security. Secondly, a continuum of concerns, for which the users need to have security and privacy provided, was identified. Thirdly, the participants responses provided a clear message that the responsibility for security is being forced on the users.

Lack of Delineation Leading to Obscured Security

Several users, in both their web logs and interviews, cited aspects of a web page as reasons to trust a website. These aspects included the VeriSign logo, privacy policies, FAQs, and terms and conditions. In line with Eggar’s guidance on how to make users of e-commerce websites feel more comfortable, participants reported feeling more confident if a local physical address is listed, and a local telephone number [40].

However, a contrasting view was reported from other participants. They wondered how hard it would be for the designer of a malicious web page to include these items, and whether they should be basing her decision on them. In direct contrast to Eggar’s guidance on how to make users feel more comfortable, a participant from the general population stated in his web log that he looked for HTTPS and the padlock symbol, in the interview he made a point of saying that he did not trust graphics such as the VeriSign logo posted on the web page, or statements made on the website about encryption.

The concerned users asked two essential questions: “Who decides?” and “How do I know that it hasn’t been faked?” The question of “Who decides” is central to users having trust in their web experience, including both the web browser and the web page. There is no overarching rationale or conceptual explanation of why programs are the way they are. Who decides if a padlock is shown? Who decides if the company name is shown in the address bar and
3.4. Findings

Findings elsewhere on the page? This is a step away from providing clearer or more obvious security indicators or warnings (such as in [78]). As a step before enhancing security indicators or warnings, the user needs to clearly understand the question of “Who decides” if the security indicator or warning is shown.

To counteract a lack of assurance that the user is connected to the correct website, Extended Validation Certificates were introduced in 2007, which has led to colour coding of address bars in modern browsers. However, not one participant in their week of logging their web-based security decisions, referred to making use of any of the additional information Extended Validation Certificates afforded them. This is in keeping with Lin et al.’s finding regarding domain highlighting, where they found in their study that domain highlighting also had little benefit [74]. Domain highlighting is another enhancement, similar to colour coding based on Extended Validation Certificates, also aimed at preventing users from visiting incorrect websites.

In this section we highlight that the delineation between what is easy for a malicious agent to manipulate, and what is hard for a malicious agent to manipulate, is currently unclear in the eyes of our studies’ participants. Once this distinction has been clarified for the users, further work is required to make transparent to the user the process of who decides when the security indicators are shown. Again, particularly the question “Who decides?” was another critical driver for the concept of “Human Followable Security”, as described in Chapter 7. Notice that this is a fundamentally different approach from previous HCI work, attempts to provide more understandable warnings, more obvious warnings. To complete the picture as to why human-followable security is a different approach to creating better warnings, the question that should be asked is, “Who decides whether the warning appears?”

Web Concerns Continuum

Participants with little computer security education had different concerns from those of the security specialists. Although practices are individualized for all participants, there are some notable differences between the concerns of security professionals and general participants. For those with little computer security education, most of their concerns revolved around the entity they were communicating with over the web. For those with significant computer security education, almost all concerns were with their own computer and with the communication
channel connecting their computer to the entity they were interacting with over the web. We characterize the range of concerns from computer to connection to entity to recipients as a web concerns continuum, noting that people with and without security education are largely concerned with different aspects of this continuum.

The concerns the participants from the general population had regarding the entity they were interacting with. These concerns included employees of the entity, combined entities, mistakenly sending information to the wrong entity, and non-entities. Multiple businesses of the same type placing information online, such that customers could log in from anywhere and access their details, made participants feel as if that one location was going to be an attractive target to hackers. Other concerns were regarding sending private information to the wrong person, or transferring money to the wrong account. Caitlin, who works from home, was most concerned that the entity may no longer exist:

*I must admit sometimes I buy stuff and I think, “What if this website is a year old and nobody is actually at the other end and people are just collecting my credit card details and nothing is actually going to happen?”*

*And even getting an email back doesn’t mean anything because so many of them are automated!*

In contrast, participants from the security community were far more concerned about their immediate interface device and the communication channel from their device to the entity they were communicating with. Not one security participant referred to the entity at the other end, in the way that the general population was concerned about issues such as the already discussed non-entities, or who was employed by the company. With respect to the device, the concerns ranged from the type of operating system, to the browser, to who was the main user of the computer, and even if the operating system was stored on re-writable memory.

In this section we have shown a range of issues on the web concerns continuum that includes the user’s computer, the channel to the website with which they are communicating, the website, the business whose website is being viewed, and the employees of the business. No participant thought of the entire continuum, but allaying fears for the entire continuum is required to satisfy all users.
3.4. Findings

Users are Responsible for Computer Security

Particularly notable is that all participants except one, when asked who was responsible for their computer’s security, answered at least partially “Me”. The exception was a design researcher with a Masters degree, Chloe, who believes that the responsibility rests entirely with the software on her computer. For most participants, the response of “Me” was an absolute, especially for their home computer. This may be a result of a culture that has developed from many years of End User Licenses for software removing all responsibility from the software developers, and placing all responsibility on the user. No participants were employed as system administrators. Effectively, this means that untrained people are drawing on their personal experience and are individually making the best use of the limited set of tools available to them, to protect their computers and online activities.

3.4.2 Users’ Actions and Motivation

Having identified that there is a problem, that the participants feel the need to protect themselves, the responses of the participants revealed how they were protecting themselves and what they were protecting.

Individual Mitigation of the Absence of Rules

The most consistent theme observed through the analysis of the web logs and interviews was the diversity of security techniques employed while using the web. All participants used their computer daily, all were university educated, and a third of the participants were security researchers. Therefore, some level of consistency in line with a collective best practice may be expected. Instead, there was little or no consistency, especially when each participant’s technique was explored in detail in the interview. Each participant was found to be protecting themselves to the level of their knowledge. This is indicative of a system without defined and enforced rules.

We separate the responses into three groups: the general group who will use the system, the HCI researchers who design the systems, and the security researchers who may be considered experts to give insight into the extremes of the measures taken. In keeping with Yee’s observation that, “It is impossible to even define what security means without addressing user expectations,” [125] we
will include the participant’s definitions of what security means to them.

**General population:** Rebecca has different definitions for security if she is at home or at work. She completes her financial transactions at work since she believes the computer system and network at work is a more secure system than at home. In contrast, Georgia, a government worker with a Bachelor degree, whose definition of security asks whether she is broadcasting her information to other people, specifically will not complete any financial transactions at work. She states that this is because at work she has no control over the network, software, or how often protective programs are run. Georgia uses a low limit credit card, PayPal, and EBay, since she is confident that if she challenges a purchase attributed to her via these means, they will refund her money. Brian, whose definition of security is to feel confident and uninhibited to do the task he needs to do, searches, shops and makes bookings online but sends credit card details via fax.

**HCI researchers:** Bianca, PhD, defines computer security as only she can access her computer and no one else can. Bianca posts her credit card details using traditional mail to smaller organizations, rather than send the details over the internet. If she must send her credit card details over the internet, then she does so via email and splits the credit card number into two emails. Chloe, whose definition of computer security is protecting her data from being accessed by other people, never purchases online. She researches her purchases online, but then travels to physical stores to make her purchases.

Sarah separates her definition of computer security into protecting the computer from attacks that come over the network, physical protection such as not leaving the computer in a location where it may be easily stolen, and backing up her data. Sarah completes almost all of her shopping, including groceries, online. She has a low credit limit credit card solely for online purchases, and separate bank logins. With one bank login, she can access all of her accounts, but cannot transfer to any account that is not her own. With the other bank login, she can only access one account which normally has a very low balance, and with this login she can transfer money online to other people’s bank accounts. When Sarah needs to transfer to another person’s account, she logs into her bank with the first login. She then transfers from her main account to the account that can be accessed with the second login and logs out. Sarah then logs back in with the second login, and transfers the purposefully placed money to the other person’s
account.

**Security researchers:** Samuel, who has a Masters degree, has a security definition which he separates into three parts. Those parts are: physical security similar to Sarah’s definition; network security which is the transmission of information point to point; and information security where the main goal is to hide the information. Samuel specifically asked for, and acquired, an electronic device from his bank which provides him with a new electronic token every minute. Each time he wishes to log in to the bank, he uses his login and password, and the extra electronic token. Once logged in, he uses the changing token for any transfers out of the bank also.

Dylan has a definition of security which he separates into the computer’s security and the communication security. The computer’s security refers to the machine, operating system, and software. The communication security refers to working safely on the internet. Security is looking after assets in a broad sense - whatever is important to you. Dylan consciously chooses to use the Linux operating system when browsing the web, since there are fewer attacks designed for the Linux operating system than for more mainstream operating systems. In particular, Dylan will not make purchases or conduct online banking on his children’s computers.

Nicholas, PhD, defines computer security as being there to secure personal information, and personal transactions. Nicholas has an operating system on CD, which he boots from when he wishes to use the internet for banking, and then uses only the web browser. He does this because booting from a non-rewriteable CD minimizes his exposure to social engineering attacks via email, and exposure to malicious software is restricted to the time the computer is powered. As soon as the computer is rebooted, the system would need to be compromised again.

In this section we have provided examples of how users are individually addressing the problem of needing to protect themselves on the internet. Critically, everyone was protecting themselves to the best of their knowledge and expertise. This finding was a critical driver in the creation of the concept of Human Followable Security, as described in Chapter 7.

There was not a delineation of “those who care about security” and “those who don’t”; rather, everyone cared and already employed the best scheme they knew. A theme common to the techniques employed is the theme of separation,
for example using a separate credit card or bank account, sending information via another communication channel such as facsimile or email, or even using a separate computer or operating system. While educating users about the security and privacy settings of the latest online applications and devices may be unrealistic, general lessons such as being aware of and maintaining separations may have some merit, and HCI designs should enhance rather than undermine these separations for the sake of security and privacy behaviours.

**What Users Wish to Protect**

Having identified how the users are protecting, we now outline what participants stated was important enough for them to want the information protected, which was found to depend on the participant’s circumstances. Protected information included:

- financial details and accounts used to access money,
- personal information such as physical address and telephone number for themselves,
- details concerning children such as photos, names and schools,
- medical details, and
- information that could directly lead to identity theft such as a scanned image of their passport.

Participants typically had three levels of safeguarding, though some had only two. Always in the top level was financial information. Nicholas said the reason for this was because it was “real money.”

**Money:** Placing money-related web interaction in the category that requires the greatest protection was consistent across study participants. This categorization occurs even though some participants consciously noted that they had never heard of an online attack on a bank. Participants described that they was more aware of physical devices being added to Automatic Teller Machines, which would acquire their card details and Personal Identification Number, than any losses made due to Internet banking. Even without concrete awareness, all users placed online finances in the most protected category.
3.4. Findings

**Medical conditions:** None of the participants in the age groups younger than 45–54 mentioned concerns regarding searches for medical conditions. Almost all participants in the 45+ age groups raised concerns about searches for medical conditions being traceable back to them. These participants highlighted that if they visited a physical library and read a book inside the library, no one would have a record of that occurrence. However, searching for the information on the internet created a permanent record which could be linked back specifically to the participant’s machine. A participant drew an analogy to HIV 30 years ago, when there was a stigma associated with the disease, and that there are ailments with similar stigmas now. As an example, she considered the case of a person using the Amazon website to purchase a book on a topic with a similar stigma, and worried that the website with its knowledge of past transactions would prompt the user “You may be also interested in....” She stated, “The prompt insensitively indicates that they, an online bookseller, know all about the medical conditions you are interested in.” Patrick notes that records of the interaction would exist at potentially multiple locations such as the participant’s Internet Service Provider (ISP), the search website they used, and the subsequently visited website.

**Personal Levels of Security**

Some participants control their levels of security by password use. If a website as critically important, then the website gets its own password. Less important sites get shared passwords. One participant’s method of categorizing the websites is based on finances. Her lowest level of shared password is used for websites, such as EBay and Amazon, where obtaining her password allows an attacker to make purchases or bids on her behalf, but not pay for them. She hopes and believes that the people involved at EBay and Amazon would understand if an attack occurred. While participants wished for the ability to group their passwords, they noted that there is no consistent rule for what is a secure password. On some sites, the rule for what constitutes an acceptable password may be letters and numbers only, while others may require letters, numbers and special characters.

Most participants had three levels of concern, with one participant having a separate physical device for each of the levels. As mentioned, for banking and anything with “real money”, he boots a laptop using an operating system on non-re-writable CD. At the next level down, for personal items that could lead
to identity theft, he has an encrypted USB stick. This encrypting both restricts access under what he describes as normal “browsing the desktop” circumstances, and also protects in the case of loss of the USB stick. At the lowest level are the websites he is not concerned about, such as news, Wikipedia, and the universal library.

In contrast, some participants had only two levels. Chloe states that as long as the website is related to money, then she looks for security indicators. If the website is not related to money, then it is “just for fun” and she is not worried about privacy issues. Specifically, she does not mind sharing her name and address. Chloe’s view, that the most important protection aspect for her was protection from her data being lost, was unique for the participant group. She stated that she prefers to share the data than to lose it. Also in contrast to the three-levels of security that most participants had was that for some participants, the levels are fluid. They decide whether to enter personal information on the context, and whether they believe the organization needs that information.

In this section we have provided the range of information that users identified as being worthy of special protection. We have shown that most users sort their information into three tiers of security importance, though some users have only two tiers. Common amongst all participants was the need for financial security, and then, depending on the participant’s situation, items such as information about their children or medical information became critical for them to protect also.

3.5 Discussion and Impact to Web-Based HCI Design

We have shown that the delineation between what is easy for a malicious agent to manipulate, and what is hard for a malicious agent to manipulate, is currently insufficient in the eyes of our studies’ participants. This has led to an unknown status of security and privacy, where the participants, none of whom are trained system administrators, are responsible for the security and privacy of their systems. Each participant was therefore protecting themselves to the best of their ability, based on their experiences, with the limited set of tools available to them.

Already, this study has highlighted some major issues, such as the need for an overarching set of guidelines for browsers, which allows users to understand which
parts of what is displayed to them are browser-controlled and which parts are website-controlled. Secondly, increased responsibility for security and privacy is required with the software production companies - all end user license agreements (EULAs) examined for major browsers and operating systems had disclaimers such as “AS IS” and “WITHOUT WARRANTY OF ANY KIND.” There have been notable shortfalls in the trustworthiness of programs, even amongst the world’s biggest software companies. These include compiled-in “back door” accounts that allow access to all implementations regardless of access and password settings, and backup facilities that corrupted files as they were copied [21,118].

Our research has described a web-security continuum, which the participants have described as being the range of areas that they are concerned about regarding security and privacy. This continuum includes the user’s computer, the channel to the website with which they are communicating, the website, the business whose website is being viewed, and the employees of the business. This means there is a need to design for trust at the computer, communications, website and business levels, and the whole continuum holistically, for web usage to be unhindered by security and privacy concerns. An overview of the findings from our two human studies are presented in Figure 3.2.

We have described how, in the absence of an effective security and privacy solution, our participants have individually constructed and implemented schemes to enhance their security and privacy protection. Central to many of the disparate measures was the concept of separation, ensuring that, as much as possible, a malicious entity could not acquire what the participant would like to protect by compromising one location. This is an essential message to the designers of web applications and signup processes, who may be designing systems which undermine their users’ main protection mechanism, which is keeping information separate. An example of undermining of users’ protection schemes is described in the experience Dylan shared regarding PayPal:

*I had an issue with PayPal in the last few weeks, where they restricted my account. I hadn’t used it for several months and then I tried to use it and they said your account was restricted and then I had to go through all these kinds of security checks including giving them my credit card number again which of course one of the reasons to use PayPal is to avoid using the credit card. . . . My reaction to that was that I didn’t go ahead with that at that time. I went off and I did*
some googling and I found out various people talking about this and it turned out that several other people had also been suspicious about this kind of activity. . . . I eventually got some confidence that this was a normal PayPal procedure and eventually decided to go ahead.

Our participants have provided insight into what types of information they wish to protect, and shown that they typically need three levels of security depending on the activity and the context. Also applicable may be the realization that for Internet interaction which requires the highest level of security, for exam-
ple internet banking, some participants are employing self-protection techniques resistant to change such as using operating systems on non-re-writable media. Such websites that are subsequently adjusted to use the latest technology for aesthetic reasons may render these CDs and their browsers obsolete, which undermines the user’s protection mechanism.

3.6 Limitations

While the studies were quite useful to us there are some notable limitations of this study. Concerning the demographics, all participants were tertiary qualified and the data was collected in one country, Australia. There were no extremely novice users - the least amount of computer use was seven years. There were no significantly infrequent users, e.g. less than once a week, and there were no children amongst the participants.

Our participants, who are all tertiary educated, may present the best case scenario for how people go about protecting their own data. While the themes of everyone protecting themselves to the best of their ability, and using various forms of separation to provide protection seem likely to remain constant, questions regarding what the users view as worth or needing protection, seem likely to change.

Concerning the technique of users keeping a web log, the technique has the in-built issues that any self-reporting mechanism has, including changing behaviour due to the keeping of the diary, self-censoring, and no absolute guarantee of truthfulness (though all participants were quite trustworthy). This limitation was at least partially mitigated by the subsequent interviews, in which participants were queried about their web usage.

3.7 Summary

The study of tertiary qualified individuals has shown that users have identified that the current framework, that allows them to interact with the internet securely, is insufficient. In the absence of credible solutions, users are creating their own techniques aimed at enhancing their security, to provide them with the confidence to interact online. This finding was a major driver for the creation of human-followable security, as described in Chapter 7.
A significant contribution of the study is the mapping out of the ways in which people rationalize and develop their own individual approaches and techniques to protect their own security. As such, we have added a further piece to the landscape that Dourish et al. began to describe [38]. HCI design has a major role in the establishment of a consistent framework which users are comfortable enough with, so that they feel less need to create their own ad hoc solutions.

The literature is replete with research concerning the role of trust on the internet, usually the role of trust in determining if the user will interact the first time, and subsequently. Our research supports this. Further, our participants have identified questions of trust concerning the protocol and software. This is despite critical software, such as the operating system and web browser, having license agreements which state the software is provided on an “as is” basis. This basis for the creation of software clearly does not meet the users’ requirements for trust.

We have shown the web concerns continuum and the levels of safe-guarding users have for the data they identify as worthy of protection, which the HCI designers need to address and facilitate. Further work is also required in the user-identified lack of delineation between parts of the web interface which they could presume were difficult for a malicious party to manipulate, and the parts a malicious party could manipulate easily.

Finally, we provided a list of HCI design issues identified by the participants in our study. Since users need to protect themselves, HCI designers need to be aware of the methods they are using to protect themselves. Any technique employed by a HCI designer that undermines a user’s self-protection mechanism is counterproductive.
Chapter 4

Security Ceremonies

In Chapter 3 we researched human usage of security protocols and information systems in general. As a second critical foundation for the cryptographic tools we will introduce in later chapters, we investigated security ceremonies. In the main, the investigated ceremonies were already known to be flawed. So the benefit of this research was to consider the problems from a ceremony point of view, which had not be done before, and thereby to gain an understanding of what a security ceremony is.

4.1 Introduction

The best definition of a security ceremony we have created based on our understanding of the usage of the term, taking into account a target audience of cryptographers and security professionals, is protocols in their context of use. As such, an underlying protocol could be used in one instance (or context) to transport messages in an access card reader scenario. In another implementation the same underlying protocol could be used to update message boards in an airport. When the contextual information of the human interaction, the device reader, the door, etc. as well as the underlying protocol is considered, this becomes a security ceremony.

While this definition “protocols in their context of use” allows understanding of what a ceremony is, on another level, when extra steps are added to an underlying protocol, what is formed is another protocol. Yes there may now be
Chapter 4. Security Ceremonies

a human and other equipment involved, but it is a protocol and as such needs no special name, since “normal” protocols are made up of building blocks which may be sub-protocols. So to aid the reader in understanding how the term security ceremonies is used in the literature, perhaps think of security ceremonies as protocols which involve humans. As long as that description also conveys that now graphical user interfaces and other equipment and information flows, perhaps omitted in the past and thus considered out-of-band, may now also be considered part of the security analysis.

We will firstly present an analysis of three security protocols which involve humans, which are known to be broken. We analysed these known-to-be-broken security ceremonies to gain an understanding of the issues with ceremony analysis. Then, based on the information gained from our human trials outlined in Chapter 3 and the lessons learnt from ceremony analysis, we will summarise what we believe are the critical elements which are required in creating a secure human-and-computer protocol, defining the concept of POPS+ in the process. This will be the foundation from which we create our three tools to aid in the development of protocols involving humans.

4.2 Analysis of Flawed Ceremonies

Ellison wrote about security ceremonies in 2007 [41]. In this paper, he attributed the name ceremony as being coined for this purpose by Jesse Walker. Ellison provided several central ideas in a network security context, which can be directly applied to cryptographic protocols in general. The properties of a security ceremony that we distil from Ellison’s work are as follows:

- a ceremony is a superset of protocols;
- there is nothing out-of-band; and
- humans, when part of the ceremony, are explicitly included.

While theoretically nothing is out-of-band, our research has shown that in reality the scope of analysis needs to be set at some point. Bounding at any point that “makes sense” and, in particular, any point that reveals flaws previously unconsidered, is useful.
4.2.1 Ceremonies Example: HTTPS with MITM Attack

HTTPS is a protocol used on the internet to provide confidentiality and integrity to messages between two parties. An example HTTPS ceremony derived from Ellison’s paper is shown in Figure 4.1. This ceremony has a number of parts, between multiple “nodes” or parties. First, on the right hand side of Figure 4.1, is the root key distribution part of the ceremony. The nodes in this key distribution process have been denoted by CA, R and C. Here the certificate authority is represented by CA, and R represents the registration authority which involves a number of human steps between the CA and the human party C. The human C will use the key from the CA on C’s computer CC. The messages for placing the key on C’s computer CC are shown in messages 1 to 3. Notice that there is no time scale on the ceremony.

The attack is shown between the user C, and the user’s computer CC and the server S, in messages 4 to 6. The attack is carried out via two adversaries, A1 and A2. At some time after the user’s computer CC is set up ready to take part in HTTPS, adversary A2 sends a name (server S’s name) and an address (adversary A1’s address) to the computer CC. User C decides whether or not to proceed to the server based on the server’s name alone, because the software running on CC does not present both the name and address to user C, only the name.

From here, the ceremony proceeds as expected through messages 7 to 22, and hence the attack. User C’s computer, CC, securely connects to adversary A1 (messages 7 to 10) using HTTPS, adversary A1 securely connects to server S (messages 11 to 14) using HTTPS, and then the adversary A1 faithfully relays communication between the user’s computer CC and the server S. Specifically A1 passes on the login and password information, which adversary A1 now has in plaintext form for the future (note the decryption and re-encryption between messages 21 and 22 for the password, and similarly for the login). After message 22, adversary A1 is securely logged into server S, and is free to proceed as desired.

Ellison’s example ceremony presumes that only the name of the target, and not the target’s web address, is passed on to the human through the web browser in message 5, for the human to make their decision on. If this is the case, then this is clearly an issue that will result in the security of the ceremony being compromised. Some readers may suggest that this should not be the case any longer, due to such advances as extended certificates which have been introduced.
Chapter 4. Security Ceremonies

Legend:

CA Certificate Authority
R Human
C Computer
A2 Attacker Machine 1
A1 Attacker Machine 2
S Server
N Signed
E Encrypted
NE Encrypted and Signed for Verification (eg MAC)
r Random value
K Session key for "client"
V Verification value for K
K Session key for "server"
V Verification value for K
K Root Key

Figure 4.1: HTTPS Ceremony, of Ellison [41].
4.2. Analysis of Flawed Ceremonies

since 2007 (http://www.cabforum.org/). However, as described in Chapter 3 in one of our human studies we asked our participants to log their web usage security decisions for a week. We found that not one participant based any of their security decisions in a week of web use on any of the information made available by the extended certificate enhancements. Also, extended certificates are not yet mandated for use in HTTPS. Hence the issue remains current. Further, even if the address, as well as the name, is displayed to the user to base their security decision on, Ellison asks whether the human user will be provisioned ahead of time with the association between the address of the server and the name of the server, and the correctness of the name [41].

The above means that, in the ceremony shown in Figure 4.1, the user (C) believes that their computer (CC) is securely connected to the server (S). Indeed, CC is securely connected to something, just not the intended server. The point is that the HTTPS protocol is not broken, there are successful usages of the protocol between CC and A1, and between A1 and S. But the security ceremony, which includes the HTTPS protocol, is fatally flawed.

4.2.2 EMV Smart Card Security Ceremony

EMV is a suite of protocols for use in smart credit cards (credit cards with a chip). EMV was named after the initial companies involved in the creation of the standard (Europay, Mastercard and Visa). With widespread use in Europe, and now being phased in throughout the world, the companies that are currently signed up to the EMV standard are Visa, Mastercard, American Express and JCB (http://www.emvco.com/about_emvco.aspx).

A standard use of an EMV card is for the user, a client of a financial institution, to pay for a purchase in a retail outlet. The user will insert their card into the point-of-sale (POS) terminal, and then enter their personal identification number (PIN) to authorise the transaction. The chip on the smart card will verify the PIN. If the PIN is accepted, the POS terminal will contact the financial institution over a network to check if the purchase will be accepted by the institution. If the purchase is accepted, a sales receipt will be printed at the POS terminal stating the words “Verified with PIN”.

One reason for the widespread uptake of EMV has been at least partially the financial benefit to the institutions of accepting this new technology, at the customer’s cost [82]. When a transaction is disputed, where the card holder
denies making the purchase, there has been a liability shift from the merchant to the card holder. This shift has been engineered by the creators of the card system, and moves the liability by having the transactions verified with a signature to being verified with a PIN [11, 83].

The EMV ceremony from Figure 4.2 can be segmented into a number of parts. The physical process which is described by Figure 4.2 is that instead of a card holder inserting their card into a card reader at a POS terminal, the attacker inserts a man-in-the-middle (MITM) device, which would respond to the POS terminal through a card-like interface. The POS terminal would believe it is communicating with a card. The MITM device’s other end would be connected to a stolen card, with which the adversary wishes to make a purchase without having had access to the card’s PIN.

Therefore, after the MITM device is inserted in the terminal, the terminal asks the user for a PIN. The PIN is sent from the terminal to what it sees as a card, but is in fact the MITM device (message 1). Regardless of what PIN

Figure 4.2: Murdoch et al.’s EMV Ceremony [83].
is entered, the MITM device simply responds with a card verification message that would result from the correct PIN having been sent to the card (message 2). The terminal then asks the card to generate an authorisation request cryptogram (ARQC), which contains the terminal verification results (TVR). This process ordinarily simply encrypts the TVR, some issuer application data (IAD), a message authentication code (MAC) and a description of the transaction which would be later forwarded on to the bank. The issue is that the TVR states only that the card holder verification was a success, not how the card holder was verified, so a card holder verification technique of no verification (as is allowed by unmanned terminals with no PIN acceptance capability, such as parking machines) may have been used.

The ARQC containing the TVR is sent on to the card through the MITM device (messages 3 and 4). The card accepts that the transaction has been verified, and creates the ARQC. From this point on, the ceremony proceeds as per the ideal case, with the MITM device faithfully passing the messages on in both directions. The ARQC is sent to the bank, and the bank creates an authorisation response code (ARC) based on the contents of the ARQC, and sends the ARC back to the terminal (messages 5 to 8). Considering the ARQC says that the card holder has been verified, the bank will have no reason to reject the purchase (unless the card is already overdrawn or similar), and hence the ARC will state that the transaction has been approved.

The final part of the process is still the ceremony proceeding as normal. The terminal receives the ARC, and creates an authorisation response cryptogram (ARPC) which is typically a MAC over ARQC and ARC, which will allow the card to verify that the ARC corresponds to the ARQC that the card created (messages 9 and 10). The card then generates a transaction cryptogram (TC) signifying that it is authorising the transaction to proceed, and this TC is sent back to the bank via the terminal (terminal and bank keeping a copy - messages 13 to 15). At this point the terminal prints two copies of a receipt with the words “Verified with PIN” on it. One copy is given to the customer (in this case the adversary) and the other copy kept by the store.

The central reason the MITM attack works in the EMV ceremony is because the TVR message, which the terminal sends to the card to encrypt, contains only the fact that the card has been verified. The TVR does not show which method was used to verify the card. EMV cards have a number of methods of
verification, suitable for the range of uses that credit cards have today [83]. This ambiguous TVR message is encrypted and sent on to the bank (see messages 3 to 7 in Figure 4.2).

This means that the MITM device can simply prevent the PIN comparison message sent from the terminal to the card from reaching the card. The MITM device sends back to the terminal the equivalent of PIN verified without passing the PIN attempt on to the card. The ceremony then proceeds as if the card holder has been verified by a technique other than the PIN, while the terminal is left in a state which believes that the user verification has been completed with a PIN. Added to the significance of the traditional MITM attack, is the final step in the ceremony depicted in Figure 4.2 (step 16). The receipt generated by the terminal and given to the human has the words on it “Verified with PIN”.

4.2.3 Opera Mini Ceremony

Opera Software ASA is a company which develops a suite of multi-platform web browsing software programs (http://www.opera.com/company/). For the three years from August 2009 to August 2012, Opera had the greatest market share of any mobile web browser in the world 1. There are different versions of Opera web browsers for different purposes. The three main variations of the browser being:

- standard Opera for PC/Mac
- Opera Mini for mobile telephones
- Opera Mobile for devices such as PDAs

Opera Mini Design

Opera Mini is the version for devices such as mobile telephones, which have restricted computing power and resources. Opera Mini has no full rendering engine on the device. Instead, Opera has proprietary servers which handle the internet requests made on the mobile.

This process of sending requests to the internet via a server which handles the rendering and compresses the data before sending the resulting page back to the mobile telephone, has benefits both in a reduction of the computing power

---

required on the device, and also reduced bandwidth requirements to the device which is running Opera Mini. The issue from a security point of view is that there is no end-to-end security. The requests from the mobile telephone to Opera’s server are encrypted using Opera’s proprietary encryption, but the messages are decrypted from Opera’s proprietary encryption at the Opera server, and then the data is re-encrypted using standard HTTPS and the certificate of the actual target website (http://www.opera.com/help/mini/faq). As the Opera Mini FAQ on security reads:

“Opera Mini uses a transcoder server to translate HTML/CSS/JavaScript into a more compact format. It will also shrink any images to fit the screen of your handset. This translation step makes Opera Mini fast, small, and also very cheap to use. To be able to do this translation, the Opera Mini server needs to have access to the unencrypted version of the webpage. Therefore no end-to-end encryption between the client and the remote web server is possible.

If you need full end-to-end encryption, you should use a full web browser...” (http://www.opera.com/help/mini/faq#security accessed 18/05/2013)

### 4.2.4 Opera Mini Ceremony Analysis

Opera mini’s use in a mobile phone is a quintessential security ceremony. There is one protocol between Opera’s server and the internet, another protocol between the mobile telephone and Opera’s server, and finally there is a human user making security decisions based on what they see on the browser on their mobile telephone.

Of particular interest in the Opera Mini ceremony is the use of standard icons to indicate security to the user. In, for example, Internet Explorer, which almost one in three desktop users currently use worldwide\(^2\), the use of the padlock symbol means that the connection between the user and the website the user is interacting with is secure via use of HTTPS. By secure, we mean that confidentiality and integrity are assured such that no computer on the path between the user and the website can decrypt any of the information or change the message.

\(^2\)http://gs.statcounter.com/#browser-ww-monthly-201207-201306
Figure 4.3: Opera Mini Secure Connection - viewing NAB’s secure logon page.

that is sent by the user or the website. The padlock icon is used similarly in all other major browsers.

However, as shown in Figure 4.3, Opera Mini displays a padlock symbol (top right of picture) when there is not end-to-end security. This means that Opera Mini users, who know what the padlock symbol means in other browsers, are led to believe that they have a confidential connection to the website they are viewing, when they do not.

Figure 4.4 describes the Opera Mini ceremony. The ceremony begins with the user of a mobile telephone typing the address of their bank’s website into the Opera Mini web browser (message 1). A process similar to HTTPS then occurs between the mobile telephone and Opera’s Server (approximated by messages 2 to 5). As Opera ASA states:

The communication is protected by 256-bit RC4 and the key exchange is done by 1280-bit RSA. All hashes are created using SHA-256. These are the algorithms used by most SSL sites today. (http:
4.2. Analysis of Flawed Ceremonies

Figure 4.4: Opera Mini Ceremony.
A HTTPS connection is also formed between Opera’s server and the bank’s server (messages 6 to 9). Once this is complete, the request for the page is passed through to the bank (messages 10 and 11), and the bank replies with its customer login page (message 12). The Opera server renders this page, and sends the compressed output to the user’s mobile telephone device (message 13). On the mobile telephone, Opera Mini then displays the webpage, including the padlock symbol (message 14). The user sees the padlock symbol, and chooses whether to input their login information and password. If the user does enter their login and password (message 15), then this is sent back to the bank’s server via the Opera encrypted channel (message 16), decrypted at the Opera Server, and then re-encrypted and sent on to the bank’s server via the HTTPS encrypted channel (message 17).

As stated in Chapter 3, we investigated security decisions made by users in a week of standard web usage. We found that most users made the choice of whether or not to interact with websites that had direct financial interfaces, such as banks or online retail, based on whether or not the padlock symbol was shown [92]. Users presumed that a padlock meant that no one, apart from the website they were communicating with, could see their financial details and confidential information, such as login and password, in plaintext form. Opera’s intimation of confidentiality by the depiction of the padlock symbol is not in keeping with Opera’s statement in the Opera Mini FAQ which says “if you need full end-to-end encryption, you should use a full web browser...” (http://www.opera.com/mobile/help/faq/#security).

Recent developments have raised similar issues in many other devices such as smart phones and tablet PCs. On the smart phones, since there are no enforced standards for security of applications, app developers choose whether to display a padlock symbol or not when using SSL/HTTPS. Some apps, even from legitimate companies, displayed the padlock when SSL/HTTPS was not used [110]. Similarly to Opera Mini, other low powered devices with web browsers use a proxy server configuration. Amazon’s Silk browser uses a similar proxy server setup to Opera Mini, to speed up the browsing experience, but states that while all non-SSL connections go via Amazon’s proxy server, all SSL connections are made directly between the browser and the target website. As such, with so many possible configurations, there is no way for the average user to know if the
displayed padlock means they are communicating confidentially with the website they are connected to or not.

Interestingly, while the plaintext state of messages through the Opera Server clearly is a security issue and probably not realised by most Opera Mini users, the design has some security benefits. If the user trusts Opera Mini with all their communication with every party they communicate with on the internet, then this design of accessing the internet through a proxy provides essentially anonymous internet usage, as well as protection against various JavaScript-based malicious software (malware).

4.2.5 Lessons Learnt

By re-investigating known security flaws from a ceremony point of view, we identified a set of common flaws. This list included:

- each individual protocol remained secure, but the critical security information was not passed from one protocol to the next (see Sections 4.2.1 and 4.2.2);

- the information passed on to the human was inadequate for the human to have any chance of making a correct decision (see Sections 4.2.1, 4.2.2 and 4.2.3);

- it is clear that lessons long since learnt for protocols, such as requiring an indication of freshness, have not been transferred into security ceremony knowledge (see Section 4.2.1). This leads to many known flaws such as replay attacks. We address this freshness for protocols involving humans issue in Chapter 7.

While ceremony analysis has been demonstrated to capture known flaws, and therefore is useful, the technique is not without pitfalls. The most significant flaw is highlighted by our definition for a ceremony, stated at the start of Section 4.1, which was that security ceremonies were protocols in their context of use. This means that, even if the underlying protocols are found to be secure for a given context, they may well not be secure in even a slightly different context, leading to the situation of requiring a new ceremony analysis for the same set of protocols in each new context.
All of the ceremonies examined have been use cases, the context of use, of the underlying protocols, and therefore the first job of the ceremony analyser is to create a list of use cases to create a rigorous security proof for. Of particular concern for the ceremony analysis technique are areas where the context of use for the protocols, for a specific ceremony, do not yet exist. Ceremony analysis will therefore, by necessity, trail behind users’ use of any given system. For example, the people responsible for the security of new smart card driver licenses will only be able to analyse certain security ceremonies once users of the smart card have been interacting with (potentially previously unknown) third parties. This interaction with new third parties may be a new context, and hence a new ceremony will be created which will be able to be analysed only in retrospect. This is a significant step down from the ideals of provable security, which aims to ensure that, once a protocol is proven secure, it will be secure regardless of how it is used.

Therefore the common flaws revealed in the ceremonies analysed to date suggest these assessments should be completed on security ceremonies prior to deployment.

- Look for protocol-like deficiencies, such as outlined by Abadi and Needham [1]. Treat each constituent protocol as a node in the ceremony, and check that nonces and identification are being passed between nodes. Note that this finding drove the work presented in Chapter 7, since it meant a nonce-like device would need to be presented to a “human node” if replay attacks to the human were to be prevented.

- Ensure that key cryptographic information is being transferred between nodes in the ceremony.

- If the ceremony includes a protocol including a human as part of the protocol, and if the protocol comes with a proof of security, re-examine the proof of security for the assumptions that were made concerning the human.

- Examine the human’s role in the ceremony. If the only way for the human to accomplish their goal is via a particular route through a security decision point, the human will take that route.

- Examine the human-factor considerations of the ceremony. These issues include how many items a human can remember (for example, web address...
and store name pairs, as per the HTTPS ceremony) and the prior knowledge and education required. For example, in approving the usage of a HTTPS ceremony, do humans realise that the most critical information is the address? The study we presented in Chapter 3 indicated that they did not.

4.3 Investigation of a Provably Secure Protocol

As has been presented in Chapter 2, in 2008 Gajek et al. expanded on Bellare and Rogaway’s concept of practice oriented provable security [5]. The significant enhancement that Gajek et al. made to previous security models was that they proved a protocol including a human to be secure [44]. They achieved this by adding formal actions render and recognise to a security model. Render is the process of a web browser rendering a HTML page, based on the browser’s state, and presenting that page to the user. Recognise is the process of a user viewing the webpage, judging if the Human-Perceptible Authenticator (HPA) is correct, and outputting either true or false depending on the results of that test.

The protocol that Gajek et al. proved to be secure, what they called browser-based user-aware mutual authentication over TLS, is a non-trivial security ceremony. In the protocol, there is a user who has a computer, a browser running on the user’s computer, and the user is interacting with a server via their computer’s browser. Gajek et al. take the important step of extending the definition of the underlying TLS (Transport Layer Security) protocol to include the human user. A sketch of their protocol follows:

1. The protocol is between a server, a human’s computer running a web browser (which has state), and the human.

2. Before the protocol begins, the human has selected a HPA and provided that HPA to the server. The HPAs suggested by Gajek et al. are a personally selected image or voice recording.

3. Both the server and the human’s computer have authentication certificates and associated private keys, and a secure TLS connection is established between the browser and the server, when the browser on the human’s computer opens the server’s webpage. This process authenticates the server to the human’s browser and the human’s browser to the server.
4. The server sends the human the HPA that the human has stored with the server (by completing a lookup of the human’s browser-specific certificate, to know whose HPA to send), via the web browser which renders the HPA for the user, and this authenticates the server to the human.

5. Having recognised the HPA, the human sends the server their traditional login and password, thus authenticating the human to the server.

Investigation of the Gajek et al. protocol, model and proof reveals a number of salient points. These points may be categorised into HCI issues and cryptographic issues.

**HCI Issues**

For the points of interest that can be drawn from the Gajek et al. case, we will assume the HPA is an image (though these comments apply equally to voice and several other types of HPA). As stated in Section 2.2, one of the reasons protocols proven to be secure fail, when subjected to use by a human, is due to the protocol specification not extending far enough into the HCI implementation. Thus, HCI designers, who are not security professionals, are making decisions that security professionals should have made. Issues that could result from the Gajek et al. protocol include:

1. Perhaps the most significant issue is requiring the designer to ensure that at least the image is fully displayed (i.e. images have not been turned off in the browser, and the image is fully downloaded) before the login and password box is presented to the user. Otherwise, there is no authentication from the server to the human, not even potentially any authentication from the server to the human, and authentication from the server to the human is the aim of the protocol. This goes beyond the rushing user concern, which this protocol does not resist at all, since the human can enter their login and password regardless of what image, or whether an image, is sent.

2. As soon as multiple people send images to a server, design decisions will be made regarding what format to store them in, what size to store them in, and what resolution to store them in. This will be done to ensure only a fixed amount of storage is used, and that similar quality images are used. The end result will be that some images (which were too small or too low quality) may be rejected, and other images will lose significant detail.
3. Since the decisions at the client end are also not specified, different designers of website login forms will make different decisions about how to display the images. These decisions include the shape of the image (at least, portrait or landscape) and the size of the image area on the webpage, which will all impact how many HPAs are human distinguishable from the complete set of HPAs.

We have seen a variant of the Gajek et al. protocol implemented by a financial institution. In this real world example, the user does not have a certificate, and instead the user’s username is sent from the user to the bank, which the bank uses to identify which HPA to send back to the user. Upon the receipt of the HPA from the bank, the user sends their traditional login and password information to the bank.

Exploring this real world example is worthwhile to determine the sorts of design decisions that can be made by implementers of systems. Design decisions, that the creators of this login ceremony have made, include:

1. The bank’s users are presented with a set of images to choose their HPA from. That is, the bank has overcome the issues concerning the range of image sizes, shapes, formats, resolutions etc., by providing the set of images to choose from. Unfortunately, this set of images is quite small, less than 20, so the dictionary space $|W|$ of this part of the HPA is quite small.

2. The implementers have added a pass-phrase which the users submit when they select their image in the once-off setup stage. Both the image and the passphrase (two parts to this HPA) are sent from the bank to the human at each login.

3. The bank’s login proceeds without the image part of the HPA being downloaded. That is, even if the user turns off image downloads in their browser, the login and password entry fields still appear and the user can still login to the system.

4. This protocol is in no way rushing-user resistant. That is, the user can enter their login and password without looking at the HPA at all, and hence the protocol can be completed without the recognise task being executed.
Cryptographic Issues

The main cryptographic issues that surround the Gajek et al. protocol are entwined in human issues. From a cryptographic point of view, both security of the channel and authentication of the two parties is achieved by the use of HTTPS and certificates at both ends (TLS in *client authentication* mode). The reason *HPAs* are used, is due to the recognition that users do not check, know to check, know how to check, certificates. There are three central observations:

1. Essentially the server’s *password* (user’s *HPA*) is being sent to the human before the human has been authenticated. Most particularly, the separation of the human from the human’s browser-computer combination, means that while the browser has been authenticated to the server via the browser’s certificate, anyone, especially someone other than the intended user, could be sitting at the terminal. This would allow an adversary, sitting at the user’s terminal, to acquire the *HPA* and later masquerade to the user as the server.

2. Further, in the real world implementation, since the human’s browser has no certificate, then the server is sending the *HPA* without authentication at the client end, ensuring replay and MITM attacks are possible. A *man-in-the-middle* (MITM) attack is an attack where a third party intercepts messages between two communicating parties, typically without either intended party detecting this, allowing the MITM attacker to listen in and manipulate messages. The material presented in Chapter 6 can address this need for one-off *HPAs*, and Chapter 7 allows for the formal analysis of protocols to ascertain if the *HPA* being recognised by the human is fresh (mitigating replay attacks).

4.4 Application to Future Protocol Design and Security Proofs

We target two central improvements and considerations which should be included in authentication protocols involving a human. They include:

1. Rushing user resistance.

2. A security proof at a level above the cryptographic level.
4.4.1 Rushing User Resistance

Mutual authentication, for example where a bank authenticates itself to its account holder, as well as the account holder authenticating themselves to the bank, is important. In most protocols where an entity is authenticated to a human, there will be a step similar to the *recognise* function of the Gajek et al. protocol proof [44]. In this step, the entity will show *something* (a *HPA*) to the human, and the human is meant to examine this *HPA* and if it is correct they proceed, and if the *HPA* is incorrect they should abort the protocol run. Unfortunately, as we have shown, both in the research literature and in commercial implementations, quite often there is no assurance that the human has completed the *recognise* assessment - a human who skips such a step is called a *rushing user*. The term *rushing user* is used by Kumar et al. to describe a user who, in a rush, takes the shortest path through a protocol, skipping steps which are not required for subsequent steps to work [68].

To increase the chances of humans completing the recognise step, rushing user resistance should be included in the protocol. A construction that could be added to most such protocols is to send the human user not just the real *HPA* (*HPA*₁), but also a false *HPA* (*HPA*₂) in random order. Now, beyond sending to the server their user name and password, the human must also select which of the two *HPAs* was their *HPA*. If the human selects the wrong *HPA*, then the server must abort the protocol even if the login and password the human provides are correct.

There are a number of intricacies with this solution, especially when trying to combine the cryptographic elements with the human elements:

- This solution does not enhance the cryptographic security of the protocol. Rather, this step is only in place to ensure that the human follows the protocol. This element is not captured in current computational-based security proofs and models.

- Beyond not enhancing the cryptographic security, this action decreases the cryptographic security in that the adversary now has twice as many chances of sending the human a legitimate *HPA* (if only two *HPAs* are sent to the human) since two *HPAs* are now sent to the human.

- Whether the human is completing the *recognise* step is being checked by the server, in that if the wrong *HPA* is selected then the server should...
abort the protocol and force the human to start again. If the server is the adversary, then the adversary will accept the username and password regardless of which HPA the human chooses. So this training of the human to follow the protocol correctly will only work while legitimate protocol runs occur with the real server.

- The improvement to the human’s behaviour in following the protocol will happen over time. This is another concept not captured in current security proofs and models.

4.4.2 Security Proofs at a Level Above the Cryptographic Level: POPS+

Modern cryptography has matured enough, and a necessity for provably secure human-computer protocols has become critical, such that a timely paradigm shift concerning the building blocks of secure protocols is required. Just as Bellare and Rogaway defined POPS in 1993 [7,8] (see Section 2.2), thus shifting the focus to protocols and primitives in use at the time [5], we propose that there is now a requirement for a further paradigm shift, to move to a higher level of abstraction [94]. That is, to treat building blocks for human interaction protocols, such as HTTPS, as primitives, and to create security proofs based on that in the interests of creating protocols better suited to humans. In this way, we propose POPS+. The technique remains the same, as does the quality of the proof. That is, if you believe that HTTPS is secure, and a reduction can be made from the security of HTTPS to the security of a protocol, then, as long as there remains no program that can break HTTPS, the protocol will remain secure. This approach to work at a higher level and treat HTTPS as a building block has been recently formalised in the work of [62], in which they define a version of the HTTPS channel to have authenticated and confidential channel establishment (ACCE) properties. We build on this ACCE work when defining human perceptible freshness in Chapter 7.

Using the concept of POPS+, a security ceremony is, at the level that most security professionals consider security, simply a protocol which includes a human. In the same way that practice oriented provable security (POPS) of block ciphers is not proven by examining a protocol including a block cipher, the POPS+ security of a higher level cryptographic building block such as HTTPS should not
be proven by examining a protocol which includes HTTPS. The proof of HTTPS is completed elsewhere, and, once proven secure, the super-protocol which uses HTTPS is proven secure under the assumption that HTTPS is secure. In this way we have moved beyond ceremonies being protocols in their context of use to being protocols which include lower level protocols.

To allow this analysis of suitability of a protocol for human use to happen, ideal instances of the cryptographic building blocks can be used. For example, an ideal secure channel providing confidentiality, integrity, and authentication for the participant with the private key, where the other participant is known to check the certificate, would be used for a HTTPS secured channel. Cryptographers would argue that if the communication channel is secure then the proof of security of a human protocol using HTTPS becomes trivial. However, a secure channel is no guarantee that the correct information is being passed to and from the human, which is the focus of the POPS+ level of analysis. By assuming that cryptographic building blocks, such as the channel, are secure, greater attention can be focused on the protocol flows that interact with the human allowing for quicker and easier ceremony design and analysis.

4.5 Summary

We have shown that security flaws in complex systems of protocols, with human interaction, can be analysed using security ceremonies. The analysis of the EMV smart card ceremony and the Opera Mini ceremony, followed by the analysis of the TLS protocol which had been proven secure for human use, shows that a ceremony analysis is capable of capturing a greater range of security flaws than protocol analysis alone.

In the process of analysing these ceremonies, we have constructed an approach for analysing further security ceremonies. We also highlight the role that the designer plays in ensuring that the ceremony is secure. This role necessitates a grounding in security considerations, and similarly that creators of protocols are aware of typical design considerations at the human-computer interface.

This chapter has presented significant movement in what the definition of a security ceremony is. Previously, security ceremony analysis may have been considered by some to be a more complete version of protocol analysis which explicitly includes human interaction, setup steps and OOB communication, thus
proving a ceremony secure is proving the protocol secure. Such an understanding, while true, does not capture the significance or complexity of ceremony analysis. We have presented research which shows that each ceremony is a protocol in its context of use. This warns against the presumption that a protocol shown secure in one ceremony will mean that the same protocol is secure in another ceremony. The development of a list of use cases for the protocol becomes critical. Finally, we went a step further, treating the underlying protocol as a cryptographic primitive or building block, and considering the ceremony as a protocol which uses that building block protocol, such as TLS.

We have highlighted that cryptographic building blocks, such as TLS, have become mature to the point where a further level of abstraction is possible from the level that was applied when practice-oriented provable security (POPS) was promoted by Bellare and Rogaway 20 years ago. This allows, for the security proof of security ceremonies that include humans, to abstract away the cryptographic building blocks and extend the security proofs into the human-computer interface. We have called this paradigm shift POPS+. The philosophy remains the same, and that is, a reductionist proof such that the way to break the protocol is to break the cryptographic building block, and as long as the building block remains secure, the protocol remains secure. We employ this approach in the construction of a secure channel which provides human-freshness assurances, outlined in Chapter 7.
Chapter 5

Formalising Human Recognition

In Chapter 3 we described the human trials we conducted to learn what human participants in protocols do. If we are to create security proofs for protocols involving humans, we need to formalise as much of the human behaviour as possible. In Chapter 4, by investigating from an all-encompassing security ceremonies perspective protocols which involve humans and which are known to be broken, we learnt which parts of the human behaviour to focus on with this research. In this chapter we focus on formalising the human’s ability to recognise, which is a critical component in seemingly any protocol involving human authentication.

5.1 Introduction

As described in Section 2.2.1, practice oriented provable security, for authentication protocols, was introduced by Bellare and Rogaway in 1993 [7]. Since this time, many authentication protocols have been proven secure in theory, only to fail to meet this level of security when used in practice by a human. Increasingly there has been a realisation that to create secure protocols which involve humans, human capabilities need to be an explicit consideration of the security model, as exemplified by Shostack and Stewart’s statements, “...our approach to information security is flawed” and “the way forward cannot be found solely in mathematics or technology” [111].

We focus on authentication protocols involving a human, in which the human is expected to authenticate the party they are communicating with. This may be
either mutual authentication or one-way authentication. For such protocols, the human is expected to recognise an authenticator, or perform some task which accepts some security information as input and outputs either accept or reject. In this way we build on the recognise function introduced by Gajek et al. [44,45] in their work on authentication to a human in a protocol using TLS. We focus specifically on the recognise functionality a human must perform, and create a formalisation which may be reused in all such protocols.

A formal construction of the human’s recognise capabilities can be used in a variety of types of protocols. As already mentioned, there are uses such as the Gajek et al. protocol, which is similar to protocols that have been implemented by numerous financial institutions. Similarly, the construction allows for formal analysis of implementations of the widely used Verified by Visa protocol, in which a message that the human entered on first use of the system is sent back to the human to authenticate Visa to the human in all future executions of the protocol.

Less obvious real world protocols, which our construction may allow for formal analysis of, are authentication protocols over a telephone between two humans. A common example is where an investor calls her stockbroker using the telephone, says a password to the stockbroker, and the stockbroker ensures that the password said by the investor matches the password stored for that investor. Even login messages meant to allow users to compare the last time the system recorded their login credentials were used, against the last time the human remembers logging in, could be analysed for their security properties.

The formalisation captures the distinction between a targeted attack, where adversary knows the identity of the victim and may conduct research and specific social engineering attacks against that person, and a trawling or general attack [12], where the adversary has no direct knowledge of who the victim is, and therefore must rely on population-wide trends. Schechter et al. highlighted the significance of targeted attacks, when they researched the security of using personal questions as an authentication mechanism to reset passwords [107]. At one level, geographic homogeneity was a factor in allowing the successful guessing of participant answers 13% of the time within five attempts. As the attack became more targeted, success rates increased, with non-trusted and semi-trusted acquaintances being able to guess correctly 17% of the time, and trusted acquaintances being able to guess correctly 28% of the time [107]. This means that the severity and likelihood of success of an attack varies greatly, depending on
whether the attack is a targeted attack or a trawling attack.

We will show a range of uses for our formalization, with two examples of adaptations of security proofs. We will show how our approach may be applied in the case of a web-based mutual authentication protocol (see Section 5.3), and in the case of human-assisted pairing of two Bluetooth devices (see Section 5.4.1). The former will cover the case of human-selected authenticators, while the latter will provide an example of device-selected authenticators. In both cases the central human recognize step remains critical and constant.

5.2 Security Model for Human-Based Recognition

The security model describes the human’s role in recognising information sent to the human, which is a typical process in a protocol where the human authenticates a second party. The model formally describes how an attacker interacts with the human, and what capabilities and constraints the attacker has.

5.2.1 Formalisation

We begin by describing the situation where a human generates a HPA which they memorize. As described in Section 2.2.2, a HPA is a human perceptible authenticator. Subsequently, some information, $HPA'$, is generated by another party and is sent to the human, and the human assesses the $HPA'$ by comparing the received $HPA'$ to the HPA they have stored in memory. A concrete illustrative example may be the case where the human selected $HPA$ and the $HPA'$ generated by the other party are images and the human checks to see if the two images $HPA$ and $HPA'$ are equal.

A central ideal we have incorporated is that this behaviour will be different from human to human. That is, HPAs that a human generates will be human specific. We shall call HPAs that are specific to a human $HPA_H$. While in one context an example use of the HPA is that the human shares the HPA with an entity, so that the entity can use the HPA to authenticate to the human, this may not be the case all of the time as HPAs could be generated by the entity and given to a human in a setup stage. Therefore we focus exclusively on the recognize step common to most device-to-human authentication protocols. That
is, a $HPA'$ is received by the human, the human compares the $HPA'$ with their $HPA$, and either accepts or rejects that $HPA'$ is the same as $HPA$.

**HPA Scheme**

We defined $HPASpace$ to be the space of HPAs. To use the traditional example of alphanumeric passwords, for a specific protocol this space may be bounded by the 94 character possibilities, consisting of 26 lowercase, 26 upper case, 10 numerals and 32 special printable characters, and the number of characters accepted for the password. For example, for eight characters, this is $94^8$ possibilities.

The value of $HPA_H$ is the output of a probabilistic algorithm $GenHPA$ which is specific to each human, and hence accepts as inputs the human $H$ and the $HPASpace$ of the protocol being analysed,

$$HPA_H \leftarrow GenHPA(H, HPASpace).$$

To return to the example of an alphanumeric password, it is widely known that while the possibility of selecting a randomly selected eight character password may be $94^{-8}$, or approximately 53 bits of security, non-random human selection typically brings this figure closer to 30 bits of security for humans in general [19]. However, for a specific human, the set of alphanumeric passwords generated may be far smaller [19], and hence the output $HPA_H$ will be part of the human’s specific $HPASpace$ i.e.

$$HPASpace_H = \{HPA_H | HPA_H \leftarrow GenHPA(H, HPASpace)\}.$$  

Notice that $HPA_H \in HPASpace_H \subseteq HPASpace$.

The function $Recognise$ is defined to model the human’s ability to take two inputs, $HPA$ and $HPA'$, one potentially in memory and the other being presented to the human, and compare the two inputs. If, in the human’s opinion, there is a match between $HPA$ and $HPA'$ then $Recognise$ outputs a one, otherwise $Recognise$ outputs a zero. This models the human’s assessment of “The two values are the same.” Therefore, the $Recognise$ algorithm depends on the human:

$$0/1 \leftarrow Recognise(H, HPA, HPA')$$
5.2. Security Model for Human-Based Recognition

Of course, no human performs the recognise function perfectly. There will be both false positives and false negatives from the Recognise function. False positives occur when the human assesses there is a match between $HPA$ and $HPA'$ and yet the two values are different. False positives are seen as being a result of human’s inability to distinguish between two objects, if they are similar enough though not identical, and are discussed in detail below. False negatives result when $HPA = HPA'$ and yet the human assesses there is no match, and the recognise function outputs a 0. False negatives, for example the human being presented with two identical pictures and yet assessing them as different, are seen as an error condition, modelled by an error probability $\epsilon$. False negatives are in agreement with Hopper and Blum’s ($\alpha$, $\beta$, $t$) method of describing human protocols, where value $\beta$ is the probability of the human not successfully executing the protocol [59].

**Human Indistinguishability**

We denote the set of different $HPA'$s which the user recognises as being indistinguishable from their chosen $HPA$ as being the set $W_{H,HPA}$. This is the set of the actual $HPA$ and false positives. Note that a $HPA'$ that is recognised by a human as being indistinguishable from their $HPA$ may come from their human specific $HPASpace_H$ or from the general $HPASpace$. For the sake of analysis of a specific protocol, we explicitly exclude any other object in existence which is not from $HPASpace$. That is, if the $HPASpace$ of a system is defined as a four digit personal identification number, then a five digit personal identification number would not be a valid $HPA$. This commonly accepted constraint is a limitation of our approach, since there may be objects from outside $HPASpace$ which the human will accept as being their $HPA$.

To aid in the understanding of how the set of false positives $W_{H,HPA}$ interacts with $HPASpace$ and $HPASpace_H$, the relationship is depicted in Figure 5.1. These false positives form a set of HPAs, specific to a human, which are similar enough (or human indistinguishable) that the Recognise function will output a 1 for a given $HPA$:  

$$W_{H,HPA} = W'_{H,HPA} \cup W''_{H,HPA}$$

where $W'_{H,HPA}$ is the set of HPAs from $HPASpace_H$ and $W''_{H,HPA}$ is the set of
HPAs from $\text{HPASpace}$ which are not in $\text{HPASpace}_H$. Formally:

$$W'_{H,\text{HPA}} = \{\text{HPA}' \in \text{HPASpace}_H | \text{Recognise}(H, \text{HPA}_H, \text{HPA}') = 1 \text{ with } \text{HPA}_H \leftarrow \text{GenHPA}(H, \text{HPASpace})\}$$

and

$$W''_{H,\text{HPA}} = \{\text{HPA}' \in \text{HPASpace} \setminus \text{HPASpace}_H | \text{Recognise}(H, \text{HPA}_H, \text{HPA}') = 1 \text{ with } \text{HPA}_H \leftarrow \text{GenHPA}(H, \text{HPASpace})\}.$$

The relationship between $W'_{H,\text{HPA}}$ and $W''_{H,\text{HPA}}$ and $\text{HPASpace}$ is shown in Figure 5.2.

**Security and Usability**

An interesting distinction between false positives and false negatives is made when considering security and usability. False positives result in a less secure system. That is, the adversary can now produce not just the exact $\text{HPA}$, but any of potentially many $\text{HPA}'$ (i.e. $|W_{H,\text{HPA}}|$) which the human will accept as indistinguishable from $\text{HPA}$.

In contrast, false negatives do not impact security. If a human is presented with $\text{HPA}'$ which equals $\text{HPA}$ and yet still does not assess that this is a match,
5.2. Security Model for Human-Based Recognition

then the protocol will be aborted and hence the system will remain secure. However, false negatives do impact usability in that if the protocol does not successfully proceed when $HPA' = HPA$, then the human will not be able to use the system, or at the very least the human will need to execute the protocol one time more than they needed to. This is similar to the well known trade-off in biometrics.

Probabilistic versus Deterministic

In our formalism, $W_{H,HPA}$ represents the set of $HPA$'s which the user recognises (mistakenly, except in the case where $HPA' = HPA$) as being indistinguishable from their chosen $HPA$. We have captured that this will vary from human to human. We also, in Definition 5.2.2, capture that sometimes the real $HPA$ will not be recognised as being the real $HPA$ by the human. Finally, we capture that the set of HPAs that a specific human may pick will be different from human to human, in our definition of GenHPA.

However, while this variability is captured and gives useful results, the set $W_{H,HPA}$ is constant and deterministic for a human in our model, whereas in reality such a set may vary over time particularly with context. Further, the set $HPASpace_H$ is constant in our model, and again this set may vary with context.
5.2.2 Security Definition for Human Recognition

In our model, we define security in terms of the adversary $A$’s ability to obtain a $HPA$ value which will cause the human to output a 1 from the $Recognise$ function.

The security game proceeds as follows. A $HPA$ is generated for a specific human $H$ using the $GenHPA$ algorithm. This models a human selecting their $HPA$.

$$HPA_H \leftarrow GenHPA(H, HPASpace)$$ (5.1)

The adversary $A$ knows $HPASpace$ and gets oracle access to $Recognise(H, HPA_H, \cdot)$ and $GenHPA(\cdot, HPASpace)$.

$$HPA' \leftarrow A^{Recognise(H, HPA_H, \cdot), GenHPA(\cdot, HPASpace)}$$

Access to the $GenHPA$ oracle models $A$’s ability to gain information about expected $HPA$s from humans, including the target human. That is, we have not limited $A$ to only using the $GenHPA$ oracle on the target human. This allows $A$ to use the $GenHPA$ oracle to effectively construct $HPASpace'_H$, including frequency distribution, for both the target $H$ and other humans. We call this constructed $HPASpace'_H$, “$HPASpace'_H$”. This is now a targeted attack, which is more damaging than a general attack. Access to the $Recognise$ oracle allows $A$ to test $A$’s selected $HPA'$ to see if the selected $HPA'$ is accepted.

$$Recognise(H, HPA_H, HPA') = 1.$$  

Upper Bound on $A$’s Probability of Success

The adversary can either work from $HPASpace$ or generate a $HPASpace'_H$ for the human using the $GenHPA$ oracle. As shown in (5.1), $HPASpace_H$ contains at most $q_{gen}$ elements, where $q_{gen}$ corresponds to the number of $GenHPA(H, HPASpace)$ queries.

If the adversary selects from $HPASpace$ then the attack is described as a general attack, whereas if the adversary selects from $HPASpace_H$ then the attack is described as a targeted attack.

**Targeted Attack** $A$ selects an authenticator $HPA$ from $HPASpace_H$, creating a targeted attack. In this case,
5.2. Security Model for Human-Based Recognition

\[ \text{Succ}_{\text{A}_\text{human}} = \Pr[A \text{ wins}] \leq \max_{\text{HPA}^* \leftarrow \text{GenHPA}(H, \text{HPASpace})} \frac{q'|W_{H,\text{HPA}^*}|}{|\text{HPASpace}_H|} \cdot \Pr[\text{HPA}^*] \quad (5.2) \]

where \( q' \) is the number of queries to the \textit{Recognise} oracle to test a \textit{HPA} generated using queries to the \textit{GenHPA} oracle, and \( \Pr[\text{HPA}^*] \) is the probability of \( \text{HPA}^* \) being generated by \textit{GenHPA}. This is the case of the targeted attack against a human, by \( A \) somehow having knowledge of \textit{HPA} choices for that human (perhaps by knowing \textit{HPAs} used by that human on other systems).

Prior work by Gajek et al. [44] considered only non-human-specific indistinguishability, and from the entire \textit{HPASpace} (|\textit{W}_\text{HPA}|). In addition to adapting this to the human-specific targeted attack setting, the upper bound on \( A \)'s probability of success must take into account the likelihood that a \textit{HPA} is generated by \textit{GenHPA}(\( H \), \textit{HPASpace}).

Intuitively, a \textit{HPA} with a large \( |W_{H,\text{HPA}^*}| \) is unlikely to be the upper bound on \( A \)'s success probability if the likelihood that \textit{HPA}^* is picked as the authenticator is minute. This introduces the concept of the frequency of use of \textit{HPA}^*.

Therefore, for the targeted attack, informally the maximum of the frequency of use \( \Pr[\text{HPA}^*] \) combined with the size of set \( W_{H,\text{HPA}^*} \) is the upper bound on \( A \)'s success.

**Trawling Attack** \( A \) picks an authenticator from the general \textit{HPASpace}. In this case,

\[ \text{Succ}_{\text{A}_\text{human}} = \Pr[A \text{ wins}] \leq \max_{\text{HPA}^* \leftarrow \text{HPASpace}} \frac{q''|W_{H,\text{HPA}^*}|}{|\text{HPASpace}|} \quad (5.3) \]

where \( q'' \) is the number of queries to the \textit{Recognise} oracle with corresponding no prior query to the \textit{GenHPA} oracle. This is the case of the general trawling attack, with no prior knowledge regarding the human’s choices, such that the adversary has to select from the entire \textit{HPASpace}.

Usually we can expect that the targeted attack is more likely to be successful than the trawling attack, but there could be exceptional cases in which this is not true. Therefore, in general the probability from Equation (5.2) may be a more exact upper bound on the adversary’s success probability. However without access to \textit{GenHPA} the upper bound remains as in Equation (5.3). We can now
define what it means for our schemes to be secure and correct.

**Definition 5.2.1** (δ-security). We say a HPA scheme is \( \delta \)-secure, meaning that the scheme can be used as an authentication scheme for an entity to a human, if

\[
Succ_A \leq \delta.
\]

**Definition 5.2.2** (\( \epsilon \)-correctness). We say a HPA scheme is \( \epsilon \)-correct if, for all \( HPA_H \) in \( HPASpace \), where \( HPA_H \leftarrow GenHPA(H, HPASpace) \),

\[
Pr[Recognise(H, HPA_H, HPA_H) = 1] \geq 1 - \epsilon
\]

where \( \epsilon \) represents the false negative rate of correctness. For correctness, we are not concerned about false positives, which is covered by \( W_{H,HPA} \).

### 5.2.3 Analysis and Discussion

The \( HPASpace \) will be system specific. Using a classic human authenticating scenario which may be adapted to create mutual authentication, a system for banking Personal Identification Numbers (PINs) may have a \( HPASpace \) limited to four numerical digits, while other systems may have graphical or alphanumeric \( HPASpaces \). The effect of \( GenHPA \) taking as an input \( HPASpace \) is that comparison between security results for different systems can be made.

Giving the adversary \( A \) oracle access to the human specific \( GenHPA \), allows the modelling of the effect of HPA reuse and of the preferences of the user. If the size of \( HPASpace_H \), the output of \( GenHPA \), is less than the size of \( HPASpace \), then the adversary receives an advantage. There may be instances where \( A \) does not get this capability, depending on whether the attack is a targeted or trawling attack. A targeted attack is by far the stronger and more damaging attack, as in the real world this would model the case where an adversary has knowledge of a human’s HPA choices. This knowledge may exist because the adversary may be a legitimate server where the human logs in elsewhere, and thus the adversary has seen many prior examples of the human’s HPA choices.

In general, ensuring that \( GenHPA \) and \( Recognise \) are human specific functions, allows for modelling of targeted attacks at a specific human. Giving \( A \) oracle access \( Recognise(H, HPA_H, \cdot) \) to the human specific \( Recognise \) function means that an adversary with infinite resources could create the set \( W_{H,HPA} \) for a given \( HPA \).
5.3 Human-Specific HPAGen

In this section we will describe the case where a human chooses the HPA. In the next section, we will cover the alternative case where a device selects the HPA for the human.

Our formalisation, as defined allowing for targeted attacks, is ideal for use in existing device-to-human (D2H) authentication scenarios, such as the protocol by Gajek et al. [44], or protocols involving authentication by humans in general.

5.3.1 Gajek et al. Browser Based Mutual Authentication over TLS

As discussed in Section 2.2.2, Gajek et al. have created a mutual authentication protocol including a human and a HPA. A sketch of the Gajek et al. protocol, including a description of where the HPA is used and how the human recognise function is applied, follows [44,45]:

1. The protocol is between a server, a human’s computer running a web browser (which has state), and the human.

2. Before the protocol begins, the human has selected a HPA and provided that HPA to the server. The HPAs suggested by Gajek et al. are a personally selected image or voice recording.

3. Both the server and the human’s computer have authentication certificates and associated private keys, and a secure TLS connection is established between the browser and the server, when the browser on the human’s computer opens the server’s webpage. This process authenticates the server to the human’s browser and the human’s browser to the server.

4. The server sends the human the HPA that the human has stored with the server (by completing a lookup of the human’s browser-specific certificate, to know whose HPA to send), via the web browser which renders the HPA for the user. This step authenticates the server to the human if the human recognises the HPA.

5. Having recognised the HPA, the human sends the server their traditional login and password. This step authenticates the human to the server.
Our formalisation makes proof of such a protocol more complete by replacing game 20 of their proof [45]. The sketch of their proof is that SSL is proven in games 0 to 19, and in game 20 the ability of the adversary to guess a HPA that the human will recognise is considered. Further, the analysis presented [45] could be simplified since the initialisation stage can now be explicitly comprehended as running the GenHPA algorithm, and the process of recognition realised as an invocation of the Recognise function. Including the concepts of Gajek et al.’s specific proof, the result of this game would become:

\[ |\Pr[Win_{20}] - \Pr[Win_{19}]| \leq Succ_{A_{\text{Human}}} \]

where \( Succ_{A_{\text{Human}}} \) is defined by either Equation 5.2 or Equation 5.3. This is in contrast to the corresponding equation in the original proof [45], using our notation:

\[ |\Pr[Win_{20}] - \Pr[Win_{19}]| \leq \frac{q |W|}{|\text{HPA\text{Space}}|} \]

where \( q \) is the number of executions of the protocol.

The above demonstrates how our formalism can be used in a certificate or SSH-based key exchange protocol. Our proof goes beyond the Gajek et al. proof in the following areas:

1. In the Gajek et al. proof, \( A \) selects from all of \( \text{HPA\text{Space}} \), roughly equivalent to our trawling attack, whereas our proof allows for a targeted attack where \( A \) selects from the human-specific \( \text{HPA\text{Space}}_H \).

2. Our model covers the concept of frequency of use of a HPA, not just the size of \( |W| \).

Furthermore, the technique used here can be applied to any authentication protocol, password authenticated key exchange (PAKE) protocol or key agreement protocol which requires the human to authenticate a message in the protocol. While these examples are network based, there are examples of use involving just the human and a device, such as a trusted computing scenario where a computer’s trusted platform module (TPM) could be used to securely assess the computer and securely present a HPA to the user.

An example trusted computer scenario may involve a login procedure where a picture is constructed by the trusted module for the human. That is, the TPM creates a list of hashes from different stages of a computer’s boot sequence,
and these hashes are graphically presented to the user in some way. Thus, the computer’s TPM could be used to securely assess the computer and securely present a *HPA* to the user where the *HPA*’s construction (rendering by the TPM) depends on the status of the computer. As long as nothing has changed on the computer, then the same *HPA* will always be shown to the user to recognise each time she logs in, otherwise a completely new *HPA* will be sent to the user (i.e. in the same way that a one bit change will create a completely new HASH value).

So a possible concrete implementation may be a 64 pixel black and white picture, 8 pixels high by 8 pixels wide, so the size of *HPASpace* would be $2^{64}$. *GenHPA* would be generated by the trusted platform module to generate *HPAs* uniformly over *HPASpace*. A typical $W_{H,HPA}$ may consist of any picture with a similar number of white (or black) pixels as *HPA*. Since the *HPAs* are device controlled, $\mathcal{A}$’s advantage would be limited by Equation 5.3. Modelling the login procedure in this way would allow adjustment of security parameters such as the number of pixels and the number of colours in the image constructed by the TPM and presented to the user when they login, to arrive at the desired security for the login procedure.

### 5.3.2 Human-based Recognition with Non-human Controlled Authenticators

This leads us to the many practical protocols in use where *HPAs* have been chosen by devices or the system, rather than by the human. From a usability perspective, a human may be able to better remember a *HPA* they have selected. However, the essence of a D2H authentication protocol, where a human recognises some information (*HPA*) sent by the device, is that the *HPA* has been previously agreed on by the human and the device. Whether a human registers a *HPA* at a bank, or whether the bank sends the human a *HPA* in a setup stage, is irrelevant from the perspective of whether the protocol will function. In either case, the *HPA* can be sent to the human by the bank in all future protocol runs to authenticate the bank to the human.

Having the *HPA* chosen by the device means that *HPASpace*$_{H}$, the subset of *HPASpace* which the *HPA* will be in, encompasses all of *HPASpace*. This is because the device will choose the *HPA* from the entire *HPASpace* by a probability distribution, presumably a random distribution. From a security perspective, this maximises the space that the adversary has to select from to acquire a $HPA'$.
which the user will recognise as their HPA. Having the HPA chosen by something other than the human, shifts the upper bound available to the adversary from a more powerful targeted attack to a weaker trawling attack.

We now adapt our formalisation towards non-human specific HPA selection, where non-targeted general trawling attacks apply. We focus on more general cases, such as human-assisted pairing protocols.

5.4 Non-Human-Specific HPAGen

Humans are often called on to play a part in protocols for the authentication of devices. For example, an authenticated key agreement (AKA) protocol employing short authenticated strings (SAS) may be used to manually pair wireless devices by having the user check matching values on each device [86].

In such device pairing protocols, two HPAs are sent to a human, and if the human recognises the two HPAs as matching and accepts, then the human takes an action and the devices become paired. As always, the HPAs can be any human perceptible authenticator, such as two series of sounds, two series of flashing lights, or text or images displayed on a screen. We formalise this process by setting the HPA variable to the output of a probabilistic algorithm $GenHPA$ which selects from the HPASpace of the protocol being analysed, i.e.

$$HPA \leftarrow GenHPA(HPASpace)$$

Note the removal of the human from the $GenHPA$ step, and hence the space of the generated HPA is $HPASpace$ not $HPASpace_H$ as it has been in the human generated HPA case. The $Recognise$ function remains human specific, and the adversary’s win condition remains:

$$Recognise(H, HPA, HPA') = 1$$

Since $HPASpace_H$ is now the size of $HPASpace$ and hence the upper bound is as for a general trawling attack, the bound on $\mathcal{A}$’s probability of success for a
device chosen $HPA$ is

$$\text{Succ}_{\text{Device}} = \Pr[A \text{ wins}] \leq \max_{HPA^* \leftarrow \text{GenHPA}(\text{HPASpace})} \left| \frac{q_{W_{H,HPA^*}}}{|\text{HPASpace}|} \cdot \Pr[HPA^*] \right|.$$  

### 5.4.1 Human-based Recognition with Two Devices

An example implementation, which could now be rigorously examined using our formalisation, is pairing protocols for Bluetooth devices. When using human-assisted pairing of devices using Bluetooth, one method is for protocols based on short authenticated strings (SAS). In these protocols, the SAS are somehow represented on two devices for a human to compare, using for example audible tones or flashing lights. Now these protocols involving a human can be examined formally using our model.

![Figure 5.3: Pasini and Vaudenay’s SAS Protocol [86]](image)

We will demonstrate how to incorporate our formalism by providing a security
proof of Prasad and Saxena’s human-assisted Bluetooth pairing protocol [91] which is based on Pasini and Vaudenay’s SAS protocol shown in Figure 5.3 [86]. Pasini and Vaudenay’s protocol is designed to allow for the authentication of a potentially large message, by comparing two short strings. Note that by ensuring the messages came from a specific party, i.e. authentication, this also ensures integrity has been maintained. That is, if a third party changes the message, thus breaking integrity, then the message no longer originates from the initial party that sent the message. So the intention of the Pasini and Vaudenay protocol is to have no confidentiality of the messages sent, but have assurances of integrity and know who has sent the messages.

In Pasini and Vaudenay’s protocol, there is a critical final step over an authenticating channel. Using Vaudenay’s definition of this channel from [119], “The authentication channels provide to the recipient of a message the insurance on whom sent it as is. In particular the adversary cannot modify it (i.e. integrity is implicitly protected). . . . (the) channels are not assumed to provide confidentiality.” The SAS protocol in Figure 5.3 employs a commitment scheme (commit, open) and a keyed hash function. An overview of the protocol is:

1. Alice selects a string $K$ of length $\kappa$ at random.

2. Alice commits to the value $K$ and their message $m_A$. This commitment is $c$. The commitment is such that Alice cannot later manipulate the value of $K$ having seen what value Bob later sends.

3. Alice sends $c$ and $m_A$ to Bob.

4. Bob then picks $R$ at random and sends $R$ and their message $m_B$ to Alice.

5. Alice sends $d$, used to open the commitment scheme, to Bob.

6. Bob uses $m_A$, $c$ and $d$ to create (or “open via the commitment scheme”) $K$. The hats on the values (shown in Figure 5.3), that is, comparing $c$ with $\hat{c}$, indicates that the value has passed over an insecure channel. Thus the value at Bob, $\hat{c}$, may no longer be the original $c$ created by Alice.

7. Finally, both parties create a string, $SAS$, using a keyed hash function. Both parties XOR ($\oplus$) $R$ with the output of a hash, keyed with $K$, of the value $m_B$. 

Informally, to give an intuitive understanding of the benefits of the protocol, note that by both parties XORing $R$ with the output of a hash of the value $m_B$, with the hash keyed with $K$, and by comparing the SAS values the two parties create, if the SAS values are equal then the following values must have been the same:

- **$K$** Otherwise the hashes would output different values.
- **$R$** Otherwise the result of the XOR would be different. Note the use of the commitment scheme which generates $K$ and the order of the protocol ensures $R$ is sent and known before the output of the hash can be computed.
- **$m_B$** Otherwise the hashes would output different values.
- **$m_A$** Otherwise the opening of the commitment at Bob would have resulted in a different $K$ at Bob, which would have meant the keyed hashes would have output different values.

Thus, if the two SAS values are the same, then the messages $m_A$ and $m_B$ have been successfully sent between the parties without being changed by a third party.

Prasad and Saxena’s adapted protocol exchanges and authenticates public keys by instantiating $m_A$ from Figure 5.3 as $pk_A$, and, critically, the human makes the authentication assessment thereby taking the role of Pasini and Vaudenay’s authenticating channel. Thus, by comparing two short strings, the (far longer) public keys are authenticated. Note that the protocol of Pasini and Vaudenay’s is well suited for exchanging public keys, as the keys are public but there must be assurances that the keys belong to the original senders, as is the goal of the Pasini and Vaudenay protocol.

The authentication assessment is made possible by transforming the constructed SAS string into a series of audible tones (beeps) or flashing lights (blinks) for the human to compare and, if the assessment is that the devices are the same, to accept and pair the devices. We shall call this protocol the *Beep-Blink* protocol. The adapted authentication stage of the protocol [91] is shown in Figure 5.4.

Prasad and Saxena [91] state, “The security of our scheme is equivalent to the security of the underlying SAS protocol under the assumption that the user does not commit any errors.” This leaves open to what extent human errors
Chapter 5. Formalising Human Recognition

Figure 5.4: Prasad and Saxena’s human-assisted device pairing protocol based on SAS protocol [91]

may impact the security of the protocol and how these can be dealt with in the security analysis. We have shown that it is necessary to go beyond the presumption of humans acting perfectly. Indeed, their human study [91] clearly showed firstly that the humans did not act perfectly, and secondly that the level of error depended on the type of HPA used.

An interesting philosophical point is that even 1s and 0s for voltage levels in a circuit have already undertaken a transform to be human recognisable. So the SASs of Pasini and Vaudenay could be considered HPAs, and the subsequent lights by Prasad and Saxena are also, though different, HPAs from different HPASpaces. In this way, the SAS-family of protocols, or the transformed SAS into beeps and blinks, can be seen as GenHPA algorithms, the probability distribution of which may be very well known, which output a HPA in the form of a visual or audible SAS.

Instead of the human being pre-loaded with a HPA and being sent a HPA′ as the two HPA inputs to the Recognise function, HPA_A will be on device A, and HPA_B will be on device B. The formalisation for the human specific version of Recognise, including the concept of \(W_{H,HPA}\) as the set of HPAs which are indistinguishable for that human, still apply. While the intention of the Prasad and Saxena protocol would seem to be that the human either accepts on both devices or rejects on both devices [91], there clearly exists the case where on one device the human selects “Accept” and on the other device the human selects “Reject”. As such, for device A, their (potentially transformed) SAS is held as
HPA and the SAS supplied by device B will be $HPA'$; and similarly for device B. The authentication stage of Prasad and Saxena’s protocol, using our formalism, is shown in Figure 5.5. We now use our contribution to present a proof of the Prasad and Saxena protocol, capturing some useful human considerations.

![Diagram](Figure 5.5: Prasad and Saxena’s human-assisted device pairing protocol considered using our formalism)

Prasad and Saxena’s protocol [91] may be seen as an instantiation of Pasini and Vaudenay’s protocol [86], with the human in Prasad and Saxena’s protocol playing the role of Pasini and Vaudenay’s magical authentication channel. The protocol outlined in Figure 5.5 is a more concrete description of how the protocol of Prasad and Saxena could be implemented. Beyond capturing the inability of humans to perfectly execute a protocol, and hence that the security of Prasad and Saxena’s instantiation is not at all equivalent to that of Pasini and Vaudenay, our formalisation allows for capturing and analysis of a far more realistic set of issues. For example, using the concrete protocol outlined in Figure 5.5, there is no reason why there could not be two potentially spatially separated humans doing the comparison (one for device A and one for device B) rather than one human in the one location; and there is no reason why the protocol cannot end with device A believing it is paired with device B, without device B believing it is
paired with device A (and vice versa). This lack of equality between the pairing status of the devices may be through either Recognise failing or else through time delays in the human making a choice.

5.4.2 Proof of human-assisted device pairing protocol

A Bellare-Rogaway 1993 based model [7] was used to provide a security proof for Pasini and Vaudenay’s SAS protocol [86], upon which Prasad and Saxena based further human centred protocols [91]. However, in Prasad and Saxena’s work, no formal security analysis could be given to the BEEP-BEEP, BLINK-BLINK and BEEP-BLINK variations of the underlying SAS protocol. Rather, the claim was made that the derived protocols had equivalent security to the underlying SAS protocol [86] under the assumption that the human did not commit any errors [91].

Now, with our formalisation, the security framework exists to formally analyse such protocols.

Adversarial model

Recall the model for short authentication string-based pairing security, described by Pasini and Vaudenay [86] based on Bellare and Rogaway’s model [7]. An outline of Pasini and Vaudenay’s model is described here for completeness:

**Launch** \((n, \text{role}, x)\) launches a new protocol instance on node \(n\) playing \(\text{role}\) (e.g. either Alice or Bob) with input \(x\). It returns a new instance tag \(\pi_n^i\).

**Send** \((\pi, y)\) sends an incoming message \(y\) to the instance \(\pi\). It returns an outgoing message \(z\), or the final output of the protocol if it completed.

**Corrupt** \((n)\) injects a malicious code in node \(n\) so that its behaviour is no longer guaranteed.

These queries are standard in cryptographic models. For example, the **Send** query allows the adversary to run the protocol normally and to inject messages of his choice, reflecting the assumption that the adversary controls communications between protocol participants. The **Launch** oracle creates a unique tag \(\pi_n^i\), which allows node \(n\) to have multiple protocol instances running. **Corrupt** allows the adversary to effectively take over a node, meaning that any code the adversary wishes could be injected at the corrupted node such that the node would do what the adversary wants.
In the model, the participants $ID_n$ are located at nodes in the network. In Pasini and Vaudenay’s proof for their protocol [86] the winning condition for the adversary in such a message cross authentication protocol is “if some instance ended on an uncorrupted node with a pair $(m, ID)$ but no instance on the node of identity $ID$ with input $m$ was launched.” For a complete description of the model, see [86].

**Theorem 5.4.1** (Success probability of Beep-Blink protocol). Let $\zeta$ be $\mathcal{A}$’s success probability in the SAS protocol of Pasini and Vaudenay [86], then $\mathcal{A}$’s success probability in the Beep-Blink protocol of Prasad and Saxena [91] is bounded by $\zeta + \text{Succ}_{\mathcal{A}\text{ Device}}$.

**Proof.** We employ Shoup’s game hopping proof technique [112] augmented by Dent [32]. We employ a sequence of two games, the first game being the security game for the protocol shown in Figure 5.5 which is the human protocol of Prasad and Saxena represented with our formalism. We transform this to the security game for the original protocol of Pasini and Vaudenay shown in Figure 5.3 [86], bounding the adversary’s advantage between the two. As such our proof augments the proof of Pasini and Vaudenay to cater for the human considerations of the Prasad and Saxena protocol. We denote $\text{Win}_i$ as the probability of the adversary winning game $i$.

**Game $G_0$** describes the real protocol, as run by Prasad and Saxena [91], using our formalism (see Figure 5.5 and Figure 5.3). The game is played between a probabilistic polynomial time (PPT) bound adversary $\mathcal{A}$ and a simulator. The simulator simulates protocol participants as specified in the natural protocol specification, and answers all of $\mathcal{A}$’s queries.

**Game $G_1$** describes a game which is the same as Pasini and Vaudenay’s SAS protocol [86]. The difference between Game $G_1$ and Game $G_0$ is that in Game $G_1$, the original SAS protocol, the authentication comparison is based on equality, whereas in game Game $G_0$ the authentication comparison is based on our human formalism. Hence, remembering $\text{Succ}_{\mathcal{A}\text{ Device}}$ defined above,

$$|\Pr[\text{Win}_1] - \Pr[\text{Win}_0]| \leq \text{Succ}_{\mathcal{A}\text{ Device}}.$$

Therefore,

$$\Pr[\text{Win}_0] \leq \text{Succ}_{\mathcal{A}\text{ Device}} + \zeta.$$
Pasini and Vaudenay [86] describe a win condition of at most $2^{-\rho}$ plus the adversary’s advantage against the hash function, where $\rho$ is the number of bits of the SAS. They approximate $\rho$ by $\log_2 \frac{N^2R^2}{2p}$, where $N$ is the number of participants, $R$ is the number of runs of the protocol, and $p$ is the attack probability, with the example given for ATM-like PIN numbers of $p = 3 \cdot 10^{-4}$. Our formalism gives more meaningful values for $p$, taking into account human imperfection, and allows for the comparison of representational transform techniques (such as beeps and blinks). This means a more accurate security parameter of $\rho$ could be calculated.

In this way, a general framework may be created for all such unauthenticated key exchange protocols followed by HPA recognition.

5.5 Summary

This work presents a method of accumulating data which will allow for the comparison of schemes in which the human will need to recognise some data. When represented formally using our technique, schemes can be compared using: the size of $\text{HPASpace}$ (maximise), $\text{HPASpace}_H$ (maximise) and the ratio between the two (bring to equality); the size of $W_{H,\text{HPA}}$ (false positives, minimise); the size of $\epsilon$ (false negatives, minimise); and the frequency of use distribution (normalise). In Chapter 7 we will use the Recognise formalism presented in this chapter.
In Section 2.4 we explained that in standard authentication protocols, an assurance of freshness is looked for to mitigate the risk of replay attacks. This need for freshness and the attack it prevents has been identified for a long time. The protocol presented in Figure 2.1 was created in 1978. However, for protocols involving humans, it has been the accepted norm to have no such assurance of freshness, meaning that replay attacks are possible. Previously there has been no technique to provide the equivalence of a nonce for human users, to provide the human user with an assurance of freshness. We now present a tool to create nonces for human use.

6.1 Introduction

A common attack on protocols, particularly authentication protocols, is the replay attack. In this attack, messages sent between participants in the protocol are captured by the adversary and stored, to be re-sent at a later time to the participants. Because the messages recorded are legitimate messages, even if the messages are encrypted this replaying attack will work unless well-known countermeasures are taken. The common countermeasures to prevent a replay attack are the inclusion of a random nonce, which is a random number used only once, a timestamp, or a counter.

The replay attack and its countermeasures are well known and understood, such that commonly used security protocols between computers, such as HTTPS,
use a random nonce. Nevertheless, such measures are not in place for protocols involving a human at the human-perceivable level. Prior to this thesis, there has not been the concept of a human-created random nonce. Indeed, the most common human authentication protocols used on the Internet, such as to log into webmail, banking institutions, or social media, involve the same messages being sent every time. While underlying protocols may ensure that the messages sent between computers are unique, rendering the replay attack ineffective, this is neither visible to the human nor does the human play any part in the decision making process. The lack of a direct role for the human to play in these protocols represents a loss of control and loss of power to them and requires blind trust that the protocol used protects their privacy and security. We have filled this gap with the Computer-HUman Recognisable Nonce (CHURN).

As stated in Chapter 3, for users without a security background, mechanisms such as extended validation certificates, HTTPS, and a padlock, that is, indicators of security, are ineffective because the user asks “Who decides?” That is, who decides whether the padlock symbol is shown? Many security researchers have created methods in an attempt to address this issue. They write about education, ensuring the human’s mental model is similar to the program, and creating better security indicators [2,78,92,96].

Discussing the advantages and disadvantages of education, mental models, and enhanced security indicators is beyond the scope of this thesis; however, a useful example to put the security researchers in the position of the non-security-academic is published by Shin et al. [110]. They analysed 212 smartphone applications, of particular interest being the top 30 apps in the finance category. Their findings show that even in this finance category, not all of the 22 apps that had login pages supported the SSL protocol commonly used to secure communication in a web browser setting. For smartphone apps, there is no controlling program such as a web browser that decides whether or not to display security indicators such as a padlock. While some of the analysed smartphone apps did display a padlock, this is simply a person (the app designer) deciding to put a picture of a padlock on their app, and was no guarantee of security. This means that the security academic smartphone users are in no better position than smartphone users in the general public. The question of “Who decides?” for whether security indicators, such as the padlock symbol, are displayed on smartphones has an answer which requires blind trust and a total
loss of control. A CHURN is an important building block in creating protocols where the security properties are perceptible.

The goal of our research is to empower the human by giving more control to the human, while providing the human user with protection against replay attacks using methods that have been used extensively for many years in computer protocols. We do this via the creation of a random nonce that the human shares control in the creation of, called a CHURN.

6.1.1 Example Use Case

An example of the use of a human nonce would be in the Verified-by-Visa protocol. In standard implementations of this protocol, a user would initially give to Visa a personal message or passphrase. This message will be used by Visa to authenticate Visa back to the human in all future protocol runs. Unfortunately, this phrase is kept constant. As Dhamija and Tygar write, “The most obvious weakness of this scheme is that the bank must display the shared secret in order to authenticate itself to the user. If the secret is observed or captured, the image can be replayed until the user notices and changes it [34].”

To make the Verified-by-Visa protocol resilient against such replay attacks, our human nonce generator could be used to create a nonce for the user on each protocol run. An example protocol may include, similar to the establishment of the personal message in the existing protocol, the nonce the human has picked being sent to Visa via an out-of-band channel such as via their bank’s website; and then Visa could send this current nonce back to the human via their web login which would both provide protection against replay attacks and implicitly authenticate Visa to the human. In this way, the human would gain some control, by both selecting the nonce and by seeing their nonce come back to them. The human will be able to see and understand the security of the protocol. The human would still log in via their password to authenticate.

6.1.2 CHURN Overview

Previous limitations regarding the process of humans picking their own nonce revolve around the inability of humans to act randomly. We combine a computer random number generator with human choices to create a random sequence that the human has shared control over. We do this using the computer to pick
characters in a random manner, and present sets of four choices to the human for the human to pick from. The human selects one character from the set of four characters and this will be the first character of their CHURN. Next, the human is presented with another set of four characters, and makes another choice, and this becomes the second character of their CHURN, and so on up to the size of the CHURN.

A CHURN does not need to be remembered (as per a password), only compared and recognised (as per a standard nonce). Thus, a CHURN can be random while a password needs to be memorable and hence typically not random. We employ various design features, such as moving the location of the choices presented to the human, to create an output which is fairly similar to a random sequence. Our trial demonstrated that the tool we have developed is a method for humans to generate a random sequence to a desired security level, and demonstrates that the humans are adding a source of randomness beyond that of the computer’s random number generator.

6.2 Background

A number of ingredients are considered when designing a CHURN generator. In Section 2.4 we described nonces and the need to use them to mitigate replay attacks. We now discuss, from a human point of view, issues of control and the ability to be random. Finally we will examine the data analysis methods that present information about the degree of randomness a specific set of information has.

6.2.1 Power, Control and Trust

In the world of design, questions of power and control are significant and a central consideration and focus. As Bratteteig and Wagner state, “All decisions are a choice between possibilities, and selecting one of them and making it concrete as a change in an artefact, is a demonstration of the capacity to transform, which is a key aspect of power [15].”

Currently, very little control or power is given to the users in security protocols. For many users blind trust is required that their information is secure and that they are communicating with the party they think they are communicating with. This lack of power and lack of control continues beyond the simplest
6.2. Background

login systems to more secure two factor security systems that employ security code generation tokens such as RSA’s SecurID. In the case of SecurID, the user presses a button which causes the token to display a “random” number that can be thought of as a one-time password that can be traced to a specific token since each token will have a different seed value used in the random number generator. The immediate user and the company employing the RSA solution must trust that the seed has not been compromised.

As designers of security systems which humans will use, we can choose to empower the users of our system. Of course, there are the usual concerns such as:

- users will make errors of omission [64];
- “user behaviour tends towards the easiest path” [22];
- ask a user to “write down 100 random decimal digits, chances are very slim that he will give a satisfactory result” [65];
- and “computations involved are far beyond the abilities of most humans” [59].

A critical benefit of making the user an empowered part of the security process is that they may trust the process more [30]. Lee et al. have shown us that specifically in the context of web-based e-commerce customers who feel higher levels of trust will revisit the site more often [71].

6.2.2 Randomness, Uncertainty and Test Methods

A seemingly fundamental part of a random nonce that allows it to be used as an indication of freshness is that the nonce is random. Perhaps more particularly, the value of the next nonce must be unpredictable, and hence uncertain. Our objective is to create a method where humans can have a decision making role, ideally leading to a feeling of ownership and empowerment, in the creation of an unpredictable sequence. We have called that unpredictable sequence a CHURN, which is to be used to provide freshness assurances in protocols involving a human.

Many standard statistical measures that indicate randomness are inappropriate for assessing the results of our experiments [29, 101]. The reasons are varied as to why the standard statistical tests are not useful in our case, but mainly
center around the need for many data samples and for the samples to be large, particularly given an alphabet size of 94. In our case we need more of an indication of how random the sequence is, rather than a true/false assessment of whether the sequence is uniformly random.

For the purposes of assessing how random the sequences generated by our experiments were, we have settled on four main tests:

1. Histogram;
2. Entropy;
3. Compressibility; and
4. Hamming Distance.

Each method will now be discussed further.

**Histogram**

A histogram, or simple count of the frequencies of each character, was used for each participant for each of the different experiments. Indications of randomness are firstly the smoothness of the histogram, with a smoother histogram indicating greater randomness. Secondly, the portion of the alphabet used is also an indication of randomness, meaning that a participant who uses 95% or more of the alphabet is seen to be more random with regard to the entire alphabet than a participant who uses 40% of the alphabet.

While providing valuable insight into the randomness and uncertainty of the created sequences, this test is limited in that a participant who sequentially cycled through the entire alphabet would be shown to be perfectly random, at least in the single character frequency setting. That is, the histogram would be ideally smooth with all characters being used the same number of times. For the purposes of our experiments, which will be described in detail in Section 6.4, we conducted histogram analysis for single characters, bigrams, trigrams, and n-grams up to 8 characters in a row which exhausted all matches. Repeating some bigrams is expected given an alphabet of 94 and a sample size of 400, while repeating four characters in a row is unexpected if the values are random.
Entropy

In 1948 Shannon gave us the concept of the entropy in the information theory discrete setting, based on Boltzman’s H theorem entropy equation [109]. Entropy is a measure of uncertainty, and increases as uncertainty is increased. Shannon’s entropy H equation is:

\[ H = -\sum_{i=1}^{n} p_i \log_2 p_i \]

Where \( n \) is the number of elements in a set, and \( p_i \) is the probability of that element appearing. For a given \( n \), \( H \) is maximized when all \( p_i \) are equal, meaning that all \( p_i = 1/n \). At this point, \( H \) will equal \( \log_2 n \). So for a 94 character alphabet, \( H_{max} \) will be 6.55459. As Shannon writes, “This is also intuitively the most uncertain situation [109],” so higher entropy values will indicate greater uncertainty.

Shannon’s entropy gives a useful measure of uncertainty when our data is analysed in the single character setting. Because each sample is 400 characters and the alphabet size is 94, there is at least a chance that all elements of the alphabet may be used and entropy may be maximized. As soon as bigrams are examined, the alphabet becomes \( 94^2 \), which is 8836. This means that our samples cannot achieve a maximum value for two-character and higher entropy, simply because every n-character element in the n-character set cannot be present in our samples of 400 characters. While entropy gives a useful single comparative value for both the number of symbols used and their relative frequencies, as per the histogram test with the sample sizes we have it is susceptible to patterns that guarantee the entire alphabet is used but which are in no way random. For example, if our sample was an exact multiple of the size of our alphabet, then maximum entropy could be achieved by repeatedly typing out the entire alphabet in sequence. All values would be equally likely and would have been used exactly the same number of times each. Use of such non-random, though entropy maximising, patterns may be detected when higher order entropy values were compared.

Compressibility

In general, compression algorithms work by removing statistical redundancy in the data to be compressed. As Ziv and Lempel write, “Once the relevant source parameters have been identified, the problem reduces to one of minimum-
redundancy coding [127].” A data file which is more compressible is more predictable and less uncertain, which indicates that the source of the data was less random. Since various compression algorithms look for patterns and repeated sequences, this mitigates the use of non-random patterns not being detected by histogram and entropy analysis.

To give insight into how compression tools work, an example compression tool may be thought of as having three parts: a model structure for contexts and events, a statistics unit for estimation of the event statistics, and an encoder for the events [70]. This is sometimes summarized to a model and an encoder. Improvements to the model lead to better compression effectiveness, while improvements to the encoder tend to lead to better compression efficiency [80].

We picked three compression tools to illustrate the uncertainty of the output from our trials. Firstly, an arithmetic encoder by Moffat, Neal and Witten [80] was used. Arithmetic encoders have history that started with Shannon, and ideally approximate entropy [70]. This means that if the single character entropy of a file was 5.8 of a maximum 6.5, then the file may compress down to 89.2% of its original size, if the model used was a single character model. We used a single character model.

Our second compression tool was gzip. Gzip is a widely used and accepted compression tool. Gzip employs a variation of the LZ77 algorithm [127] which replaces second occurrences of duplicated strings with a pointer to the first string to create the set of literals and match lengths. Our third compression tool was bzip2, a compression tool that uses the Burrows-Wheeler transform as opposed to the LZ77 algorithm of gzip.

**Hamming Distance**

Hamming described the distance between two code points $D(x, y)$ as being the number of coordinates for which $x$ and $y$ are different [52]. This distance is referred to as the Hamming Distance. As an example, the distance between any two of the following code points on a three dimensional cube is 2 (that is, two changes are required to move from any 3 digit code to another) [52]:

\[
\begin{align*}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1
\end{align*}
\]
The Hamming Distance is used to measure the number of differences between two sequences of equal length. So if two 20 character sequences are compared, the greatest Hamming Distance possible is 20 (meaning the two sequences of characters have no common characters in the same position in both sequences) and the minimum Hamming Distance is 0 (meaning the two sequences are exactly equal to each other and hence there are no differences).

6.3 Developed Program

Based on our prior human trials and the knowledge we have gained from others’ research, there were a number of critical design decisions made in the development of our prototype CHURN-creation program. These include symbols and alphabet used, displaying choices, random choice locations, and constraining the user, which we will now discuss further.

6.3.1 Symbol Alphabets Used

The size of an alphabet is one of the adjustable security level variables available with a CHURN. As a random sequence used only once, the symbols must simply be able to be recognized by users. As such, playing cards and Mahjong tiles are just two possibilities to use as alphabets for the human user to draw from, and alphabets may be adjusted for specific cultures or organizations. The alphabet size is critical, because if there are twenty symbols in the CHURN then before any further processing or considerations (discussed further below) there are $\alpha^{20}$ combinations possible where $\alpha$ is the size of the alphabet.

Because we wanted to have a baseline to compare the output of our CHURN-generator with, we used the keys available on a standard keyboard. There were other analysis advantages, since each character used was a single byte. So we used all digits, upper- and lower-case letters and special characters from a standard American keyboard. That is, the alphabets we used were:

0123456789
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
~!@#$%^&*()_+-={}[]\:";'<>?,./'
As such, there were ten digits, twenty-six lower case alphabetic characters, twenty-six upper case alphabetic characters, and thirty-two special characters, giving a total alphabet size of 94 characters. It should be clear that sub-alphabets may be used together especially if the method of providing the choices to the human user is graphical and touch based (or mouse-click based) rather than restricted to a keyboard. This means that the set of 52 playing cards could be added to this 94 character alphabet to create a combined alphabet of 146 characters. This would mean that there are 146 choices for each symbol position, and for 20 symbols there would be $146^{20}$ combinations possible.

### 6.3.2 Displaying Choices

Our general technique to provide computer-aided random sequence generation from a user is to use a computer-based random generator to create a set of choices for the human to pick from. We did this to overcome Knuth’s observation that humans are typically unable to create a random sequence themselves [65].

There are two major effects of presenting random choices for the human to choose from. The first effect is that the human is restricted to choices that are as random as the computer pseudorandom number generator used can create. The second aspect is that then the human adds a second source of randomness by making a choice, if their choices are random. One of the main focuses of our trial is whether humans will act as a second source of randomness. A significant benefit of our design is that if the human choices are not predictable, then even if the computer’s random number generator’s seed value and algorithm is known by an adversary, the adversary still needs to guess which of the values the human will have selected in each position. Since the CHURN is used only once, the adversary needs to make the correct guess the first time.

We settled on four choices presented to the user at any one time for the user to make a selection from. Four choices may be represented by 2 bits. This means that up to 2 bits of security comes from the human choices for every character in their generated CHURN, and as such this is a second security parameter variable by adjusting the number of characters per CHURN and the number of choices presented per character. At four choices per character and twenty character positions, for our trial we had a potential 40 bits of security from the human choices.

We do not restrict the choices such that if a "Z" has already been selected
for the first possible choice, a “Z” could still appear in each of the other three choices.

6.3.3 Random Choice Locations

As is clearly understood in the security literature, “user behaviour tends towards the easiest path [22].” While a CHURN is in no way meant to be used as a password, particularly since passwords are meant to be memorable and CHURNs are not, some useful comparisons can be made with computer-assisted password generators. For example there is the computer assisted password generator here: http://www.generatepasswordfree.com/. On this password generator, the user clicks on a background of “random” characters and generates their random somehow-memorable password. With such a technique, there is nothing to stop the user leaving their mouse in the same place and clicking the same letter every time, thus constructing a password of “GGGGGGGGGG” for example. The user is not forced to move the mouse.

By displaying the four choices in random locations, we force the user to move and to make a separate choice each time. As one of our participants of the trial stated, “I like how you move the buttons around. If the button didn’t move I would be tempted to sit on the same button and keep clicking it.” We move the position of our four choices which are presented to the user randomly both on the X and Y axis inside a rectangular area for each choice. The restriction we place on the position of the choices is that the choices cannot overlap each other. As an example of how the buttons move and do not overlap, see Figure 6.3 on page 101.

6.3.4 Constraining the User

Displaying multiple choices for the user to pick from, particularly when there are sub-alphabets such as digits and lowercase letters involved, means that a user’s preferences in their decision making may significantly reduce the randomness, entropy and security of the output. For example, with 32 of the 94 choices being special characters, there will be a special character in many of sets of four choices presented to the user. If the user had a preference to special characters, she may be able to restrict the output from the 94 possible characters the input had down to 32 possible characters. For a twenty character sequence, this reduces
the number of possibilities by $2.9 \times 10^{39}$. Indeed, in the part of our trial where we did not constrain the choices presented to our participants, one participant said, “Oh no! I wanted to create a string without any alphabet characters and I was forced to pick a character.” Without any constraint, such subversion either deliberately or due to unconscious preference, may be possible.

Decomposition of the number of characters in each sequence via a weighted decision tree as per Shannon, yields that for a 20 character sequence with digit probability of $10/94$, lowercase probability of $26/94$, uppercase probability of $26/94$ and special character probability of $32/94$, yields two digits, six lowercase, five uppercase, and seven special characters. While clearly each sequence having this mix of characters is non-ideal because lowercase and uppercase have the same probability, and yet lowercase has six characters while uppercase has five, restricting the user such that each sequence has this combination of characters prevents various forms of undermining the method for creating a CHURN.

Here is a five-character example of how we enforced this constraint on the users. In the five-character example there will be one digit, one lowercase character, one upper case character, and two special characters. Firstly we created four fresh sequences, because this is the number of choices presented at once to a user. These sequences had in the first position a random digit, second position a random lowercase character, third position a random uppercase character and in the fourth and fifth positions random special characters. The example four sequences are shown in Table 6.1.

| Sequence 1 | 6 | r | W | # | [ |
| Sequence 2 | 4 | t | E | ; | = |
| Sequence 3 | 1 | e | F | > | ? |
| Sequence 4 | 6 | c | B | \ | - |

Secondly, we created a fresh single random permutation for this set of four sequences. For example, a random permutation of positions 1,2,3,4,5 may be 3,5,2,1,4. Finally we rearranged each of the four sequences using the same permutation, such that the “digit” is in the same location in all four sequences. The result of applying the permutation 3,5,2,1,4 is shown in Table 6.2.

Now, the user of our CHURN-creator will be presented with the choices from the first position of all four sequences first. That is, the user will be asked to pick
between "W", "E", "F" and B". Once they have made that choice, they will be presented with the choices from the second position of all four sequences. That is they will be asked to pick from "[", ",", "?", and ",", and so on.

In this way, each sequence is guaranteed to have random characters in random positions, but a fixed number of each character type per sequence. By way of illustration, Figure 6.1 shows an example twenty sequence output created by one of the participants:

```
vO3f;Hw{x+OrK2h>$.[C
U=8w(S',5^-WQkyyGqk'
hhjpN+7F5$?L/Y\{'kJa[
vhLOLq1^:-3@FxPH'|k])
93w#X\y\e~D[Qu]%IdQb
owc>Pv&3J=0ZY.V%g@j;
5saMrXI'cV,),?'mGu,?i
/ #$3IY(idF\tc#b=KJIr
LYr?iJqdf#_.!45_aVIC,
p?WZyLN)10$D(/1&g+mf
4uU;bo*Mf(V1E?.%w+E[x
_tR;"c(I_KF{w_c0vf7L
f!41De)bWR+rNIf+_#*0
6pZi?wMo}\-W5\'cplnJ
9er9K'=Q\jy@N:CI~z$v
Q\)kmFm*SRp7\?K5f:\{k
3!omk\H'(5Rb\;R+hPXw
W9L{-8B~dNqFo"hka;'=
T(hb&P--@sG_mL4W_7yf
+%k0hqE(\+:2ANrVT<x1
```

Figure 6.1: Example twenty sequence constrained output
Such a constraint reduces the number of possible sequences generated. For an alphabet of 94 characters, 20 characters per sequence means

$$94^{20} = 2.90 \times 10^{39}$$

possible combinations. Using our constraints, the number of possible outcomes becomes

$$\frac{10^2}{2!} \times \frac{26^6}{6!} \times \frac{26^5}{5!} \times \frac{32^7}{7!} \times (20!) = 3.52 \times 10^{37}.$$ 

This is a reduction by a factor of approximately 100, or reduced to 1.2%. The reason why the constrained version reduces is because there is no distinction between multiple characters from a sub alphabet. For example, two identical digits that appear somewhere in the 20 character sequence. That is, if the first of the two digits is a “5” and that is permuted to sequence position 1, this will result in the same combinations as when the second digit is a “5” and the second digit is permuted to sequence position 1.

Using our random number generator, the inbuilt random function in C#, we generated sequences both constrained and unconstrained for 5,040,000 characters and analysed the results. The single character entropy values were very similar: 6.554573 for the unconstrained sequence versus 6.554312 for the constrained sequence. This similarity was repeated through the n-gram tests, with subsequences in the unconstrained set having eleven more six-character sequences used out of the possible 689,869,781,056 six-character sequences, than the constrained set. It should be noted that even a 5,040,000 character sample is too small for analysis at anything higher than three character positions for entropy, because there will not be enough n-gram symbols to cover all the possible n-gram sequences. The 3-gram entropy for the unconstrained sequence was 19.541 and for our constrained sequence it was 19.515, out of a maximum of 19.664.

### 6.4 Test Setup and Hypotheses

Our goal is to compare a CHURN-generator, which is the constrained graphical version, with how random humans can be when unassisted. Our hypotheses were:

- **H1:** Sequences created by humans using our CHURN-generator would be significantly more random than sequences created by humans without the aid of a CHURN-generator.
H2\textsubscript{1}: There will be a significant second source of randomness due to the human input to a CHURN-generator.

H3\textsubscript{1}: Humans will feel more in control creating a random sequence using a CHURN-generator than when they are given a random sequence created by a computer.

To investigate these hypotheses we asked the participants to complete four tasks: keyboard entry, mouse entry unconstrained, mouse entry constrained, and a small questionnaire at the end. In a commercial implementation of a CHURN, a useful indication of freshness would occur using just five characters. However, for the purposes of gaining the most data possible from our participants, all sequences in our tests were 20 characters long. We will now discuss the tasks performed by each participant further.

### 6.4.1 Keyboard Entry

To compare how random an unassisted human can be we needed an unassisted human baseline for each participant, and as such we asked each participant to type in 20 sequences of 20 characters as random as they could using a keyboard. The entry form from our prototype is shown in Figure 6.2.

Before each keyboard entry trial we outlined to each participant what the usable keys were by pointing to them individually as we said them, particularly for the special characters. Essentially the range of possible keys to be used equated to every key on four rows of the keyboard with the exception of the Enter key and the Tab key. Participants were also told not to use the spacebar.

As users entered their sequence they could see what they had typed. As soon as each of the 20 character sequence was completed, it was hidden from the users view to contribute to the sequences being distinct from each other. This had the effect of not allowing participants to return to earlier sequences and adjust them. Once all 20 sequences were complete, all sequences were revealed back to the participants in a non-editable form and the sequences were saved to file.

### 6.4.2 Mouse Entry Unconstrained

Keyboard entry was unconstrained in that a sequence of twenty characters could have been all digits. Our CHURN-generator is constrained such that all twenty character sequences are guaranteed to have a set number of characters from each
of the sub-alphabets. As a step between the unconstrained keyboard and the constrained mouse entry, we used unconstrained mouse entry.

Users were presented with a window, approximately the size of a smart phone, with four options to pick from by clicking with a mouse. The four options were at locations which had been generated randomly, and each option’s value had been generated randomly from the alphabet of 94 characters without constraints such as “all choices shown at a time must be of the same sub-alphabet”. The process is shown in Figure 6.3.

As may be seen in the left picture of Figure 6.3, the first four choices presented to users were \( z \ S \ b \) and \( \} \). The user in Figure 6.3 selected \( S \), which took the user to the middle screen with \( S \) shown as selected at the bottom in “Your values” and a second set of four choices were presented. This second set of four choices contained \( 7 \ t \ * \) and \( g \). The user selected \( g \) which moved them to the third screen in Figure 6.3 which has \( S \ g \) as the selected so far sequence, and another choice of four characters being \( . \ J \ ~ \) and \( H \). In this way 20 sequences of 20 characters were selected by each participant.
6.4. Test Setup and Hypotheses

Figure 6.3: Unconstrained mouse entry form.
As a secondary capability of this test, even though the positions of the choices and the choices themselves were generated randomly, they were generated randomly and saved. At which point all our participants were shown the exact same four choices in the exact same position for all 20 choices per 20 sequences for the test. What this allows us to do is to examine how random the users’ choices were. Each trial participant would select from the same four choices in the same four positions as every other participant to construct a sequence, and then each participant’s sequence could be compared to every other participant’s corresponding sequence using the Hamming Distance. Two participant’s first sequences are shown in Table 6.3 to illustrate the concept.

Table 6.3: Hamming Distance example from first sequence.

<table>
<thead>
<tr>
<th>Participant</th>
<th>First Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>b7JEzL&lt;+KgZ~es4sK'C#</td>
</tr>
<tr>
<td>B</td>
<td>JtJESL.?sC;f8s:sgZ4q</td>
</tr>
</tbody>
</table>

Table 6.3 shows two twenty character sequences with five matches. A point to note is that the first three choices were made from the choices shown in Figure 6.3. The Hamming Distance for the two sequences is $20 - 5 = 15$. Each participant’s responses may be compared in this way with every other participant’s responses, for each of the twenty sequences. For fifty participants, $50 \times 50 = 2500$ comparisons per 20 character sequence. We remove 50 comparisons from the 2500 comparisons because for each participant one comparison would be with themselves, leaving 2450 comparisons. Since participant A compared with participant B is the same as participant B compared with participant A, we divide by two to get the number of unique-pair comparisons, which is 1225 unique-pair comparisons per twenty character sequence. This gives significant data for comparison with a binomial distribution, which the distribution of Hamming Distances for unique pairs for each sequence will have if the participants are choosing at random.

### 6.4.3 Mouse Entry Constrained

Our CHURN-generator is constrained to reduce the user’s ability to minimize the alphabet size used. Therefore our third test is a test which shows how the CHURN-generator works, specifically with a constrained range of choices and with each participant being shown fresh random choices each time. How the CHURN-generator is constrained is illustrated in Figure 6.4.
6.4. Test Setup and Hypotheses

Figure 6.4: Constrained mouse entry form.

The left picture in Figure 6.4 shows four choices that are presented to the user, and they are constrained such that all choices are lower case alphabetical. The right picture in Figure 6.4 shows a second set of choices presented to the user, the user having selected m to move to the right hand picture, and now the choices are constrained such that they are all digits.

6.4.4 Questionnaire

At the end of the data entry, each participant was asked to complete a questionnaire. The questionnaire mainly had demographic questions such as age, gender, and education level. The final three questions were particular to our study and they asked:

- How many times a week do they use an authentication protocol to access a website on the Internet? Specifically, banking, ecommerce, Facebook, and webmail were mentioned.

- Compared with getting a random sequence of characters given to them by a bank, how in control of their random sequence did they feel when they were allowed to pick each position in the sequence from a set of four choices, as per the mouse entry methods?

- Comparing their keyboard entry and the computer-assisted mouse entry,
which did the users feel resulted in a more random sequence being generated?

### 6.4.5 Ethics

Permission was gained from our university’s Human Ethics committee for this trial. Risks were minimal compared with every day computer usage or compared with playing a computer game. All participants signed a consent form prior to taking part and were able to leave at any time. All participants did finish the trial.

### 6.5 Results

We separate the analysis of our results into the following areas: participant analysis; the human side to randomness; histograms; entropy; compression tools; and Hamming Distance. We will discuss each area further now.

#### 6.5.1 Participant Analysis

Fifty participants took part in and completed our trial. While common criticisms of academic security trials are that the vast majority of the participants are university educated, male, and in the 18–25 year old age bracket, we managed to secure a range of participants to take part in our study. The analysis of our participants is presented here.

**Age, Gender, and Qualifications**

More than a third of our participants were female (19), and approximately a third had no university qualification and were not current university students (16). The age ranges of the participants are shown in Figure 6.5.

**Logins per week**

Each login secured with a password to a website on the Internet is an example of where a CHURN could be used to enhance security. The number of times a week that participants would log into a website such as banking, e-commerce, Facebook or webmail using a password ranged from three participants answering zero times,
6.5. Results

Figure 6.5: Histogram of participant ages.

through to a participant reporting 140 logins per week. The distribution of the number of logins a week per participant is shown in Figure 6.6.

Figure 6.6: Histogram of logins per week.

Feeling in Control

A critical part of secure human protocol design is to empower the user and make them feel as though they are in control. More than two thirds of our participants reported feeling more in control using the CHURN-generator to create a random sequence than being given a random sequence to use by a computer. Five participants reported feeling less in control. A participant stated:

"The reason I would like to choose my own random numbers rather
than be given them by the bank is because there is less chance of someone else having them.”

6.5.2 The Human Side to Randomness

There are two general observations to make about human randomness. Firstly, that humans cannot make uniformly random choices and secondly that humans will see patterns in uniformly random things, such that they will think that something that is uniformly random is not uniformly random. This ability to detect patterns, and to presume that patterns indicates non-randomness, typically leads to humans not picking two digits the same in a row when asked to write random digits down, even though approximately one in every 10 digits should be the same as its predecessor when selected at random [65].

Two examples of verbal responses from participants as they used the unconstrained mouse entry form were:

“There does not seem to be enough numbers. It makes me want to pick numbers.”

“...I picked a capital last time, so I’m trying not to pick a capital this time.”

A further consideration is that humans may repeat a certain character more often than others. While this behaviour is restricted in both of the pick-one-value-from-four-presented methods employed in the trial, we had a participant state while using the unconstrained mouse entry method, “Let’s try finding an ‘A’ and always click an ‘A’.” Therefore humans can err on the side of not being random due to too much repetition; as well as erring on the side of not enough repetition, unless a CHURN is constrained to minimize such behaviour.

As the final question on the questionnaire that all participants completed as part of the trial, participants were asked to compare the randomness of the sequences they generated by themselves compared with the randomness of the sequences they created via the CHURN generator. More than a third of the participants thought that the sequences they created unaided were more random than the sequences generated by the CHURN generator. The analysis of the results of the randomness and uncertainty tests on the aided sequences and the sequences generated by the CHURN generator are presented now.
6.5. Results

6.5.3 Histograms

An illustrative example of the range of values used by each participant in the different tests is a histogram, the simplest of which is the single character histogram. We will first discuss single character frequencies, then n-gram frequencies.

**Single character histograms**

For the purposes of the single character frequencies, the characters used in each of the twenty sequences are combined for each participant. The single character histogram for each participant for both the human-unassisted keyboard and CHURN-generator are shown in Figure 6.7.

The data in Figure 6.7 is not meant to be readable on a per-participant or per-character basis. In particular, on the x-axis only every second character is labelled, even though all values are presented. However, two observations are clearly noticeable.

Firstly, no participant used all characters from the available alphabet of 94 characters in the un-assisted human keyboard input test. Indeed, the maximum number of characters from the alphabet of 94 possible characters used by any of the fifty participants when they used the keyboard input was 84. The mean was 59.4 and the minimum was 23. In contrast, in the computer assisted CHURN output shown in Figure 6.7, in many cases all characters were used by each participant. Eighteen of the fifty participants used all 94 characters available in the alphabet, the lowest number of characters used by any participant using CHURN was 90 and the mean was 92.82. The minimum number of characters used via CHURN being six characters more than the maximum number of characters used by any unassisted participant, is clear supportive evidence that CHURN gives more random output.

The second critical observation to be made of Figure 6.7 is with regards to the y-axis. In the unassisted human keyboard creation of random sequences, many participants used the same character a significant number of times. Specifically, exactly half the fifty participants used one of the characters from the 94 character alphabet thirty or more times in their random keyboard entry sequences. Thirty or more times, out of 400 characters, means that 7.5% or more of the output was the one character for half the participants. In contrast, the most any one character was used by any participant when using the CHURN-generator was 14 times. There were four participants who selected the same character
Figure 6.7: Histogram of single character usage per participant via a keyboard and CHURN-generator.
14 times when using the CHURN generator. Again, maximum usage of individual characters is a strong indicator that the computer-assisted human, using the CHURN-generator, produces a significantly more random and unpredictable output than an unassisted human can.

**N-gram histograms**

The process for analysing the sequences in an n-gram fashion was to look at non-overlapping n-grams inside each sequence. This means two things. Firstly, if bigrams (two character sequences) are considered, then there are ten bigrams per sequence, and 200 bigrams per participant. This ensures independence between bigrams. Secondly, a 5-gram could not start in one sequence and end in another, though this would not occur because five is a divisor of twenty. In this way we keep the sequences distinct.

A repeat of four character sequence is very significant. Each 4-gram should appear every $94^4$ 4-grams on average. That is, every $78,074,896 \times 4$ characters, or every $312,299,584$ characters. Using the unassisted keyboard input, fifteen participants had at least one 4-gram repeat in their twenty sequences of twenty characters (400 characters total) generated. Two participants had at least one 8-gram repeat in their unassisted keyboard input. If this was occurring randomly, an 8-gram should repeat on average every $94^8 = 6,095,689,385,410,816$ 8-grams.

Using the unassisted keyboard input, forty participants had 3-gram repeats. In contrast, using the CHURN-generator, not one participant had a 3-gram repeat. A 3-gram should occur on average every $94^3 = 830,584$ 3-grams, and each participant had $6 \times 20 = 120$ 3-grams analysed using our non-overlapping each-sequence-distinct method.

### 6.5.4 Compression Tools

As discussed in Section 6.2.2, compression tools come in a variety of underlying algorithms. As such, we have settled on three compression tools: an arithmetic encoder described by Moffat et al. [80]; gzip; and bzip2. In general, the smaller the compressed file the less random and less uncertain the original uncompressed file was; and similarly larger files indicate more random and greater uncertainty. All input files were the same size. Of particular interest will be if any algorithm reverses the order of the files compared with another algorithm’s order. That is, if gzip compresses the keyboard output to a file size smaller
than the CHURN-generator output; while the arithmetic encoder compresses the CHURN-generator’s output more than the keyboard output.

The compressed file sizes for each participant are presented in Figure 6.8 for the arithmetic encoder, in Figure 6.9 for gzip, and in Figure 6.10 for bzip2. At all times the unassisted-human keyboard output compressed to a smaller size than both the unconstrained and constrained computer-assisted mouse output. Keyboard output always being smaller than CHURN-generator output, for every participant and for all three compression tools, is a strong indicator that the CHURN-generator creates a more random and unpredictable sequence than an unassisted human can produce.

![Arithmetic Compression](image)

Figure 6.8: Arithmetic compression of participant’s keyboard, unconstrained mouse, and CHURN generation output.

Of interest is that for the same files compressed using the three different tools, some tools compressed the CHURN generator’s output to be larger than the unconstrained mouse output, while other tools had the unconstrained mouse output’s compressed file as larger than the CHURN generator’s compressed file. By compression tool:

- the arithmetic encoder compressed the CHURN generator’s output compressed to a larger file size than the unconstrained mouse’s compressed output for 23 participants (just under half)
- the gzip compression tool compressed the CHURN generator’s output compressed to a larger file size than the unconstrained mouse’s compressed output for 30 participants ($\frac{3}{5}$)
6.5. Results

Figure 6.9: gzip compression of participant’s keyboard, unconstrained mouse, and CHURN generation output.

Figure 6.10: bzip2 compression of participant’s keyboard, unconstrained mouse, and CHURN generation output.

- the bzip2 compression tool compressed the CHURN generator’s output compressed to a larger file size than the unconstrained mouse’s compressed output for 33 participants ($\frac{2}{3}$)

Compression of files is very algorithm- and implementation- specific, and will give different results for different files. As such, significant conclusions cannot be drawn from the values of “which output type creates a larger file” for the mouse unconstrained and CHURN methods, particularly because the differences between the files in many cases are less than 10 bytes (about 2.7%). The con-
clusion to draw from these results comparing the CHURN generator’s output with the unconstrained mouse output, is that the constraints placed in CHURN do not significantly or even noticeably with sequences of 400 characters decrease the randomness of sequences compared with an unconstrained solution. Since the constraints offer significant protection against reduction of the full alphabet to much smaller alphabet, the constraints and CHURN creation method is recommended.

Our conclusions are based on statistical analysis. As an example, to test that CHURN-generator output creates a larger, and hence more random, compressed file using the arithmetic encoder than a compressed keyboard output file, we used the following null hypothesis:

\[ H_0: \text{The mean compressed file size for the keyboard entry files and the mouse entry files are the same.} \]

First we calculated the differences between the compressed file sizes for the keyboard and CHURN-generator files for the arithmetic encoder. We then completed some statistical analysis on these difference values. The mean was 46.46 bytes, the variance was 399.36 bytes, the standard deviation was 19.98 bytes, and given our 50 samples our margin of error was 2.826 bytes. For 49 degrees of freedom and a 95% confidence interval, our T-statistic is 2.009575. Using the T-statistic multiplied by our margin of error gave us a 95% confidence interval of 40.78 - 52.14 bytes. Because our confidence interval does not include 0, we rejected \( H_0 \) thus concluding that the CHURN-generated files do create larger, and hence more random, compressed files.

6.5.5 Hamming Distance

We applied Hamming Distance calculations to the mouse unconstrained data to ascertain the variation that humans were adding to the process. We could do this because all participants saw the same four choices in the same four locations for every set of four character choices in their twenty sequences of twenty characters.

If the selections made by each participant when they are selecting from the four choices presented to them follow a normal random distribution, then the Hamming Distances between each of the participants’ corresponding sequences should be a Binomial distribution. Based on a probability of 0.75 (that is, for each four-choice character position the probability of a match is 1/4 or 0.25, and the probability of a mismatch and hence +1 to the Hamming Distance is 0.75),
and 1225 unique pairings, the expected values for the Hamming Distances for each of the twenty sequences of twenty characters is shown in Table 6.4.

Table 6.4: Binomial distribution of Hamming Distances based on 1225 pairs and probability of 0.75.

<table>
<thead>
<tr>
<th>Hamming Distance</th>
<th>Binomial probability</th>
<th>Expected number of unique pairs with this Hamming Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9.095E-13</td>
<td>1.11E-09</td>
</tr>
<tr>
<td>1</td>
<td>5.457E-11</td>
<td>6.68E-08</td>
</tr>
<tr>
<td>2</td>
<td>1.555E-09</td>
<td>1.91E-06</td>
</tr>
<tr>
<td>3</td>
<td>2.799E-08</td>
<td>3.43E-05</td>
</tr>
<tr>
<td>4</td>
<td>3.569E-07</td>
<td>4.37E-04</td>
</tr>
<tr>
<td>5</td>
<td>3.426E-06</td>
<td>0.004</td>
</tr>
<tr>
<td>6</td>
<td>2.570E-05</td>
<td>0.031</td>
</tr>
<tr>
<td>7</td>
<td>1.542E-04</td>
<td>0.189</td>
</tr>
<tr>
<td>8</td>
<td>7.517E-04</td>
<td>0.921</td>
</tr>
<tr>
<td>9</td>
<td>3.007E-03</td>
<td>3.68</td>
</tr>
<tr>
<td>10</td>
<td>9.922E-03</td>
<td>12.15</td>
</tr>
<tr>
<td>11</td>
<td>0.027</td>
<td>33.15</td>
</tr>
<tr>
<td>12</td>
<td>0.061</td>
<td>74.59</td>
</tr>
<tr>
<td>13</td>
<td>0.112</td>
<td>137.70</td>
</tr>
<tr>
<td>14</td>
<td>0.169</td>
<td>206.55</td>
</tr>
<tr>
<td>15</td>
<td>0.202</td>
<td>247.86</td>
</tr>
<tr>
<td>16</td>
<td>0.190</td>
<td>232.36</td>
</tr>
<tr>
<td>17</td>
<td>0.134</td>
<td>164.02</td>
</tr>
<tr>
<td>18</td>
<td>0.067</td>
<td>82.01</td>
</tr>
<tr>
<td>19</td>
<td>0.021</td>
<td>25.90</td>
</tr>
<tr>
<td>20</td>
<td>3.171E-03</td>
<td>3.88</td>
</tr>
</tbody>
</table>

The observed values for the Hamming Distances for each of the twenty sequences are shown in Figure 6.11, along with the ideal curve from Table 6.4. While the observed Hamming Distance values from Figure 6.11 do resemble the expected values shown in Table 6.4, the graphs have been skewed to the left. So while humans are contributing a source of randomness, the full 2 bits of security is not gained from each character position in the CHURN sequence. Extremely promising is the fact that out of 1225 comparisons for 20 sequences (24,500 comparisons), no two participants created the same two sequences despite being given the same choices in the same positions. Our study represented trying conditions where 800 mouse clicks were required from participants in a row, suggesting that participants started to tend towards the easiest path possible and “click on the
nearest value”. 50 participants and four choices per position means that at least 13 participants started on the same first value of the 20th sequence, and even so, for the 20th sequence there was only one pair with the smallest Hamming Distance of four, out of 1225 20th sequence comparisons.

6.5.6 “Who Decides?”

As described in Section 3.4.1, there is the question asked by human users in security ceremonies of who decides? Who decides which part of the browser (or application) can be changed by a developer, and which parts are fixed by the web browser? Who decides whether HTTPS is shown in the address bar of a web browser; if a padlock is shown in the status bar or address bar; if the address bar is a certain colour? As stated at the beginning of this chapter, this question of “who decides” is of further concern in the case of smart phone applications, where the answer is “the application developer decides on an individual basis,” thus necessitating blind trust.

The answer to the question “Who decides” in the case of a CHURN is the human; at least partially. This goes beyond simply feeling in control, which is also important and which two-thirds of our participants stated they felt, as described in Section 6.5.1. For each character position, the humans are making choices, choosing between four symbols presented to them in random locations. If humans made the selection of a symbol perfectly randomly, this would mean the probability of selecting a particular choice would be 0.25, and that each of the twenty sequences presented in Figure 6.11 would approximate a binomial distribution with a centre at 15 (that is, 0.75 of 20 possible matches for Hamming Distance).

In Section 6.2.2 we stated that we knew human decisions were not completely random, and that typical uses of such tests as the Chi-Squared ($\chi^2$) tests were of little use to us because the test would show us that the values in a CHURN selected by a human would have a very low probability of having resulted from a random process. So we wished for a measure for “How random” the choices were, rather than “Is this truly random?” As such, we have used compression tools and other methods to approximate how random the overall sequence created is.

Now we can go a step further. Using the values created from the mouse unconstrained data we can analyse just the human contribution to randomness. We can do this because all participants saw the same four choices in the same
Figure 6.11: Observed Hamming Distances for participant pairs for each of the twenty sequences.
four locations for every set of four character choices in their twenty sequences of twenty characters, and we have completed the Hamming Distance analysis on those choices in Section 6.5.5.

By changing the binomial probability from 0.75, which it should be for Hamming Distance if 1 value was chosen randomly from 4, to lower values, we can match a binomial probability curve to the data acquired from our tests, on a per-sequence basis. When the constructed binomial curve results in a Chi-Squared test showing a probability of at least 0.5 that the values from the actual Hamming Distance curve for a sequence came from that binomial distribution, we will have a good approximation for the binomial probability for our actual data.

To acquire these best-match binomial values, we used the unconstrained non-linear optimization \textit{fminsearch} function in Matlab. The \textit{fminsearch} function takes a function as its parameter. Looking at the curves in Figure 6.11 we can see that the peaks of the curves are at approximately 13, or a probability of 0.65. Therefore, in the function supplied to \textit{fminsearch} we stepped through probability $p$ values from 0.5 to 0.8, with a step size of 0.001 and calculated this calculation at each probability:

$$\sum_{i=1}^{1225} \ln \left( \frac{20}{y_i} \right) + y_i \ln p + (20 - y_i) \ln(1 - p)$$

where $y_i$ is each of the 1225 Hamming Distance values.

As a graphical example of the process, we show a graph of the 1225 participant comparisons from Sequence 8 in Figure 6.12. As can be seen in Figure 6.11, Sequence 8 is our worst case from our twenty sequences. Sequence 8 is the curve furthest to the left, meaning its probability of a mismatch is lowest. Figure 6.13 shows the same curves as Figure 6.11 with the addition of the matched curve for probability 0.6271 from Table 6.5.

Once we have a binomial probability for the Hamming Distance curve for each sequence, we can subtract it from 1 and invert this number to find the number of choices. So the ideal case of a probability of 0.75 would become $1/(1 - 0.75)$ which is 4. By taking the log to base 2 of the generated number-of-choices number, we get the number of bits required to represent this number of choices. By multiplying the number of bits per choice, which is the number of bits per character position, by the number of characters in the sequence, we get the number of bits of security that the human choices have in the resulting
6.5. Results

Figure 6.12: Observed Hamming Distances for participant pairs for Sequence 8 versus the ideal $p=0.75$ $n=1225$ Binomial Distribution.

This bits-of-security number, which depends on the security parameter which is the length of the sequence, is our best number for an approximation for how much randomness the human choices add to the process. That is, if the random number generator algorithm for the underlying four choices presented to the human for each character position is known, and the seed for the random number generator also becomes known, this would mean that the four values presented to the human for each character position is known. Even with the adversary knowing which four values are presented to the human for each character position, the adversary would still have this bits-of-security number, added by the human, to overcome.

As stated in Section 6.4, in some instances a CHURN of only 5 characters may provide the security properties desired. Therefore, the fitted probabilities for each of the sequences and the resulting bits of security if the CHURN is
5 characters or 20 characters are displayed in Table 6.5. A summary of the minimum, maximum, and average values are presented in Table 6.6. This will provide useful data for making a decision regarding the security parameter of how many characters to have in a CHURN, as this will be the resulting security if the random number generator and seed become known.

The figures presented in Tables 6.5 and 6.6 become particularly important in the case where the algorithm for generating the random choices, and critically the “seed” or “initial value” for that algorithm becomes known. As discussed in Section 6.2.1, a random number provided by a device such as the RSA SecurID token, requires blind trust on the behalf of the human that the generated number is random and cannot be predicted by a malicious party.

In March 2011, a security breach occurred at RSA which meant that the values presented on SecurID tokens were predictable. This knowledge was later used in a breach at Lockheed Martin [51, 75, 100].
Table 6.5: Fitted binomial probabilities for human-randomness contribution to each sequence.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Fitted Probability</th>
<th>Number of choices</th>
<th>Number of bits if 5 characters</th>
<th>Bits of security if 5 characters</th>
<th>Bits of security if 20 characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequence 1</td>
<td>0.6982</td>
<td>3.3135</td>
<td>1.7283</td>
<td>8.64</td>
<td>34.57</td>
</tr>
<tr>
<td>sequence 2</td>
<td>0.6519</td>
<td>2.8727</td>
<td>1.5224</td>
<td>7.61</td>
<td>30.45</td>
</tr>
<tr>
<td>sequence 3</td>
<td>0.6906</td>
<td>3.2321</td>
<td>1.6925</td>
<td>8.46</td>
<td>33.85</td>
</tr>
<tr>
<td>sequence 4</td>
<td>0.6646</td>
<td>2.9815</td>
<td>1.5760</td>
<td>7.88</td>
<td>31.52</td>
</tr>
<tr>
<td>sequence 5</td>
<td>0.6903</td>
<td>3.2289</td>
<td>1.6911</td>
<td>8.46</td>
<td>33.82</td>
</tr>
<tr>
<td>sequence 6</td>
<td>0.6727</td>
<td>3.0553</td>
<td>1.6113</td>
<td>8.06</td>
<td>32.23</td>
</tr>
<tr>
<td>sequence 7</td>
<td>0.6825</td>
<td>3.1496</td>
<td>1.6552</td>
<td>8.28</td>
<td>33.10</td>
</tr>
<tr>
<td>sequence 8</td>
<td>0.6271</td>
<td>2.6817</td>
<td>1.4231</td>
<td>7.12</td>
<td>28.46</td>
</tr>
<tr>
<td>sequence 9</td>
<td>0.6594</td>
<td>2.9360</td>
<td>1.5538</td>
<td>7.77</td>
<td>31.08</td>
</tr>
<tr>
<td>sequence 10</td>
<td>0.6455</td>
<td>2.8209</td>
<td>1.4961</td>
<td>7.48</td>
<td>29.92</td>
</tr>
<tr>
<td>sequence 11</td>
<td>0.6278</td>
<td>2.6867</td>
<td>1.4259</td>
<td>7.13</td>
<td>28.52</td>
</tr>
<tr>
<td>sequence 12</td>
<td>0.6572</td>
<td>2.9172</td>
<td>1.5446</td>
<td>7.72</td>
<td>30.89</td>
</tr>
<tr>
<td>sequence 13</td>
<td>0.6492</td>
<td>2.8506</td>
<td>1.5113</td>
<td>7.56</td>
<td>30.23</td>
</tr>
<tr>
<td>sequence 14</td>
<td>0.6607</td>
<td>2.9472</td>
<td>1.5594</td>
<td>7.80</td>
<td>31.19</td>
</tr>
<tr>
<td>sequence 15</td>
<td>0.6375</td>
<td>2.7586</td>
<td>1.4639</td>
<td>7.32</td>
<td>29.28</td>
</tr>
<tr>
<td>sequence 16</td>
<td>0.6301</td>
<td>2.7034</td>
<td>1.4348</td>
<td>7.17</td>
<td>28.70</td>
</tr>
<tr>
<td>sequence 17</td>
<td>0.6661</td>
<td>2.9949</td>
<td>1.5825</td>
<td>7.91</td>
<td>31.65</td>
</tr>
<tr>
<td>sequence 18</td>
<td>0.6896</td>
<td>3.2216</td>
<td>1.6878</td>
<td>8.44</td>
<td>33.76</td>
</tr>
<tr>
<td>sequence 19</td>
<td>0.6656</td>
<td>2.9904</td>
<td>1.5804</td>
<td>7.90</td>
<td>31.61</td>
</tr>
<tr>
<td>sequence 20</td>
<td>0.6958</td>
<td>3.2873</td>
<td>1.7169</td>
<td>8.58</td>
<td>34.34</td>
</tr>
</tbody>
</table>
Table 6.6: Minimum, maximum, and average values for human contribution to randomness.

<table>
<thead>
<tr>
<th>probability</th>
<th>Number of choices</th>
<th>Number of bits</th>
<th>Bits of security if 5 characters</th>
<th>Bits of security if 20 characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>0.6271</td>
<td>2.6817</td>
<td>1.4231</td>
<td>7.12</td>
</tr>
<tr>
<td>max</td>
<td>0.6982</td>
<td>3.3135</td>
<td>1.7283</td>
<td>8.64</td>
</tr>
<tr>
<td>average</td>
<td>0.6631</td>
<td>2.9684</td>
<td>1.5697</td>
<td>7.85</td>
</tr>
</tbody>
</table>

In contrast, in the case of a CHURN there is a significant number of combinations that any malicious entity would need to cater for even if the random number generator of the device, and the random seed, became known. This is because a critical element in the design and construction of a CHURN is the human-ownership of the constructed random sequence. Thus, even though the malicious entity may be able to completely predict which four values, in which four positions, are presented to a user, the human will still be making choices from those four values. Tables 6.5 and 6.6 tells us that the minimum number of bits of security from any of our twenty sequences, when compared across the 1225 comparisons for each of the twenty sequences, was 28.46 bits of security for a sequence of 20 characters. This means that the malicious entity would still have effectively 369,244,195 possible combinations to select from, even if the human selection distribution (thus the reduction from 40 bits of security to 28.46 bits of security for a 20 character sequence) was known.

### 6.6 Limitations

A CHURN is designed to provide humans with confidence of freshness. Providing a freshness assurance does not protect against other attacks such as Man-In-The-Middle (MITM) attacks, rather freshness only mitigates replay attacks.

A CHURN is a random sequence that a human had some control over the creation of, and as such does not stand on its own. Protocols which include humans will need to be developed that use CHURNs in the same way that computer-to-computer protocols use random nonces.

A critical part of the use of a CHURN is that the human looks at it and recognizes the CHURN as their fresh random sequence, when the CHURN is sent
back to the human by the party they are communicating with. Some technique will be required to ensure the human follows the protocol. For example, sending back to the human multiple random sequences such that the human has to pick which sequence was their CHURN and that information will be included in the next protocol message.

6.7 Summary

Returning to our three hypotheses, we have shown that sequences created by humans using our CHURN-generator are significantly more random than sequences created by humans without the aid of a CHURN-generator. We have shown there is a significant second source of randomness due to the human input to a CHURN-generator via the Hamming Distances and how close the Hamming Distances were to a true Binomial distribution. We have provided security values both for our tested 20 character sequences, and for sequences of 5 characters which would still provide useful security assurances for many real world scenarios. Finally, more than two thirds of our participants reported feeling more in control using the CHURN-generator to create a random sequence than being given a random sequence to use by a computer.
Chapter 7

Human Perceptible Freshness

Humans have had a need to communicate securely for thousands of years, with documented evidence of the use of a scytale (used for transposition ciphers) as early as 475BC [81]. With the proliferation of computers over the last half century, and the capacity computers provide for cryptanalysis, calculations in confidentiality-ensuring cipher schemes have quickly become too complex for the general populace to complete by hand. This has led to a situation where trust is required, not just by the general user but by informed security professionals also.

We shall use HTTPS, a widely used protocol researched and discussed in Chapters 3, 4 and 5, as an example to explain why human-followable security is required. From there, we will define human perceptible freshness and use the recent work by Jager et al. [62] to provide an example of how such a security property may be used in a cryptographic proof of security.

7.1 Introduction

There is a set of people, in the minority of the populace, who are security professionals who feel confident that they know what security properties HTTPS provides and how HTTPS achieves this. They know what the HTTPS indicators are in a web browser to give them the confidence that HTTPS is being used. However, few people, even in this security trained minority who believe they know how HTTPS works, would have set up a test to capture network packets
on a connection using HTTPS, and would have looked at the packets to ensure encryption has been used. A smaller subset would have then taken the extra step, having found an “encrypted” packet, of trying to decipher the packet to ensure that decryption, or perhaps decoding, was not trivial.

The reason to focus on this set of people, who are the security researchers and professionals, is because they are the people creating new academic and real world solutions. There is a persistence in thinking that users need to read warnings (and hence we need to write better warnings [56,61,67]), users need to learn about indicators (and hence the research should be “how to create better indicators” [28,33,58,78,123]), or that the users need to learn more so that their mental model of what the application is doing matches what is being done by the application [2,3,17,121]. Until this thinking is adjusted, the improvements we need will not be realised. If we can highlight that these approaches do not work, even for the security researchers and professionals, then this will motivate the need for a new approach to security protocols involving humans, significantly.

Some commonly used web browsers give little indication of there being a form on a webpage delivered over HTTPS that has a completely insecure HTTP address as the form’s target. Since the webpage containing the form is sent to the user’s machine over a HTTPS connection, the typical HTTPS security indicators of showing “HTTPS” in the address bar, displaying extended validation certificate information, and a padlock to show HTTPS is in place are all displayed to the user. Even if the user goes to the trouble of viewing the certificate information for the connection, there is no indication that the link may be insecure. However, all of these indicators are showing information regarding the connection over which the webpage was loaded, they are not showing indicators for where the information submitted on a form on the webpage may go.

An example is our university, the Queensland University of Technology, which has a secure connection for a website used by staff and students called “QUT Virtual”. QUT Virtual is a webpage unique and private to each staff member and student, with information and a number of forms on it. One of those forms provides the capability to type in some text to search the library catalogue, see Figure 7.1. The QUT Virtual webpage has been delivered over a secure connection. While several other forms on the page have target addresses which are HTTPS addresses, the form for searching the library catalogue has a HTTP address as the target and the information on that form will be sent over an
insecure connection. Note that in Figure 7.1, for this webpage with the library search form with the insecure HTML target address, “https” is shown to the user at the start of the address bar and a padlock is shown at the end of the address bar.

![Image](https://example.com)

**Figure 7.1:** Example HTTPS webpage with a HTTP form viewed using Internet Explorer 10.

While no security for a library search may be fair, the same display of all the security indicators would occur for a webpage delivered over HTTPS which has a form via which users submit their bank login information. Even the security professional would need to look at the source code for the website, on top of the checks of all the security indicators, to ensure that the target of the form is a HTTPS address. The source code check would need to be done every time the webpage is visited, as the content may have changed. Perhaps checking the source code every time a webpage is used may be done by someone very security conscious and diligent. However, such a check does not take into account the possibility of the webpage being updated dynamically, such that the source code shown is not the source code used by the webpage when the user clicks the “submit” button on the form.

To further emphasise this point regarding trust, and that even the security professional minority require significant trust, the case of smart phone appli-
cations was presented in Chapter 6. Since there is no overarching program to load the app(-lication)s in, as per a web browser for webpages, each application’s designer makes individual choices over what security to use and what security indicators to show to the users. To ask the question asked throughout this thesis again, “Who decides?” the answer in the smart phone case is that the designer of the applications is the person who decides, and they may have no security training at all. As such, there is no enforced connection between security indicator shown, and security provided. Shin et al.’s [110] analysis of 212 smart phone applications showed both secure applications with no security indicator and insecure apps with indicators shown. Both of these cases occurred in applications supplied by financial institutions. In such an environment, expert knowledge such as “What does an extended validation certificate mean?”, “What is HTTPS?”, “How does a user view the certificate information and what do the different fields mean?” will not help the user. This results in users who are information security experts being placed in the same position as the general population is with web browsers, that is, having no easy method to discern a secure application from an insecure application.

This means that there are only two ways forward regarding use of the device. The first option is that the user will not use the smart phone for anything, such as banking, which needs security. This option is increasingly problematic in the case of the web browser on a personal computer, since many large organisations such as government have forms that are only accessible via the Internet. The second option is that the user applies blind trust, and crosses their fingers and hopes. In such an environment as this, users will create their own security mechanisms, which is exactly what we found in the case of web browser usage on computers, as described in Section 3.4.2.

Such a need for blind trust is highly undesirable, leading to both the situation where secure systems are not trusted and also where insecure systems are trusted. Therefore, we wish to move towards a concept of human followable security. This is a step beyond Hopper and Blum’s \((\alpha, \beta, t)\) human-executable protocols [59]. Rather than the human being able to follow the steps and execute a protocol, the question being asked is “Is the security understandable?” It is also a shift from a common traditional HCI-security standpoint of trying to get the users to “do the right thing” [2]. What is human followable security? The answer to this question is too large to be fully encompassed by this thesis. However a step
towards human followable security is *human perceptible freshness*.

In this chapter we define *human perceptible freshness* (HPF), building on prior work to define how HPF may be instantiated in cryptographic protocols. As discussed in Section 2.4, the need to ensure that messages are new, or fresh, is a very common protocol requirement to prevent replay attacks [13]. However, even though the need for freshness of protocol messages has been clearly understood for some time, typical protocols involving humans have a password sent by the human, or HPA sent to the human, repeatedly used. As discussed at the beginning of Chapter 6, the Verified-by-Visa scheme sends the same recorded-at-initialisation message (HPA) to the user on every protocol run, to authenticate Visa to the human. This has a known weakness that if the secret is captured it can be replayed [34].

By using out-of-band channels a HPA could be fresh each time it is sent to the human inside a protocol. For example, the human could send a fresh message to a trusted party such as their bank, the bank could pass on the fresh message to a third party such as Visa, and Visa could use the fresh message in a Verified-by-Visa protocol run. Note that the connection between the human and the bank is pre-existing, meaning the human is familiar with and has used the connection to their bank previously. Similarly, the connection between the bank and Visa is pre-existing, as this is how credit card payments are authorised, see Section 4.2.2.

One method of providing HPF would be via the use of CHURNs (see Chapter 6), which the human would know has been recently created because the human assisted in the creation of the CHURN. The human could send a CHURN via a secondary channel, and when they receive the CHURN back they *recognise* (see Chapter 5) the CHURN, thus gaining an assurance of freshness and implicit authentication.

### 7.2 Human Perceptible freshness (HPF)

As discussed in Section 2.4, there is a need to ensure that messages in a protocol are fresh, that is that they have not been sent before, to prevent replay attacks. While freshness has been achieved for some time in cryptographic protocols between computers via techniques such as nonces, timestamps and counters, there seems little or no focus on freshness in protocols involving humans, particularly
in the direction of “to” the human.

Intuitively, we say a protocol has human perceptible freshness if an object that the human had control over the creation of, and which has not previously existed to the human’s knowledge, is sent back to the human and it is the first time the object has been sent to the human. The critical elements to this human perceptible freshness concept are:

**First use** As is common in protocol design, a freshness value must have the property that it can be guaranteed not to have been used before [13].

**Human-control** A concept that is unique to HPF, the human must have some control over the creation of the freshness value.

**Non-predictable** The freshness value must be non-predictable.

We shall now discuss the implications of these attributes.

**First Use** The concept of needing to be able to guarantee that a freshness value has not been used before suggests some sort of computer pre-selection, or other random generator preselection, of values. Otherwise, if a human were to select freshness values unaided, we have shown in Section 6.5.3 that the likelihood that the values could be guaranteed to not have been used before is remote.

**Human-controlled** While values may be pre-selected by a random generator to construct the freshness value from, the human must have some control over the creation of the freshness value. In this way, the human can understand that the value is unique (fresh) to them. A contrasting case would be if a computer provided the human with a freshness value, in which case the human could not be confident that other computers in the world were not all outputting identical freshness values, thus providing no surety of freshness.

**Non-predictable** In cryptographic protocols, predictable freshness values such as timestamps and counters are viable because simply by having a part of what will be the “plaintext” change will ensure that the “ciphertext” is different, thus preventing replay attacks of the ciphertext. However, when using human perceptible freshness, the ability to recognise that a value is fresh also stems from the value being unpredictable. This is because firstly, humans are not able to measure time accurately and it is unrealistic to assume they will maintain counters. As such, timestamps and counters, guaranteed to provide a new value each time, are not suitable as the freshness value in the case of human perceptible freshness. However, because there is no human-perceptible ciphertext which the
freshness value usually ensures cannot be replayed, the human freshness value itself needs to have non-predictable qualities. In this way, HPF goes beyond standard freshness definitions. Now, a critical ingredient required to ensure that a value is fresh is that the human must also be able to identify that the value is their value.

It is not by chance that the intuitive definition of human followable freshness aligns closely with the design requirements for CHURNs presented in Chapter 6. CHURNs were designed to provide a human equivalent to the concept of a computer nonce, and form one example of a set of values that could provide human perceptible freshness.

If we presume the freshness value will be sent to the human to recognise, we can use the recognise formalism presented in Chapter 5 to capture the concept of whether the human accepts that the freshness value sent back to them is the value they created. We shall label the freshness value $HPA_{fresh}$. Now we define human-perceptible freshness.

**Definition 7.2.1** (Human-perceptible freshness). For a given freshness value $HPA_{fresh}$, it should be infeasible for a polynomially bound $A$ with access to all prior $HPA$ values to construct a $HPA'$ such that

$$\text{Recognise}(\cdot, HPA_{fresh}, HPA') = 1.$$  

### 7.3 Model

Utilising the concept of $POPS+$ described in Section 4.4.2, we focus on the human element to the proof of security. That is, there need to be security proofs for each of the higher level cryptographic primitives such as HTTPS used, after which the human considerations can be analysed. As such, we need two aspects. Firstly, we need security proofs for “higher level cryptographic primitives” such as HTTPS. Secondly, we need a way of binding the messages that will form part of the human-side of the protocol to the cryptographic protocol. From a proof of security point of view, this binding requires that different sessions can be identified, such that the requirements for fresh messages can be formally specified. It turns out that recent work by Jager et al. [62] provides a concrete solution to these two requirements.
7.3.1 Existing Work

We build on the authenticated and confidential channel establishment (ACCE) work by Jager et al. [62]. An outline of the concept of ACCE is that there are two phases, a pre-accept phase and a post-accept phase. In the pre-accept phase, the communication partners are mutually authenticated and a session key $k$ is established. In the post-accept phase, the communication partners use the key $k$ to encrypt and decrypt data transmitted between them using authenticated encryption. Thus all future messages, including any messages for the human-side to the protocol, are delivered over this channel that binds the cryptographic properties and the session information to the human messages.

ACCE Definition

Jager et al. [62] define an ACCE protocol as a protocol which is executed between two parties. The protocol consists of two phases, being the pre-accept phase and the post-accept phase.

In the pre-accept phase each party is authenticated with the other party (mutual authentication), and a session key $k$ is established. The phase ends when both parties reach an accept-state.

In the post-accept phase, data communicated between the two parties is encrypted and decrypted with the key $k$ created in the pre-accept phase.

Model

Jager et al. [62] expand on the work of Bellare and Rogaway [7] to provide a security model. Jager et al.’s model is shown here for completeness.

Each oracle $\pi^*_i$ keeps as additional internal state a bit $b^*_i$ which is chosen at random at the beginning of the game.

Send$^{\text{pre}}(\pi^*_i, m)$ : The adversary can use this query to send message $m$ to oracle $\pi^*_i$. Send replies with an error symbol $\bot$ if oracle $\pi^*_i$ has state $\Lambda = \text{accept}$. (Send-queries in an accept-state are handled by the Decrypt-query below).

Reveal($\pi^*_i$) : Oracle $\pi^*_i$ responds to a Reveal-query with the contents of variable $k$. Note that $k \neq 0$ if and only if $\Lambda = \text{accept}$.

Corrupt($P_i$) : Oracle $\pi^*_i$ responds with the long-term key $sk_i$ of party $P_i$. If Corrupt($P_i$) is the $r$-th query issued by $\mathcal{A}$, then we say that $P_i$ is $r$-corrupted. Non-corrupted parties have $r := \infty$. 
Encrypt\((\pi^*_i, m_0, m_1, \text{len}, \text{Hdr})\) : This query takes as input two messages \(m_0\) and \(m_1\), length parameter \(\text{len}\), and header data \(\text{Hdr}\). If \(\Lambda \neq \text{accept}\) or if the encryption of \(m_0\) or \(m_1\) results in an invalid ciphertext, e.g. if \(\text{len}\) is invalid, then \(\pi^*_i\) returns \(\bot\). Otherwise, a ciphertext corresponding to \(m_0\) or \(m_1\) is returned, depending on the random bit \(b^*_i\).

Decrypt\((\pi^*_i, C, \text{Hdr})\) : This query takes as input a ciphertext \(C\) and header data \(\text{Hdr}\). If \(\pi^*_i\) has \(\Lambda \neq \text{accept}\) then \(\pi^*_i\) returns \(\bot\). Since this is authenticated encryption and decryption, as a means to capture the ability of \(A\) to construct a legitimate ciphertext, if the random bit \(b^*_i = 0\) then \(\pi^*_i\) returns \(\bot\); otherwise \(C\) is decrypted and returned. Returning \(\bot\) if \(b^*_i = 0\) allows \(A\) to ascertain \(b\) and win the security game, if \(A\) can construct an authenticated ciphertext to decrypt.

The first three queries are standard in cryptographic models. The Send query allows the adversary to run the protocol normally and to inject messages of his choice, reflecting the assumption that the adversary controls communications between protocol participants. The Reveal query returns the short term key of the party, modelling the ability for the adversary to acquire short term keys from parties. Corrupt returns the long term key of the party, thus allowing the adversary to effectively act on that party’s behalf, with the addition in this model being that the timing of the corruption is critical. The position of the Corrupt query in the sequence of queries relates to the definition of freshness, as the party is said to be fresh before the Corrupt query and not fresh after the Corrupt query.

The last two queries build on the work by Paterson et al. for length hiding encryption [87]. The Decrypt oracle forbids decryption for any ciphertext which the adversary could have obtained from the Encrypt oracle in the same state. Therefore, if the scheme is secure, we expect the Decrypt oracle always to output fail. If, on the other hand, the scheme is insecure and the adversary can construct a valid ciphertext of his own (a forgery) then he can use the Decrypt oracle to obtain \(b\) and therefore win the game. If decrypt returns an \(m\) then \(b = 1\), otherwise \(b = 0\).

**Definition 7.3.1** (Jager et al. [62]). The probability of \(A\) being able to successfully guess the value of \(b^*_i\) is \(\epsilon\) distance from random chance, that is, \(1/2\). An ACCE protocol is \((t, \epsilon)\)-secure if there exists no adversary that can break the protocol with probability greater than \(\epsilon = \epsilon_{\text{auth}} + \epsilon_{\text{enc}}\) in time \(t\).
7.3.2 Updated Model

The concept, that Ellison’s [41] early security ceremony example captures, is that a user may be securely connected to an entity via HTTPS, but that the entity that the user is securely connected to may not be the same entity that the user thinks they are connected to (see Section 4.2.1). Indeed, even when extended validation certificates are used, and are understood by a user, the user must already have a correct and exact knowledge of the company’s name. The presumption of such knowledge is not practical or realistic in many cases. Consider the first purchase from a website from a company that the user has no prior knowledge of, or cases where typing in a web address causes the redirection to a seemingly unrelated address. An example of a large online entity redirecting to another address is Hotmail.com redirecting their users to Live.com for their login.

Attack

There exists an attack, or perhaps simply a human mistake, where the adversary has a web address, company name, and an extended validation certificate for the purposes of HTTPS, which may be for a company other than the company that a human wishes to be communicating with. A HTTPS channel, with both parties having certificates, which Jager et al.’s ACCE work targets [62], will create a secure channel with “some entity”. The proof of security of an ACCE channel states that except with negligible probability, party A will definitely be communicating with the party who owns the security certificate for Party B, and Party B will definitely be communicating with the party who owns the security certificate for Party A. This assurance is made between the two computers.

Above the level of the two computers communicating with each other, at one or both parties, may also be a human. Returning to the Verified-by-Visa example discussed in Section 7.1, if a user visits an online store for the first time, and the store redirects (possibly through a sub-form) the user to Visa for payment, what assurance does the user have that they are communicating with Visa? In the past, the authentication from Visa to the user has come via the reuse of a personal message or passphrase the user recorded in a setup phase (see Section 6.1.1). If the Verified-by-Visa protocol was updated with a CHURN replacing the personal message, then the message authenticating Visa may be fresh each time. Such human-level authentication, above the level of the ACCE channel, allows for a check regarding whether the server $S'$ that the human’s
computer $C$ is securely connected to, is in fact the server that the human wishes to be connected with.

**H-ACCE Protocol Definition**

We define a human ACCE protocol, H-ACCE, which is based on the ACCE protocol of Jager et al. [62]. Such a H-ACCE protocol consists of three phases, being the *pre-accept* phase, the *human-accept* phase, and the *post-accept* phase.

In the pre-accept phase, as per the definition of an ACCE protocol (see Section 7.3.1), each party is authenticated with the other party (mutual authentication), and a session key $k$ is established. The phase ends when both parties reach an *accept*-state.

In the human-accept phase, which is entered once the two parties reach an *accept*-state, a $HPA'$ is sent from a party, for example $S'$, to the other party $C$ over the created ACCE channel, using the key $k$. The second party $C$ decrypts the $HPA'$ and displays the $HPA'$ to the human $H$. $H$ uses the recognise function from Section 5.2.1, $\text{Recognise}(H, HPA, HPA')$. If $H$ accepts, then the protocol continues to the post-accept phase. If $H$ rejects, then the protocol is aborted. By $H$ accepting we say that $S'$ is in the set of accepted parties of the entity $S$ that $H$ sent their $HPA$ to.

$H$ will have sent their $HPA$ to a party they trust $S$. $S$ may then either use the $HPA$ themselves or may have forwarded $H$’s $HPA$ on to a third party $S''$ from $S$’s set of accepted parties. This captures the idea that a human may know and have a secure connection to their bank and hence may send a $HPA$ to their bank, and the bank then forwards that $HPA$ onto another party such as Visa, through the existing secure channel between the bank and Visa. Thus, if Visa sends the $HPA$ to the human through a channel created by an ACCE protocol, then the human gains assurance that Visa *is* Visa, or at least is an entity that their bank would release the $HPA$ to. For example, this could also be Mastercard or similar.

In the post-accept phase, as per the definition of an ACCE protocol, data communicated between the two parties is encrypted and decrypted with the key $k$ created in the pre-accept phase.
Updated Model

To facilitate the creation of proofs of security for protocols using H-ACCE, we must modify the model presented in Section 7.3.1 to allow for the separation of parties into humans and computers. Parties must be able to send messages to the human, via their computer, for authentication. We restrict these human authentication messages to the human-accept phase of the H-ACCE protocol, after the pre-accept phase. Since messages sent in the pre-accept phase are modelled using the Send query, and messages sent in the post-accept phase are modelled using the Decrypt and Encrypt queries, we create a new query SendCorrectHPA for obtaining the encrypted correct HPA from the server.

There are now two different end points to the security game. After \( \mathcal{A} \) gains the encrypted HPA, \( \mathcal{A} \) can halt the game and output a \( \text{HPA}' \). The adversary wins if \( \text{Recognise}(H, \text{HPA}, \text{HPA}') \) accepts. Or \( \mathcal{A} \) can continue the game and later output a bit \( b \) as per the original ACCE model.

This model thus provides for the real-world case where a human’s computer is securely connected with a server, but the server may not be the server that the human wishes to communicate with. Beyond simply being the wrong server, the server could be deliberately trying to masquerade as the server \( H \) wishes to communicate with and as such will inject HPAs.

The model also still provides for the case where the adversary tries to break the encryption. The real world scenario this models is where a human is connected with the correct server and hence the adversary faithfully forwards that correct HPA message on to \( H \). \( H \) will accept the HPA, transitioning the protocol into the post-accept phase. Subsequent further communication occurs in the post-accept phase, at which point the \( \mathcal{A} \) can still try to break the authenticated encryption scheme. Therefore \( \mathcal{A} \) is a benign adversary for the human-accept phase.

SendCorrectHPA can be used only in the human-accept phase. Only one SendCorrectHPA can be used per protocol run. The SendCorrectHPA query is as follows.

\[
\text{SendCorrectHPA}(\pi^s_i, \pi^c_j, Hdr) \quad \text{This query can be called once for each protocol session. If either } \pi^s_i \text{ or } \pi^c_j \text{ is not in the accept-state, or if } \pi^s_i \text{ and } \pi^c_j \text{ are not partnered, then } \bot \text{ is returned. Otherwise, a ciphertext corresponding to the correct } HPA \text{ is returned. If } \mathcal{A} \text{ does not halt and output a } HPA, \text{ then if } \text{Recognise}(H, \text{HPA}, \text{HPA}) \text{ outputs } \text{reject then the session is aborted at the client.}
\]
Otherwise the game transitions to the post-accept phase.

Since the existing model has a \textit{Reveal} query, this means that $\mathcal{A}$ has access to all prior HPA$s$ for all parties. Allowing $\mathcal{A}$ to inject and display to the human HPA’s captures the concept that the party that the human’s computer is ACCE securely paired with may be malicious, and may have access to previously used HPAs.

\textbf{H-ACCE Security}

To provide the intuition of the concept of a server check to mitigate the attack outlined in Section 7.3.2, we will write in terms of the client and server as per HTTPS, which is used in the ACCE discussion [62]. The client can be separated into a human $H$ and a computer $C$, as per Section 5.3.1.

At the end of the ACCE pre-accept phase, the client has authenticated the party that they are communicating with is server $S'$. After a HPA is sent from $S'$ to $H$ (via $C$) to authenticate the server to the human, if the human \textit{accepts} then the human believes they are communicating with server $S''$, and that the $S''$ that the human believes they are communicating with is the same as the $S'$ that $C$ has a secure connection with. As discussed in Section 7.3.2, this confirms that $S''$ is in the set of trusted parties of $S$ that the human sent their HPA to.

We now define a secure human ACCE protocol, based on Definition 5.2.1.

\textbf{Definition 7.3.2} ($\delta$-$\epsilon$-H-ACCE-Security). An H-ACCE protocol $\Pi$ is $\delta$-$\epsilon$-secure if the protocol is based on a $(\epsilon, t)$-secure ACCE protocol, and no efficient adversary can win the H-ACCE security game with probability greater than $\delta$.

\textbf{7.4 Compiler}

The method of using a fixed technique to transform one protocol type into another protocol type is known as a compiler. We now construct a compiler which will turn an ACCE protocol into a H-ACCE protocol. The steps are these:

1. (additional step) Offline, the human $H$ selects a HPA which the human remembers and sends to their trusted server $S$.

2. The standard ACCE pre-accept phase precedes as defined by Jager et al. [62] between a client $C$ and a server $S'$, see Section 7.3.1.
3. (additional step) After the pre-accept phase, $S'$ tries to authenticate to user $H$. $S'$ asks the trusted server for $H$’s $HPA$.

4. (additional step) $S'$ sends $HPA'$ to the human via their computer $C$ over the ACCE channel using key $k$.

5. (additional step) The $\text{Recognise}(H, HPA, HPA')$ function is executed and, if the human accepts, the protocol proceeds; otherwise, if the human rejects, the protocol aborts.

6. The standard ACCE post-accept phase.

Using TLS we can provide a concrete example of an instantiation of an ACCE protocol, shown in Figure 7.2.

![Diagram of compiled H-ACCE protocol based on TLS protocol proven to be a secure ACCE protocol](image-url)
Theorem 7.4.1 (H-ACCE Security). If $\pi$ is a $(\epsilon, t)$-secure ACCE protocol, then the compiled protocol $\pi'$ is a $\delta$-secure H-ACCE protocol.

Proof. The intuition of our proof is that by simulating of an ACCE protocol we can reduce a H-ACCE protocol to a Recognise game; and secondly by simulating the Recognise game we can reduce a H-ACCE protocol to an ACCE protocol. Thus there are two ways for the adversary to win:

Case 1 by winning the security game of the underlying ACCE protocol (see Section 7.3.1); or

Case 2 by winning the Recognise security game (see Section 5.2.2).

Since we have the adversary’s advantage for both of these cases, we can provide the security of the H-ACCE protocol.

Suppose there exists a successful adversary $A$ against a H-ACCE protocol.

Case 1 We build an adversary $A_{ACCE}$ against the ACCE security game using $A$. Let oracle $\pi_i^c$ be the client and $\pi_j^s$ be the server partner of $\pi_i^c$. The ACCE game challenger, $C_{ACCE}$, creates all public key and parameter information and sends this to $A_{ACCE}$ who faithfully passes this information on to $A$. $A_{ACCE}$ responds to all queries from $A$ using $C_{ACCE}$, except in the case where $A$ asks the special H-ACCE query $SendCorrectHPA$. When $A$ asks the $SendCorrectHPA$, $A_{ACCE}$ generates a $HPA$ using the $GenHPA$ algorithm, and calls $Encrypt$, with both $m_0$ and $m_1$ being the generated $HPA$, to $C_{ACCE}$ and returns the response to $A$.

At some stage $A$ outputs a bit $b$ which $A_{ACCE}$ passes on to $C_{ACCE}$. In this case, $A_{ACCE}$ wins the security game exactly when $A$ wins the security game, and thus the advantage of the H-ACCE adversary is the same as the ACCE adversary.

Case 2 Again we employ Shoup’s game hopping proof technique [112] augmented by Dent [32]. We employ a sequence of two games, the first game being the security protocol shown in Figure 7.2. We transform this to the security game for the Recognise function from Section 5.2.1, bounding the adversary’s advantage between the two. We denote $Win_i$ as the probability of the adversary winning game $i$.
Game $G_0$ describes the compiled H-ACCE security game of 7.2. The game is played between a probabilistic polynomial time (PPT) bound adversary $A$ and a Recognise adversary $A_{Rec}$. $A_{Rec}$ simulates protocol participants and answers all of $A$’s queries. $A_{Rec}$ generates all public keys and parameter information, and sends this to $A$.

Game $G_1$ describes a game which is the same as Game $G_0$, except that when $A$ calls the SendCorrectHPA query, $A_{Rec}$ responds with a random string of the correct ciphertext length. At some stage $A$ halts and outputs a HPA. $A$’s advantage is bounded by its advantage over the ACCE encryption algorithm, hence from Definition 7.3.1

$$|\Pr[Win_1] - \Pr[Win_0]| \leq \epsilon_{enc}.$$  

Since in $G_1$, $A$ gets only a random string, $A$ gains no information from its interaction with $A_{Rec}$. Therefore, remembering Definition 5.2.1 and thus $\Pr[Win_1] \leq \delta$, $A$’s total advantage is bounded by $\delta + \epsilon_{enc}$.

Since the adversary either wins with case 1 or case 2, then $A$’s advantage is the maximum of $(\epsilon_{auth} + \epsilon_{enc})$ and $(\epsilon_{enc} + \delta)$. Therefore, $A$’s advantage must be bound by

$$\text{Succ}_{A_{H-ACCE}} \leq \epsilon_{auth} + \epsilon_{enc} + \delta.$$  

\[
\]

7.5 Human-Perceptible Freshness for H-ACCE

This chapter describes a new concept, human-perceptible freshness, for cryptographic protocols involving humans. In standard cryptography, freshness is assured based on devices such as nonces, timestamps and counters. When freshness is defined, there is a way of binding the values for nonces, timestamps and counters to particular executions of a protocol. Typically these previous executions of the protocol are called sessions.

7.5.1 Session Identifiers

There are various ways to create session identifiers, typically called session IDs or SIDs [24]. The session ID is critical, because this is how two parties will know
they are a pair, and ensuring communication is only with the correct party is a critical reason for both the use of cryptography and a necessity for systems to interact correctly.

An entity, such as a bank, on the internet will have several concurrent executions of a protocol happening at once. This analogy can be taken a step further by considering who might be interacting with the bank. An entity, such as a supermarket, may have multiple different protocol executions (for example, from different checkouts) to the same bank, at the same time. There needs to be a method to make these protocol sessions distinct from each other and separately identifiable. There needs to be a method of creating distinct session identifiers.

An example of a method for the creation of session identifiers is the concept of matching conversations between the two parties, as defined by Bellare and Rogaway [7]. In later work, Bellare et al. used concatenation of the messages exchanged in the execution of the protocol to identify matching sessions [6].

Of particular importance to the concept of freshness is that because the session identifiers are being created by a function of the messages being sent between the parties, then simply having a new item such as a nonce, timestamp, or counter means that the current session is discernible from all prior sessions. Indeed, the definition for freshness in cryptographic protocols is usually based on this concept, for example by the nonce corresponding to the protocol run with that session identifier being new.

Therefore, a critical aspect of how to discern if the messages in a protocol are being sent freshly becomes the question of how to identify and separate the current protocol execution from all prior protocol executions. It is unrealistic to expect humans to be keeping and comparing transcripts of protocol messages. It is to this end that we have used the ACCE channel as our building block, because simply by encrypting using key $k$ generated in the pre-accept phase, we can cryptographically bind the $HPA$ sent to the human in the protocol to a particular session, which can be used to prove that messages are fresh. Therefore, we can now define human perceptible freshness for a H-ACCE protocol (H-ACCE-HPF).

### 7.5.2 Human Perceptible Freshness for a H-ACCE protocol H-ACCE-HPF

Based on our general definition of HPF from Definition 7.2.1, and based on the freshness section of the definition of security (Jager et al.’s Definition 8 [62]) for
an ACCE protocol using the model outlined in Section 7.3.1, we define H-ACCE-HPF as:

**Definition 7.5.1** (Human-perceptible freshness for a H-ACCE protocol H-ACCE-HPF). We say that an H-ACCE protocol has freshness and human perceptible freshness if party $P_i$ is $r_j$-corrupted with $r_0$, the query that causes $\pi_i^*\text{ to accept}$, being before $r_j$; and, given the set of previously used HPAs by $P_i$ and by all other parties, $\{HPA_{i-1}^k\} \cup \{HPA_{i-1}^m \neq s\}$, $A$’s advantage of selecting a $HPA'_{i}$ that the human recognises as their $HPA$ should be no more than $\delta$.

Note that for parties $P_i$ that are not corrupted, $r = \infty$.

In the CHURNs we presented in Chapter 6, we have a concrete example of a $HPA$ which could be used in the protocol presented in Figure 7.2 to create a protocol that satisfies Definition 7.5.1 for a H-ACCE-HPF protocol. This is because of the high cryptographic security of the resulting CHURNs which a computer aided the creation of, and also because, even if the random number generator algorithm and seed were known such that all “random” values could be predicted, the 20 character CHURN would still have a minimum of 28 bits of security (see Table 6.6). The protocol from Figure 7.2 with CHURNs as the HPAs is shown in Figure 7.3.

**Corollary 7.5.1** (CHURN Provides H-ACCE-HPF Security). If $\pi$ is a secure H-ACCE protocol, then the protocol with a CHURN as the $HPA$ has H-ACCE-HPF Security.

### 7.6 Summary

In this chapter we have introduced the concept of human-followable security, and provided a concrete construction which provides one of the pillars of human-followable security: human-perceptible freshness. By using the CHURNs presented in Chapter 6 and building on recent ACCE work by Jager et al. [62], we have created a $POPS+$ security proof as introduced in Section 4.4.2. The use of a higher level cryptographic primitive of an authenticated and confidential channel establishment protocol, which in the real world is instantiated by the TLS protocol, allows us to build protocols for human use where the human-interaction is the focus of the proof.
Table 7.3: Compiled H-ACCE protocol using a CHURN and the TLS protocol

<table>
<thead>
<tr>
<th>Human</th>
<th>Client</th>
<th>Server</th>
<th>Trusted Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>C</td>
<td>S'</td>
<td></td>
</tr>
</tbody>
</table>

Out-of-band Communication (added by compiler)

\[ CHURN \]

Pre-accept phase

\[ \text{TLS handshake} \]

Human-accept phase (added by compiler)

\[ \text{OOB}\{ \text{Request HPA}' \} \]

\[ \text{OOB}\{ \text{CHURN}' \} \]

\[ \text{CHURN}' \]

Recognise\((H, HPA, \text{CHURN}')\)

if reject \(\rightarrow\) abort

Post-accept phase

\[ \text{further communication} \]

Figure 7.3: Compiled H-ACCE protocol using a CHURN and the TLS protocol
Chapter 8

Conclusion and Future Work

We step through the conclusions drawn from each of our chapters, and present future directions for research that builds on the work presented in this thesis.

8.1 Conclusion

We started our research in qualitative human studies to gain an understanding of human use of security protocols. We also analysed protocols known to be broken when used by humans, to build an understanding of what a security ceremony was, and what security ceremony analysis would need to be. Based on the understanding gained by this research, we have made various contributions to aid cryptographers make protocols which would be secure for human use. They are: a model for human recognition, a method for humans to create cryptographically significant sequences that can be used as human nonces, and a new security definition, human perceptible freshness, useful when creating protocols intended to be secure when used by a human, and a compiler to deliver this definition.

8.1.1 Enhanced Understanding of Security Ceremonies

Through this research we have arrived at the thesis that a security ceremony, more particularly a secure security ceremony, must include human-followable security. Along the way, we have stated other definitions of a security ceremony which have been based on our research, each of which is useful in building for
others an understanding of the significant aspects of a security ceremony.

The first definition of a security ceremony was that a security ceremony is a protocol in its context of use. The implications of this definition are significant. We can no longer simply create proofs of a protocol and hope that a protocol proven secure can then be used in any context and remain secure. A similar example for ease of understanding to a cryptographer who may think that “a protocol proven secure can then be used in any context and remain secure” is exactly what a security proof should mean, is the case of a block cipher proven secure. No-one claims that simply by employing a proven secure block cipher every protocol that uses that block cipher will be a secure protocol. The block cipher may remain secure, while the protocol that employs it may not be secure.

A second implication of this concept that security ceremonies are protocols in their context of use, stems from what we have learnt from our research and what was already available in other research. That is, no design is ever used as intended. We examined what it meant for a group to develop a collective information practice, that is, their way of using the design. We have gone further than this, and highlighted that ingredients for future uses for the design may not exist yet. The impact of this realisation is that hoping to show that a given protocol would be secure for the uses of the protocol would be flawed, because the list of uses would not be complete yet. Therefore, from this point of view, all that ceremony analysis does is look at a protocol in a single context of use, and the best that can be hoped for is that the protocol is secure for that particular context of use.

The next logical definition of a security ceremony, which we have defined as a protocol it its context of use, is that a protocol with additional steps (context) added to it is simply another, higher level, protocol. As such, security ceremonies are nothing new, and perhaps no new tools are required to analyse security ceremonies simply because it is called a security ceremony. Existing protocol analysis techniques may be sufficient. However, if existing protocol analysis techniques are to be used, then new aspects of the model will be required, new security definitions and concepts will be required, and new tools to deliver the new security properties will be required. That is what we have provided by increasing the modelling capabilities by formalising human recognition, by creating new security definitions such as human-perceptible freshness, and the new tool we have developed to deliver human-perceptible freshness is the Computer-HUman
8.1. Conclusion

Recognisable Nonce (CHURN).

8.1.2 Formalising Human Recognition

This work presents a method of accumulating data which will allow for the comparison of schemes in which the human will need to recognise some data. When represented formally using our technique, schemes can be compared using: the size of $HPASpace$ (maximise), $HPASpace_H$ (maximise) and the ratio between the two (bring to equality); the size of $W_{H,HPA}$ (false positives, minimise); the size of $\epsilon$ (false negatives, minimise); and the frequency of use distribution (normalise).

We have provided an upper bound on the adversary’s probability of success, both for the case of a human generated $HPA$ and a device generated $HPA$. We have shown how our formalism may be included easily into existing proofs, providing a more complete model in the case of mutual authentication over TLS, and creating a formal proof of human-assisted device pairing protocols to be created for the first time. Many similar examples of protocols involving humans where our formalism will be directly useful exist. Such an example would be standard Verified-by-Visa protocol implementations, where, due to the large numbers of people and the large numbers of protocol runs, useful values for each of the variables in our formalism will be available. At the softer end of the scale, our formalism could be applied to human protocols which exist completely in the human realm, for example where a human may have to authenticate themselves to another human which is typically based on some sort of recognition.

8.1.3 CHURNs

We have shown that sequences created by humans using our CHURN-generator are significantly more random than sequences created by humans without the aid of a CHURN-generator. We have shown this by compression file sizes always being smaller for the unaided human, entropy values being lower, histogram values showing not all characters are used, that certain characters are used excessively, and that certain longer sequences of characters were repeated. This increase in randomness is despite a third of the participants thinking the CHURN generator was either the same or less random compared with their typing on a keyboard.

We have shown there is a significant second source of randomness due to the human input to a CHURN-generator via the Hamming Distances and how close
the Hamming Distances were to a true Binomial distribution. This means firstly that humans gain control over their CHURN, and secondly, if ever the seed value for the random number generator becomes known, the CHURN still has some level of security.

Finally, more than two thirds of our participants reported feeling more in control using the CHURN-generator to create a random sequence than being given a random sequence to use by a computer. This confirms the final hypothesis that humans will feel more in control using the CHURN-generator than being given a random sequence.

### 8.1.4 Human-Perceptible Freshness

When examined from an abstract perspective, the major points made in the original security ceremonies paper [41] may be interpreted as “there must be human followable security.” This message has flowed through our own human studies and analysis of security ceremonies. A clear and significant step towards human-followable security has been made with human-perceptible freshness.

We have shown the need for human-perceptible freshness. A critical security goal of standard cryptography is the concept of freshness, however this aspect is not employed in protocols involving humans, in a way that is followable by the human user.

We have defined human-perceptible freshness as a security goal. Protocols can now be developed which will satisfy this security goal.

We have created a compiler, which may employ the CHURN presented in Chapter 6, which delivers a protocol with human-perceptible freshness. The compiler takes as input an ACCE protocol, realised in the real world by the TLS protocol, and outputs a protocol which also satisfies the human-perceptible freshness property. We have called this resulting protocol a H-ACCE protocol.

### 8.2 Future Work

Here are some directions to take this research which we believe are meaningful and promising. Of course the ultimate goal is to create secure cryptographic protocols which include humans, and have those protocols implemented and used in the real world, thus giving human users the security assurances they hope they have now but do not.
8.2. Future Work

8.2.1 CHURNs

Significant further research and development must be conducted once the CHURNs are used in protocols by the general public. It is envisioned that the progress of CHURNs will be similar to the progress of CAPTCHAs, ”Completely Automated Public Turing test to tell Computers and Humans Apart” [120]. That is, CAPTCHAs, used widely in the Internet, have changed significantly over the past 10 years as actual usage by humans has been researched and understood, and as attacks using machines have developed. Therefore, continued research on CHURNs will be required as implementations are developed and used.

8.2.2 Formalising Human Recognition

Although our formalism makes significant steps forward for provable security in the presence of humans, as with any modelling of humans there are limitations. The most significant limitations are firstly, as already discussed, for the purposes of a security proof $A$ cannot select a $HPA'$ that is outside of $HPASpace$. For example, if we define $HPASpace$ as the set of 16 pixel black and white images, many humans may not reject images of a size other than 16 pixels, such as 25 pixels, which would be outside scope of $HPASpace$.

Secondly, we have not included the concept of context in the formalism, and as such the $Recognise$ function presented is not probabilistic. Much has been learnt in recent years about context, perhaps best summarised in Dourish’s paper “What we talk about when we talk about context” [36]. Dourish writes of the deficiencies in addressing context as a static representational problem. For example, a context rule may be that personal telephone calls should be ignored at work (i.e. in the “work context”), and in general, that rule would work and could easily be implemented on a smart phone, but that should not be the case if the call is regarding a significant problem with the worker’s child. Certain entities that users log into or use regularly, such as Facebook and Google, will have enough information to construct meaningful $HPASpace_H$’s for individuals. Therefore meaningful data will exist for the distinction we have made in this paper between $HPASpace$ and $HPASpace_H$, and hence trawling and targeted attacks. However, we leave as significant and separate future work, the concept of context. Ideally the use of context will be in regards to an “occasioned property that arises from the activity” rather than a “stable representational” view of con-
text. Context will impact both the Recognise and GenHPA functions, and getting meaningful data allowing for correct modelling on a specific-to-each-human basis will be difficult.

8.2.3 Human-Followable Security

Existing theory on protocol design for cryptographic authentication protocols states that at a minimum we need three aspects in creating a secure protocol. Firstly we need to specify who the parties communicating are, secondly we need some assurance of freshness, and thirdly we need a way of binding that information together such that it cannot be changed and keeping it confidential (see Section 2.4).

While requirements may change depending on definitions of security used and required, these three ingredients seem critical in creating secure human protocols. In Chapters 6 and 7 we have presented a method for providing one of the critical ingredients: an assurance of freshness. Due to the properties of ACCE, meaning we have a channel that already provides authentication and confidentiality, the human receiving their fresh message back from the party they are communicating with also provides authentication. While this authentication is via the use of a human perceptible authenticator HPA, this is not human perceptible authentication. That is, since the guarantee that the other party must be who the human expects is based on a process that is not human-followable (such as HTTPS), there is not a human-followable property which guarantees that the other party must be who they claim to be.

Therefore, as future work, still to be provided are the two other ingredients, being the authentication messages and the method of binding the information together in a human followable way.
Bibliography


[82] Steven Murdoch and Ross Anderson. Verified by Visa and MasterCard SecureCode: Or, How Not to Design Authentication. In Radu Sion, editor,


