Department of Computer Science

Submitted in part fulfilment for the degree of
MSc in Cybersecurity.

Investigating the Security of p≡p 's Trustword PGP Fingerprint Encoding

Aidan Fray

DRAFT PROCESSED 12th August 2019

Supervisor: Siamak Fayyaz Shahandashti

Contents

List of Figures

List of Tables

1 Introduction

The increasing use of public key cryptography by instant messaging and secure email means key fingerprint verification is an ever more important task.

One of the most significant risks to the security of the communication channel is a Man-in-the-middle (MiTM) attack. A MiTM attack involves an attacker impersonating one or both sides of a connection. MiTM attacks can fully circumvent the encryption as it allows an attacker to read all of the encrypted data.

A countermeasure for this is the verification of each parties' key fingerprint. A fingerprint is a small string of characters that is unique to each key pair. Therefore, it can be used as a way of identification.

Fingerprints can come in a number of different encodings such as hexadecimal, words and even procedurally generated avatars. Previous research has shown that the average human can only hold around 7-digits worth of data in their working memory[1], alongside this the involvement of human based verification makes the designing of user friendly schemes a task of upmost importance.

Humans have long been known as the most vulnerable part of any system. To combat this issue, solutions have been proposed to remove the manual verification with examples such as PGP's web-of-trust[2] and Namecoin[3]. However, these suffer from user adoption due to perceived complexity. This leaves manual verification in a difficult position, due the proposed solutions needing to sacrifice either usability or security. Therefore, research into improving or creating secure fingerprint encodings that work well with users remains massively relevant.

The report will asses the security and feasibility of attacking a fingerprint encoding known as "Trustwords" as justification for various design choices have been unsubstantiated.

The report will be structured as follows:

Chapter 2 provides any required background knowledge alongside a review on all relevant literature.

Chapter 3 formally defines the aim of this report alongside the research questions the project aims to answer.

Chapter 4 documents the proposed designs required to answer the research questions proposed in Chapter 3.

Chapter 5 discusses the significant implementation details of the designs formalized in Chapter 4.

Chapter 6 presents and analyses all experimentation results of the project.

Chapter 7 evaluates the success of the project alongside a discussion into relevant further work. The research aims proposed earlier in Chapter 3 will be used as evaluative metrics.

Chapter 8 concludes the project with a culmination of the papers findings alongside any conclusions that can be drawn from the research.

2 Background

The chapter explains aspects required to fully comprehend the project's achievements alongside an evaluation of current literature and the respective gaps.

The chapter will contain the following sections:

- **Project context**
  Will provide the user with the base knowledge required to understand the context and purpose of the following literature review.
- **Definition of terms**
  Specific definitions relevant to the literature review will initially be defined.
- **Literature Review**
  A review of currently available literature on relevant topics. Recommendations and research gaps will be identified and discussed.

2.1 Project context

To fully understand the literature review it is necessary to introduce and explain the base knowledge before continuing.

Public-key Cryptography and Key Exchange

Asymmetric cryptography facilitates the secure encryption of messages in end-to-end encryption (E2EE), verification of digital signatures and sharing of pre-communication secrets, among others.

The asymmetry stems from the use of "Public" and "Private" keys. The Public key is used to encrypt data that only the respective Private key can decrypt. This means the keys required to encrypt can be easily shared across insecure channels.

To construct an E2EE connection the initial stage will be a key exchange using the key types discussed previously. This is the sharing of a pre-shared secret between two verified parties, this will then be

used to encrypt subsequent messages with symmetric encryption[1]. This, however, hinges on the verification of the initial parties, if one party impersonates another and gets to view the pre-shared secret all communication becomes decryptable. Therefore, the correct identification of parties is crucial to maintaining secure communication channels. The exploitation of this is known as a Man-in-the-middle (MiTM) attack. This can be bidirectional where the attacker can sit in the middle of a communication channel and decrypt all correspondence. Figure 2.1 contains a visual representation of this attack.

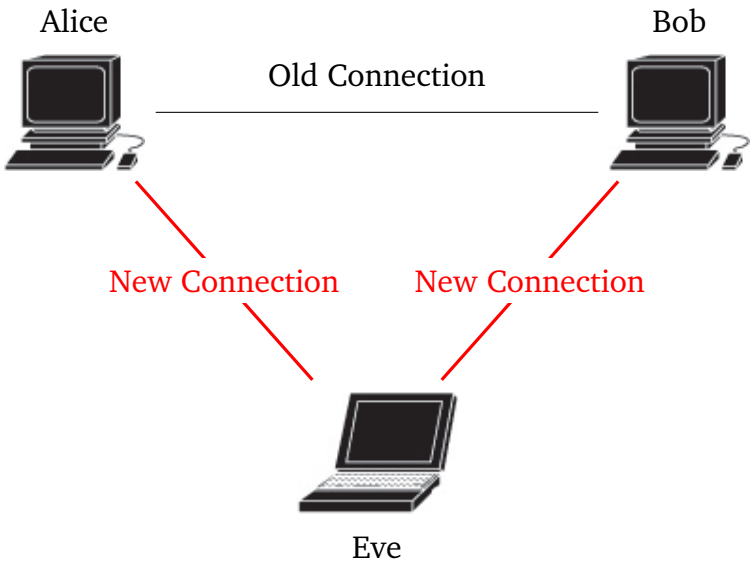

Figure 2.1: Photo depicting a MITM attack

Due to the assumption that the attacking party (Eve) does not have the same public-private key pair as either party (Alice or Bob) the fingerprint of the public key can be used for identification. There are, however, issues in how a user should link a key's fingerprint to a real world entity. This paper will, however, assume that users have access to the respective parties' correct fingerprint, as the protocols used to achieve this are outside the project's scope.

The fingerprint of the key is generated by running the main key components through a secure one way hash function such as SHA-1. This process produces a digest of fixed length that can be used to compare keys. Therefore, the comparison of expected and actual fingerprints can be used to detect MiTM attacks. Historically these have been represented as a hexadecimal string whereon verification fingerprints are compared between two substrates, for example, a monitor screen and a

[1]This is due to the speed increase of symmetrical over asymmetrical encryption

business card. This process is known as the *"authentication ceremony"*. However, system designers are moving away from this structure and are now creating a combined key from each sides' public key. In the case of WhatsApp for example, the connection fingerprint of two parties keys is the first 30 bytes of a SHA-512[2] hash of each parties' identity key, these are then both concatenated together to form a single key[4]. This produces a unique key communication pair.

Due to the manual comparison of the fingerprint, length and format is a consideration. Prior research has shown that the average human can only hold around 7-digits worth of data in their working memory[1]. This rules out the possibility of comparing complete digests. For example, SHA-1 is 40 hex digits (160-bit) and, therefore, difficult to effectively compare. Therefore, if human interaction is required; there is a need for schemes that work effectively with consideration to individual limitations.

Research in this area has taken various encoding schemes and compared their fallibility to impersonated attacks. The main elements of comparison have been the "accuracy of attack detection" and "time to compare". In this paper, these are considered the metrics of "security" and "usability", respectively.

2.2 Definition of terms

Throughout the paper, various terms are used to define elements of the authentication ceremony relevant to fingerprints verification. Explicit descriptions are, therefore, stated in this section. These are used to remove ambiguity and underlying associations readers may have for the words chosen.

Encoding schemes - This is the physical method of encoding used to represent the fingerprint. For example, "Hexadecimal" or "Words" are a unique way to represent a fingerprint and thus, are encoding schemes.

Method of comparison - This is how the user assesses the encoding scheme. The most common example and the one used as default is "Compare-and-Confirm" (CaC). CaC, as the name suggests, is the *comparison* of two fingerprints on different devices or mediums that are then *confirmed*. The following sections explain alternative methods of comparisons.

[2] 5200 iterations

2.3 Literature Review

This section will now discuss the current research around the proposed topic.

The section is organized as follows:

- **Authentication ceremony performance**
  This section explains the current research on performance of various encoding schemes and methods of comparison. This is then complemented with an assessment of the experimentation design for the respective studies.
- **Encoding schemes**
  The following section then discusses research around the design of an encoding scheme. Focus in this section will be allocated to the technical details around the creation and design of relevant schemes.
- **Attacks on encoding schemes**
  The final section will then discuss literature that investigates the creation of attacks against encoding schemes in order to deceive the respective users.

2.3.1 Authentication ceremony performance

Encoding scheme performance

Results from the literature consistently show the effectiveness of language-based encodings such as Words or Sentences with accuracies ranging from 94% [5][6][7]. In all cases, these were the best schemes from the sets assessed. The exception to this is the work performed by **Hsiao, *et al.*** [8] in 2009 with Words achieving an abnormal accuracy of 63.00%.

Aside from textual representations were graphical schemes. Examples of schemes assessed were: Random Art[9], Flag[10], T-Flag[11], Vash[3], OpenSSH visual host key and Unicorns[4], among others. These schemes had mixed accuracy with ranges as large as 50% - 94% in work by **Hsiao, *et al.*** [8]. The only other paper assessing graphical representations was the work of **Tan *et al.*** [6] where they also achieved mixed results with accuracies ranging from 46% to 90%.

In terms of usability of graphical schemes, the literature concurred on

[3] https://github.com/thevash/vash
[4] https://unicornify.pictures/

Scheme	Hsu-Chun[8]	Kainda[7]	Dechand[5]	Tan[6]
Hexadecimal			11.20	9.00
Numerical		6.00	10.60	9.00
Base32	3.51	6.00	10.20	
Words	4.63	7.00	8.70	7.00
Scentences		11.00	12.30	8.00
Chinese Symbols	5.01			
Japanese Symbols	5.07			
Korean Symbols	4.92			
Random Art	3.21			
Flag	4.28			
T-Flag	4.00			
Flag Ext.	4.02			
OpenSSH‘				5.00
Unicorns				3.00
Vash				3.00

Table 2.1: Timing results in seconds for the related schemes

their high usability. The comparison speed of these schemes were all among the quickest (See Table 2.1 for an overview of timings). Other work also indirectly concurred with graphical encodings having significantly quicker comparison times compared to non-graphical schemes [5][7]. In terms of research into the performance of graphical schemes, the literature does not contain an extensive review with literature only containing two papers. There is also no overlap in the schemes assessed with each paper reviewing a unique set. This is, therefore, a promising candidate for further research.

One unique paper in this research area was the work by **M. Shirvanian *et al.*** [12] produced in the context of secure messaging pairing. This paper was unique for several reasons. First was to consideration for “remote-vs-proximity” pairing where this is the first consideration of this aspect found in the literature. There is room for further research to compare encoding schemes in the context of “remote-vs-proximity.” Another unique aspect was the end-to-end encryption context of the study.

The findings from the paper showed a high false negative rate for all the schemes; this is a consideration also missing from the literature. Alongside this, results for usability were lower in a remote setting for all the schemes. The author, however, comments on the expected nature of this result.
Images were assessed as being the most secure method of authentication in the remote setting but voted as the method with the worst usability.

This conclusion is highly inconsistent with all other work in this area. However, this could be due to the unique setting of remote verification, resulting in distinct results from user studies. Without further work in this area, it is difficult to validate these results conclusively.

Alongside this, it was shown in work by **Hsiao, *et al.*** [8] that age and gender do not affect the accuracy of the scheme. However, younger participants were considerably faster. Furthermore, findings also showed that language comprehension helped in discerning small differences between schemes encoded in Chinese, Japanese, Korean or English. Subsequently, knowledge of the language did not assist in differentiating more significant changes in the schemes as these had high accuracy regardless. These were interesting and unique considerations. Further work could aim to corroborate these conclusions.

Scheme	Hsu-Chun[8]	Kainda[7]	Dechand[5]	Tan[6]	M. Shirvanian[12]
Hexadecimal			89.56%	79.00%	
Numerical		100.00%	93.66%	65.00%	97.33%
Base32	86.00%	90.00%	91.50%		
Words	63.00%	100.00%	94.25%	94.00%	
Scentences		100.00%	97.01%	94.00%	
Chinese Symbols	59.00%				
Japanese Symbols	57.00%				
Korean Symbols	54.00%				
Random Art	94.00%				
Flag	50.00%				
T-Flag	85.00%				
Flag Ext.	88.00%				
OpenSSH‘				90.00%	
Unicorns				46.00%	
Vash				88.00%	

Table 2.2: Accuracy of correct comparison for the encoding schemes assessed

Tables 2.1 & 2.2 contain the accuracy results of all papers assessed; this is to aid in visual comparison. Each paper used a different metric for measuring accuracy. Therefore, all results have been translated into “overall accuracy.”

Methods of comparison performance

Aside from “Compare-and-Confirm” (CaC), there is “Compare-and-Select” (CaS) and “Compare-and-Enter” (CaE). CaS is the method where one

device displays the fingerprint, and the other user is provided with several options. The user then has to choose the correct value from the list of candidates. If there is no match, the user must deny the connection attempt. The creation of CaS was due to concerns that CaC would be "too easy" for users leading to complacency and errors[13]. CaE is designed for scenarios where both devices might not have a display, i.e. pairing between a phone and a keyboard. One device displays the checksum. This checksum is entered into the other device. The first device then compares the entered string and checks for a match.

Research has been performed assessing the performance of these schemes and how they affect the ultimate security of the authentication ceremony. The literature agrees on CaC being the best overall scheme to compare fingerprints [6][13] with CaS being highlighted for its poor security and usability. CaE has had contrasting results. [13] discarded it after one round due to "poor usability." However, [6] considered it the best method overall for usability and security. These conflicting results, therefore, show polarisation in the results of CaE. However, this could be due to the different overall use-cases of the studies. Validation of results from either study would, therefore, be an area of additional study.

Experimentation methodology comparison

To further look into the validity of previously discussed results, it is necessary to asses how the respective studies reached these conclusions. Areas for consideration are scheme entropy, attacker strength and participant attributes.

The starkest limitations of the literature are the range of participants and encoding entropy. The worst studies tested only 22-bits of entropy. This makes it difficult to compare results directly. One of the papers this most effects is the early work by **Hsiao, *et al.*** [8] where their highest entropy is 28-bits. This level of entropy was inadequate even at the time of publication. There is an attempt by the authors' to address this issue in the later stages of the paper where they write *"[...] increasing entropy is not a solution because it sacrifices usability and accuracy. With more entropy, representations will contain longer sequences of characters or more minute details which will lead to increased time and errors during comparisons."*. This statement is backed up with no empirical evidence, and the authors' fail to consider how the low entropy would affect the overall security of the schemes.

Attacker strength is also another metric used to compare results concluded by each paper. Some papers failed to address their attacker strength consideration directly, but the overall strength has been inferred from the changes made to their schemes. For example, if they

decided to change a single character in a 40-digit (160-bit) SHA-1 hex digest, they are indirectly stating that the attacker can control 39-digits (156-bit). To achieve this, the attacker would have to compute 2^{156} SHA-1 compressions to find a key match. In the literature, this element ranged from 2^{28} to around 2^{242}; however, this has some relation to the size of the encoding schemes used. This is a substantial range that makes it ultimately challenging to compare and confer results confidently. Further work could exclusively look into the affects attacker strength has on the success of attacks. Moreover, all the papers assessed failed to fully consider the feasibility of attacks in terms of computer and storage requirements. This again, shows gaps in the literature.

All of the studies considered demographical data when presenting their results. Their average ages were all around ~ 35 years old with the majority of participants educated with at least a bachelor degree. This was alongside the equal split between male and female participants.

One consideration of note is that made by Dechand *et al.* [5] where they briefly consider medical conditions such as ADHD and reading or visual disorders and the way they affect the comparison's effectiveness. They highlight a slight reduction in overall accuracy, although due to their small sample size, they cannot conclusively validate these results. This is an aspect unique to all literature. Further work would be required to produce conclusive results. Therefore, highlighting a gap in the literature.

One glaring issue with the demographical health of **Ersin Uzun *et al.*** [13] study is the use of two entirely different groups of participants. Not only were their demographics different, but they were from different countries (America and Finland). Different cultures contain inherent biases and assumptions. Moreover, more concerns are raised around the dual study design of this paper with two completely different set of participants. Changes were made pragmatically regarding the results from the first round of 40 participants. These were then re-assessed and the results were directly compared. This puts huge doubts on the validity of the results with no consideration made by the authors to control external factors that may affect the performance of the method of comparison.

Hexadecimal and numerical schemes were not included as encoding schemes in work by **Hsiao, *et al.*** [8] (some of the most widespread encoding schemes). This decision was based on their claims on similarities to Base32 and "well-known deficiencies" in the excluded schemes. This point was provided with no further justification or quantified in any way. It is also not consistent with available research, for example, in *"Empirical Study of Textual Key-Fingerprint Representations"* [5] it was shown numerical representations performed significantly better than

that of Base32.

	Hsu-Chun[8]	Kainda[7]	Dechand[5]	Tan[6]	M. Shirvanian[12]
Attacker Strength[5]	$\sim 2^{28}$	$\sim 2^{40}$	2^{80}	2^{60}	$\sim 2^{242}$
Entropy Range	22-28 bits	20-40 bits	122 bits	128 bits	160-256 bits
No° Participants	436	30	1001	661	25

Table 2.3: Paper attribute comparison

Table 2.3 has been provided to visually compare the differing aspects of the papers' parameters. Clearly it can be seen from the table the large ranges in participant size, entropy and attacker strength.

Topic conclusion

Overall, this review has identified several key areas suitable for further work. The first is the performance assessment of graphical encoding schemes. Recreation of pre-existing results from [8][6] is required to corroborate current conclusions and validate results. Another gap in the research is the consideration into the utilization of encoding schemes in realistic conditions, i.e. “remove vs. proximity." This topic was initially covered by [12] but their scope was limited. Further work, therefore, could increased the scope and touch upon a large number of schemes in these settings. The final aspect for further work is the limited consideration into the feasibility of attacks on encoding schemes. All of the papers assessed simulated attacks and had minimal consideration for the execution of these attacks. Therefore, further work could delve into the implementation of such attacks and their feasibility in terms of compute and storage complexity.

2.3.2 Encoding schemes

Another area of research is investigations into the actual physical encodings of the hash digest. This section will briefly discuss the current research available on the creation and security of actual encoding schemes. The actual details of the operation of the schemes are outside the scope of this literature review, therefore, minimal attention will be allocated to these details.

Some of the oldest preliminary work into visual encoding schemes was performed by **Adrian Perrig *et al***[9]. in the creation of their scheme “Random Art" in 1999. The motivation for creating such a scheme was the perceived flaws in the ways humans verify and compare written

information. As mentioned in previous sections visual encoding schemes have been shown to have mixed success, with low security being one of their most alarming flaws. This research laid the foundation for further work in analysing the security of visual encoding schemes.

Further research into the creation of unique visual hash schemes have been performed by **C Ellison *et al.*** [10] (Flag), **Yue-Hsun Lin *et al.*** [11] (T-Flag) and work by **M. Olembo *et al.*** [14]. Each publication has provided a new way to visually represent a key fingerprint. Alongside the academic literature, there are more informally presented methods of visual fingerprints such as Unicorns[6] and Robots[7]. This list is by no means exhaustive but is used to depict the amount of research and work invested into graphical hash representations.

One paper of note is the preliminary work performed by **D Loss *et al.*** [15] in their *"An analysis of the OpenSSH fingerprint visualization algorithm"* where their aim was to spur on further research with their initial findings into the security of the OpenSSH scheme. The authors claim that the use of the algorithm in OpenSSH is only heuristically defined and there is a need for a formal proof of its security.
The paper proposed a number of ways to generate similar fingerprints. The methods proposed were: Naive brute force, Graph Theory, and brute force of a full visual set. They were only able to produce only very basic results and have proposed a large amount of potential further work. Since the paper's publication in 2009, there seems to have been no research building on the work of the authors. This highlights a current gap in the available literature.

Minimal research has also focused on basic textual fingerprint representations and their respective security. Work by **A. Karole** and **N. Saxena**[16] looked into ways to improve the security of a textual representation. This research aim was to improve the secure device pairing process of comparing two numerical values. The devices used (Nokia 6030b; Mid-range devices at the time of publication) and the SAS compared results in findings that are not directly applicable in a fingerprint comparison context.

A more specific subsection of textual fingerprints is the use of words and sentences to encode hash digests. Some of the first work in this area was produced by **Juola** and **Zimmermann** [17] and their work in generating a word list where phonetic distinctiveness was prioritised. Each word is mapped to a single byte. The unique aspect of the word list is the separation of "even" and "odd" words where "even" byte positions are sample from the even-list and "odd" from the odd-list. This effectively creates two sub-word lists. The maximisation of linguistic

[6] https://unicornify.pictures/
[7] https://github.com/e1ven/Robohash

distinctiveness of these word lists were maximised through the use of a Genetic algorithm. The paper also includes a study on effective measures of "linguistic distances" of words and provided an in-depth discussion into these areas.

Overall the paper provides a foundation for formalising the creation of effective wordlists. A limitation is the lack of empirical data gathered on the performance. However, this was later evaluated in work by Dechand *et al.* [5] and shown to be an effective encoding scheme.

Other research of note is work by **M. Goodrich *et al.*** [18] called *Loud and Clear: Human-Verifiable Authentication Based on Audio*. As the name suggests the authors were researching ways to improve current methods of secure device pairing. The unique aspect of this work is the use of a Text-to-Speech system reading out syntactically correct English sentences. The sentences are based on a MadLibs[8] where static placeholders were replaced with potential words.
The work into a potential wordlist can be seen as an extension to the work performed by Juola and Zimmermann[17] as they aimed to emulate the techniques used in PGPfone. The paper's finding are limited by the lack of empirical data backing up claims made by the author as the systems performance and security are only theoretically assessed.

Aside from this research, there have been further informal implementations of fingerprint encodings. The first being by **Michael Rogers**[9]. Rogers' implementation is a program designed to map fingerprints to pseudo-random poems. This implementation was again, empirically evaluated by Dechand *et al.*[5]. Older work by **N. Haller** with the S/KEY[19] shows the implementation of a system designed to represent a hash as a series of six short words. However, this system is designed for a one-time-password purpose and only provides word mappings for basic human usability of the password and not within a fingerprint verification context. Therefore, the wordlist has not been designed with pronounceability in mind.

Pretty Easy Privacy

A very recent implementation of a word list can be found in Pretty Easy Privacy (p≡p) implementation of TrustWords[10]. p≡p is a data encryption system that utilises PGP encryption to provide E2EE on all common channels of communication such as email or SMS. The imbedded design principles state that above all the systems should be

[8] https://en.wikipedia.org/wiki/Mad_Libs
[9] https://github.com/akwizgran/basic-english
[10] https://tools.ietf.org/html/draft-birk-pep-trustwords-03

easy to install, use and understand.

p≡p deals with the threat of MiTM attacks by having users compare the respective key fingerprints encoded as a set of words. Figure 2.2 shows the p≡p Android implementation of Trustwords. The users then authenticate the words on an OOB (Out of Band) channel such as a phone call or in-person communication. If both users decide the words match they will accept or decline respectively.

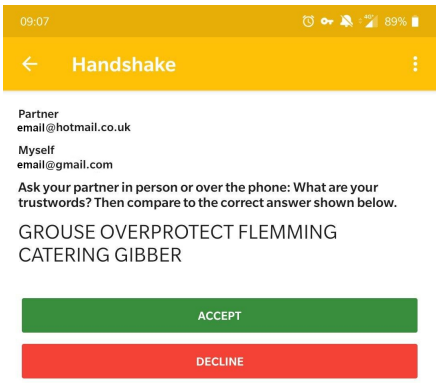

Figure 2.2: Trustword fingerprint verification

The unique aspect of TrustWords is its mapping of a single word to 16-bits. In comparison to other literature, this is the highest number of bits-per-word seen. Full mappings (no duplication of words) would, therefore, require 2^{16} words in the dictionary where arguably the dictionary size is higher than most users' vocabulary. This deviation from the norm has not been currently backed up by research.

```
    [...]
52127 ZYGOTE
52128 ZYGOTIC
52129 ZYMURGY
52130 AACHEN
52131 AARDVARK
52132 AAREN
    [...]
```

Figure 2.3: Re-mapping position in Trustword dictionary

Alongside the abnormally high number of words the design choice to exclude slang and profanities from the English Trustword dictionary requires dual-mapping of a section of words. Approximately 13633/65536 (20.8%) of words are re-mapped in the dictionary leaving 51903 unique words. The re-mapping is also done on a loop with it remaining alphabetical. Figure 2.3 shows the position in the dictionary where this occurs. This predictability within the dictionary will be explored later in the paper.

Moreover the main RFC documentation still remains in a draft stage and states *"It is for further study, what minimal number of words (or entropy) should be required."*. These aspects clearly highlight on a gap in the current literature.

Topic conclusion

In conclusion to this topic, the current research has primarily focused on the research and creation of visual representations. Research for textual fingerprints is fragmented and incomplete with work Juola and Zimmermann [17] and M. Goodrich *et al.*[18] providing meaningful research to build upon in terms of word a sentence based encodings. The fragmentation of this research leaves room for further work into this topic area. Alongside this, findings from the previous sections research shows that human language based encodings provided the best usability and, therefore, should be a target for further research looking to improve upon their security and usability.

2.3.3 Attacks on encoding schemes

This area of research studies ways to physically execute attacks on fingerprint encoding schemes. This differs from previously examined work due to papers discussing the performance and fallibility of encoding schemes simulated the attack without consideration for how the attack would be performed. Research in this area is scant, with lots of research attention being directed towards the security of Man-in-the-Middle (MITM) attacks and not the encoding schemes themselves.

Research in 2002 by **Konrad Rieck**[20] is the first formalisation of attacks on fingerprint representations. The paper titled *"Fuzzy Fingerprints Attacking Vulnerabilities in the Human Brain"* aimed to look into ways users check hexadecimal encoded OpenSSH fingerprint representations. The author created an elegant way to 'weight' more important chunks of the digest. The bytes furthest to the right and left of the digests provided the highest weight. The weight was the smallest in the centre of the digest. This provides a way to score digests and determine the best partial collisions found. For example with the target fingerprint: `9F23` a partial match `9313` is given a score of 45% even though only two characters were matching. This is due to the weightings.

The paper contains an implementation with a "1.2GHz CPU" being able to obtain 130,000 H/s (With MD5). In comparison to this, a mid-range Intel i5-3320M CPU can today obtain 111,700,000 MD5 H/s. This shows that the results obtained from the paper are significantly outdated. However, even with the low hash rate, the author was able to obtain some promising results. Figure 2.4 contains the best example used.

Overall the paper shows an interesting way to create partial fingerprint matches but is not quantified by any empirical evidence gathered on real world users. This, therefore, highlights on gaps in the coverage of

```
TARGET: d6:b7:df:31:aa:55:d2:56:9b:32:71:61:24:08:44:87
MATCH:  d6:b7:8f:a6:fa:21:0c:0d:7d:0a:fb:9d:30:90:4a:87
```

Figure 2.4: Best match obtained after a few minutes of hashing

this literature.

The only other relevant research on this topic is the work by **M Shirvanian *et al.*** [21] and their paper *"Wiretapping via Mimicry: Short Voice Imitation Man-in-the-Middle Attacks on Crypto Phones"*. Further research in the area of "human voice impersonation" has received lots of attention [22][23][24]. This paper was chosen over other alternatives due to is specific use of encoding schemes in its evaluation.

In this paper, the authors develop a way to impersonate users when authenticating Short-authentication-Strings (SAS) in pairing of Cryptophones. To achieve this impersonating they propose two methods: "Short voice reordering attack" where an arbitrary SAS string is recreated by re-ordering snipped obtained from eavesdropping a previous connection and "Short voice morphing attacks" whereby the use of previously eavesdropped audio snippets the attacker can morph their own voice to match that of the victim. With these methods, they aimed to attack encodings of Numbers, PGP word list (previously discussed work by Juola and Zimmermann [17]) and MadLib (M. Goodrich *et al.* [18] work also previously discussed). The effectiveness of these attacks were evaluated with a study involving 30 participants.

Results from the paper show the effectiveness of these methods. Compared to the baseline of the attacker's voice replacing the victim where this performed with a $\sim$18% success rate. Morphing gained an overall success rate of 50.58% and Reordering a very impressive 78.23% success rate. Showing that these attacks provide an improvement on top of the naive implementation.

One of the biggest limitations addressed by the authors was the reduction in success rates as the size of the authentication string grew. The morphing and reordering attacks become increasingly ineffective as the user has more time to detect imperfections. This is not quantified by the author and the extent of this degradation is never empirically discussed. Therefore, the results from this study are only effective and applicable in a SAS context.

Topic Conclusion

Overall the literature for this subtopic remains sparse and incomplete. Further suggested work could look into the feasibility of generating partial collisions for all textual representations alongside quantified effectiveness on users. With the possibility to concentrate on a few selected implementations. The work would aim to focus on the various physical methods used and their feasibility. This is one area the previous literature has failed to cover and has only theoretically quantified attacker strength without consideration for the actual real-world cost of these attacks.

2.4 Overall Summary

In summary of the literature gaps areas discovered; encoding scheme performance requires further work in the performance of all graphical schemes to back up the results made by other work. Furthermore, there is a gap with the assessment of encoding schemes in the context of "remote-vs-proximity" first proposed by M. Shirvanian et al. as it would arguably provide a better simulation of real world scenarios.

In the context of human attributes and the way they alter the performance of the schemes; further research is required into how the fluency of languages affects authentication ceremony. This is a continuation of the initial work by Hsiao et al and could allow the creation of schemes that are better suited to certain languages. Alongside this, there is a research gap in the investigation of how mental impairments affect the performance. This is useful due to the number of potential users being affected due to systems not being designed with them in mind. For example, 7% of the population has been identified as having dyslexic tendencies[25]. This means with a UK population of around 63,000,000[11] 4,410,000 people could benefit from encoding schemes including dyslexic oriented benefits. This is also only one of many visual impairments that could affect a scheme's performance, thus further contributing to the substantial affect further work could have on this area.

Other possible large areas for consideration is the lacks of direct consideration for attack strength when assessing the vulnerability of encoding schemes. Issues included the wide range of attack strength (2^{28} - 2^{242}) and the lack of consideration at all from certain papers. This alongside the complete lack of computation and storage complexity considerations means this is a prime area for further research as it would improve the applicability of results.

[11] From the 2011 Census collated by the Office for National Statistics

The final major research gap identified is the lack of justification into the newly created p≡p Trustwords. Abnormal design choices (16-bit per word) and its recent creation makes it another area for a wide variety of further research. This is also combined with the conclusive results from the literature on the effectiveness of language based encodings such as words. This would suggest that research into the improvement of Trustword encoding would be highly beneficial.

3 Project Aims

Following on from the previous chapter, this chapter will define the selected gaps and the subsequent research questions and aims.

The chosen project aims to asses the security of p≡p 's minimum recommend number of four Trustwords (As stated in Figure 3.1). As discussed in the previous chapter, Trustwords aims to sacrifice security for increased usability. The encoding scheme has been designed to assist this by having the user compare a reduced set of words. Moreover, issues with the dictionary's design such as the presence of homophones and dual-mapping of words shows the potential for possible vulnerabilities.

As discussed in the previous chapter, the identified research gap regarding the minimal consideration of attack complexity would be a suitable addition for this project. This is due to it providing the ability to asses actual performance and, thus, the actual real-world strength of Trustwords while providing research to a scant area. Therefore, the security assessment will be supplemented with complexity considerations and an actual implementation of the attack proposed.

```
"Short Trustword Mapping (S-TWM) requires a number of
Trustwords that MUST retain at least 64 bits of entropy.
Thus, S-TWM results into at least four Trustwords to be
compared by the user."
```

Figure 3.1: Most recent RFC security recommendation

By sampling the p≡p documentation they have defined the use case of the Trustword handshake: *A handshake is done by comparing the Trustwords between two users through a separate communication channel (e.g. in person or by phone)*.[1]. It can be assumed due to the increased globalization of the planet that the most common handshake occurrence will be over the phone, and, therefore, will not be in person. This is, therefore, the assumed context the handshake will be occurring in. This means the similarity of words will need to be quantified phonetically instead of visually.

[1]https://www.pep.security/docs/general_information.html#handshake

3.1 Research Questions

With the previously discussed points in mind the following research questions have been proposed.

1. Is the recommended minimum number of four Trustwords enough to provide a basic level of security?
2. What are the different ways phonetic similarity can be quantified?
3. Out of a chosen set of metrics, which are the most effective?
4. What is the time and computation complexity required to generate a 'similar' keys for a targeted key pair?
5. What kind of hardware is required to compute a matching key?
6. What percentage of attacks successfully deceive a user?

With these research questions defined, the project will now discuss the design of the proposed attack.

4 Design

This chapter will discuss and overview of the proposed designs for elements required to answer the research questions. Alternative solutions are explained when relevant alongside a justification of design choices.

4.1 Overall attack design

The attack on Trustwords involves generating "near-collision" keys. This will attempt to provide the base to answer Research Questions 1, 4 and 5.

Near-collision keys are keys that are deemed a match by the phonetic similarity metric (More on this aspect in the following sections). A similarity metric is an algorithm designed to determine if two words are phonetically a match. For example, the words "THEIR" and "THERE" will be a match whereas the words "DARK" and "PRINCIPLE" are not phonetically matching.

As each combined key in Trustwords is an exclusive-or of both sides' public key (Figure 4.1 shows this process) the attack is designed to target a single pair of users and will require recomputation for every attack target. This will be considered when discussing the attack feasibility. Each pair is also split into an "Uncontrolled" or "Controlled" key. Uncontrolled is the receiver of the communication, and, thus, we cannot control their key. The Controlled key is the one we are attempting to impersonate, and it is assumed that we have the ability to replace the Controlled key with our malicious option. These descriptions will be the terminology used throughout the paper. However, it should also be considered that the uncontrolled and controlled can be swapped around with the ability to compute both directions. Thus, resulting in the possibility to intercept both directions of communication. This, however, will require performing the attack separately for both directions.

$$KeyFingerprint_1 \oplus KeyFingerprint_2 = TrustwordsFingerprint$$

Figure 4.1: Creation of the combined Trustword fingerprint

When attacking the respective similarity metric will be used to compute

a list of possibilities for each position in the combined fingerprint.

Figure 4.2 shows the process of generating a subsection of the combinations. A list of each words' matches is generated, followed by the generation of the total number of near-collision keys.

```
CHOKE BLUSHING FRIGHTENING HAND
COKE
SMOKE

CHOKE
COKE BLUSHING FRIGHTENING HAND
SMOKE

CHOKE
COKE
SMOKE BLUSHING FRIGHTENING HAND
```

Figure 4.2: Visualization of the generation of near matches

In order to generate an actual list of fingerprints to search for, the near-collision words are converted back into hexadecimal and XORed with the uncontrolled key. This provides the impersonated key that will produce the desired Trustwords. Completing this for all of the near-collision permutations will produce a list of fingerprints that can be inserted into a tool designed to hash keys and search for targets. This aspect of using a large list to search for keys massively reduces the complexity of the search.

In summary, the attack steps are:

1. Compute all possible matches using a similarity metric on all words in a dictionary (Only needs performing once).
2. Select a target and allocate "Uncontrolled" and "Controlled" key identification.
3. Calculate all permutations of near-collisions for the key pair and produce a list of similar key fingerprints.
4. Use a list of similar keys in the mass computation of keys to find near-collision keys.

4.2 Similarity metrics

One of the first requirements for the attack is quantifying "phonetic similarity." This section is looking to provide an answer for Research

Question 2. There are many algorithms currently available to provide this functionality. This section will describe algorithms alongside the reasons they were selected for assessment later in the project.

4.2.1 Soundex

One of the earliest examples of a phonetic algorithm is known as Soundex. It is one of the most famous algorithm due in part to its implementation into major database clients like MySQL[26], Oracle[27] and PostgreSQL[28]. Originally designed for indexing names by phonetics alongside spelling mistakes due to transpositions in letters. Therefore, due to it being based on the phonetic features, it is highly suitable for this project's use-case.

Soundex produces a four digit code for each word assessed. The first letter of the word is retained alongside the removal of all of a, e, i, o, u, y, h and w. The remaining letters are then mapped to numbers[1]. These mappings are displayed in Figure 4.3.

```
b, f, p, v                1
c, g, j, k, q, s, x, z    2
d, t                      3
l                         4
m, n                      5
r                         6
```

Figure 4.3: Soundex mappings of letters to numbers

4.2.2 Soundex Issues

Due to the fixed length and limited digit set the initial concerns from this design is the limited number of combinations. There are a total of 5616 codes due to 26 initial letters and three digits of 6 values ($26 * 6^3$). The limited combinations will result in matches of a limited quality.

Further issues posed in [29] are discussed below and show the further deficiencies of Soundex in a dictionary word matching context.

1. **Dependency on the first letter:** Soundex cannot match words together if their first letters are different, meaning for example the words "KORBIN" and "CORBIN" will never be matched.

[1] Further steps are performed on the code, as the technical details do not add anything to the discussion, the steps of the algorithm can be found in Appendix TODO.

2. **Silent consonants.** Soundex does not have logic embedded to deal with silent consonants.
3. **Poor precision.** Due to the previously discussed point of a small code space. [29] re-iterates this point but in the context of name matching where Soundex's poor performance was demonstrated. Soundex only gained and overall accuracy of 36.37% when matching names within a provided database.

However, even with Soundex's fallbacks its algorithmic simplicity and popularity allows it to still remain relevant in this application.

4.2.3 NYSIIS

The New York State Identification and Intelligence System (NYSIIS) phonetic code was created for use matching the phonetics of American names. It was created due to the presence of hispanic names in the American based databases (this was an aspect Soundex was known to have low accuracy with). However, due to it having embedded rules to handle word phonetics it would, again be applicable in this application.

It also allows for variable length codes and, thus, allows the applicability of the application to increase due to it not confronting the limited code issue of Soundex. It was in use right up the end of 1998 within various US Government departments. Due to NYSIIS's aforementioned use in a professional setting alongside its high occurrence in literature, it was deemed suitable as one of the selected similarity metrics.

4.2.4 Metaphone

Metaphone was invented by Lawrence Philips in 1990[30] in response to the deficiencies in Soundex. It improves on Soundex by including information around inconsistency and variation in English spelling in an attempt to create a more accurate phonetic representation. Metaphone is arguably on the same level of ubiquity as that of Soundex with it finding itself implemented in languages such as PHP[31]. Metaphone was chosen due to similar reasons to that of Soundex and NYSIIS.

TODO Better argument needed?

Further work would involve the implementation of newer versions of Metaphone. Double Metaphone (2000) and Metaphone 3 (2009) that claim to improve over the original version due to further research performed by Philips.

4.2.5 Levenshtein Distance

Levenshtein distance is a string metric designed to measure the 'distance' between two strings. It is simply the number of single-character edits (insertions, deletions or substitutions) required to reach the other string. Edit distance is not technically designed as a phonetic algorithm, but due to similar-sounding words often being spelt in similar ways[32] Levenshtien distance was deemed another suitable metric. Levenshtein is also the most algorithmically simplistic of all the options, this further contributes to its appeal as a chosen metric.

An example distance between `trace` and `place` would be the substitutions of the first to letters, from `tr` to `pl` meaning the two strings have a Levenshtein distance of 2.

4.2.6 Phonetic Vectors

Phonetic Vectors is the unique addition to the chosen set. Created by Allison Parrish in 2017[33], Phonetic vectors is as the name suggests the vectorization of a words phonetics. This phonetical representation in vector space.

Phonetic features are used in this work as a way to compare the similarity of a words phonemes. Phonemes are the phonetic elements that construct a word. For example the word "RING" translated into the phonemes `/R IH NG/`.

Extensive prior work has gone into producing models of features that map to phonemes [34][35][36]. Features, therefore, are an attempt at mapping the varying and inconsistent rules around pronunciation of the English language. The vectors were created using lists phonemes from the CMU Pronouncing Dictionary and mapping them to possible features.

Phone	Features	Phone	Features	Phone	Features
AA	bck, low, unr, vwl	F	frc, lbd, vls	P	blb, stp, vls
AE	fnt, low, unr, vwl	G	stp, vcd, vel	R	alv, apr
AH	cnt, mid, unr, vwl	HH	apr, glt	S	alv, frc, vls
AO	bck, lmd, rnd, vwl	IH	fnt, smh, unr, vwl	SH	frc, pla, vls
AW	bck, cnt, low, rnd, smh, unr, vwl	IY	fnt, hgh, unr, vwl	T	alv, stp, vls
AY	cnt, fnt, low, smh, unr, vwl	JH	alv, frc, stp, vcd	TH	dnt, frc, vls
B	blb, stp, vcd	K	stp, vel, vls	UH	bck, rnd, smh, vwl
CH	alv, frc, stp, vls	L	alv, lat	UW	bck, hgh, rnd, vwl
D	alv, stp, vcd	M	blb, nas	V	frc, lbd, vcd
DH	dnt, frc, vcd	N	alv, nas	W	apr, lbv
EH	fnt, lmd, unr, vwl	NG	nas, vel	Y	apr, pal
ER	cnt, rzd, umd, vwl	OW	bck, rnd, smh, umd, vwl	Z	alv, frc, vcd
EY	fnt, lmd, smh, unr, vwl	OY	bck, fnt, lmd, rnd, smh, unr, vwl	ZH	frc, pla, vcd

Table 4.1: Phonemes to feature mapping table

Table 4.1 contains the mappings used in [33] to create the phonetic feature lists. Using this with all 133,852 entries in version 0.7b of the

CMU Pronouncing Dictionary, 949 unique properties were produced overall. The author then performed principal components analysis[2] on the unique properties to reduce them down to 50.

This metric allows for a unique set of actions to be performed on the phonetic output. Not only does this metric allow the user to measure *dissimilarity* (opposed to the similar-or-not method of the alternatives) the continuous nature of the value allows mathematical operations to be performed on the output. An example shown in [33] was the addition of word vectors.

No	Operation	Result
1	$Vec(\texttt{sub}) + Vec(\texttt{marine})$	`submarine`
2	$Vec(\texttt{miss}) + Vec(\texttt{sieve})$	`missive`
3	$Vec(\texttt{fizz}) + Vec(\texttt{theology})$	`physiology`

Table 4.2: Examples of vector addition

For example, the addition of vectors can be seen in Table 4.2. This works for any mathematical operation with examples like multiplication allowing the 'tinting' words with a theme.

The ability to perform operations and measure dissimilarity allows for a plethora of applications. One possible application of note would be the creation of a wordlist where the phonetic difference (vector distance) is maximized. This, if the vector mappings are an accurate representation could allow a creation of a phonetically distinctive wordlist. These unique aspects contributes to inclusion of this metric in the set.

4.3 Alternative Similarity Metrics

4.3.1 Match Rating Approach

Matching Rating Approach (MRA) is another algorithm designed to match names within a database, therefore, its operation can be grouped with that of NYSIIS and Soundex. MRA was discarded as an option. This is because of the lack of known utilization in any substantial real world use-cases alongside its similarity to more well established algorithms of Soundex and NYSIIS.

Further work could include the comparison of this algorithm to similar alternatives in phonetically matching of words to quantify performance.

[2]Details regarding this process is outside the scope of the project. Please, however, if interested please refer to this resource for more information: http://setosa.io/ev/principal-component-analysis/

This will be discussed further in the evaluative sections of the report.

4.3.2 Caverphone

Another notable alternative solution is that of Caverphone that was designed in New Zealand. Caverphone as is the case with the vast majority of phonetic algorithms was designed for use with name matching. Caverphone was not chosen to similar reasons to that of MRA alongside its optimization for the New Zealand accent. Therefore, the low level of utilization alongside the unique design features that suggests it may not be suited for this application. This has, however, not be assessed empirically and thus would be a candidate for further work.

Metric	Output
Soundex	T614
NYSIIS	TRAVAL
Metaphone	TRFL
Caverphone	TRF111
MRA	TRVL

Table 4.3: The various phonetic encodings of the word "Travel"

4.4 Design of GreenOnion

To answer Research Questions 4 and 5 fully, it is necessary to implement an actual code base that will be used to generate actual keys.

Inspiration of the design of this tool was taken from a tool called Scallion[3]. Scallion was designed by Richard Klafter and Eric Swanson and used to demonstrate that 32-bit PGP key ids were insufficient. Keeping with the onion-based theme the proposed tool is known as GreenOnion and is a re-write of the tools structure in C++. This language was chosen due to the well understood efficiency benefits. The proposed tool differs from Scallion most notably in its ability to concurrently search for a large number of keys, GreenOnion improves on this substantially. More implementation and experimental details will be discussed in later chapters.

The tool should take two keys as parameters (Uncontrolled/Controlled) and a chosen similarity metric and produce a list of target keys fingerprints. This list is then used as a search criteria when hashing a checking

[3]https://github.com/lachesis/scallion

a large number of keys. In order to utilise the high parallel nature of the GPU to compute the hash of a large number of keys the tool will utilise a GPGPU (General-purpose computing on graphics processing units) framework. The chosen framework was OpenCL due to its support for the chosen language (C++) and platform (Linux). OpenCL allows the creation of code chunks referred to as "kernels" to be executed concurrently, this will provide a massive speed increase compare to the sequential nature of the CPU. Technical details will be explained further in Chapter 5.

4.5 Experiment Design

In order to answer some of the research questions empirical evidence is required and, thus, experiments are required. In this section the designs and considerations of the chosen experiments will be discussed.

4.5.1 Metric performance

In order to answer Research Question 3 and reduce the number of metrics to assess in the later rounds, the performance of similarity metrics required comparison. The thinning out of metrics is required due to limited resources and, therefore, possible further work could involve repeating this work with a much more varied selection of metrics. As explained in a previous section (See Section 4.2) the chosen metrics up for assessment are Soundex, Metaphone, NYSIIS, Levenshtein and Phonetic vectors.

The design for the experiment involves assessing the quality of the metrics matches by having participants rate them on a scale of 1 to 5.

Matches

A match for each code based metric (Soundex, Metaphone and NYSIIS) occurs when the codes are identical. However, for the other cases where the difference is variable other ways of defining a match are required.

In the case of Levenshtein values of similarity are discrete due to it being the number of single character edits. Therefore, this requires less deliberation. Match size was used as a way to decide on a cut-off point.

As it can be seen from Table 4.4 the number of matches for L2 is massively larger than all metrics excluding Soundex. As the issues with Soundex have been discussed in previous sections (See Section 4.2.2) it

Metric	Matches
Soundex	1,527,554
Metaphone	412,916
NYSIIS	188,474
Levenshtein (Distance 1) [L1]	**97,730**
Levenshtein (Distance 2) [L2]	1,070,656

Table 4.4: Levenshtein number of matches comparison

can be excluded as an abnormality in this context. Therefore, due to the excessively high value for a L2, L1 was chosen as the Levenshtein cap for defining a match.

Match size was also used to define the cap for the phonetic vectors. This is less distinct that that of Levenshtein due to the continuous nature of the distance between two vectors, however the complexity was reduced by limiting it to increments of 0.5.

Metric	Matches
Soundex	1,527,554
Metaphone	412,916
NYSIIS	188,474
Levenshtein	97,730
Phonetic Vector (Distance 3.0)	14,550
Phonetic Vector (Distance 4.0)	73,962
Phonetic Vector (Distance 4.5)	**216,156**
Phonetic Vector (Distance 5.0)	685,516

Table 4.5: Phonetic vector number of matches comparison

Table 4.5 contains the match size comparison for Phonetic vectors distance. Distance 4.5 was chosen as the suitable size due to the total matches fitting well within the other metrics. Distance 4.0 was the other alternative, this was discarded due to its small size. Any metric with a number of matches below 100,000 results in an inadequate number of possible near-collision keys later in the process, a lower number of matches may imply better quality but a balance is required between computational cost and attack quality. This will be discussed in more detail in later chapters.

TODO: *Link the less than 100,000 point to experiment data*

Comparisons

Each participant is asked to rate the similarity of a match on a scale of 1 to 5. Figure 4.4 shows an example match and the connected scale.

Users will complete five of these comparisons per metric.

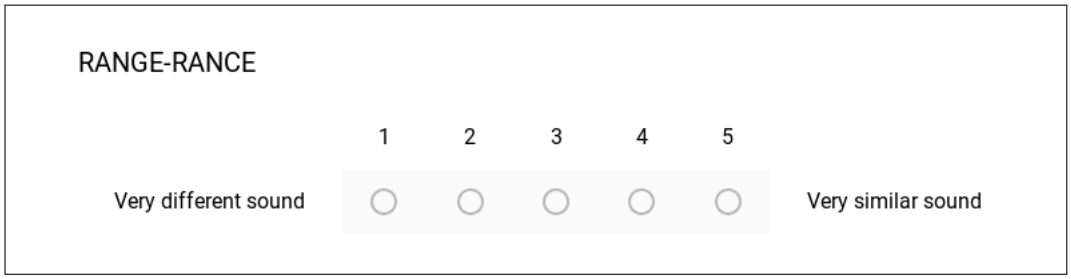

Figure 4.4: Example experiment question

The experiment randomizes the order of these comparisons per session alongside a complete refreshing of matches once per submission. This makes sure selections from the samples are fair and consistent as each user will have a different selection of matches.

Quality control

As the study was being outsourced to Amazon's Mechanical Turk the requirement to check the quality of results is important. Therefore, a couple of additions were provided to check a result's validity.

The first was the addition of five "Random matches" questions. These are matches consisting of two random words selected from the dictionary. Therefore, the expected average rating of these words should be close to 1. This allows a simple check to see if the user is providing valid results. If their average is too high, the results will be discarded.

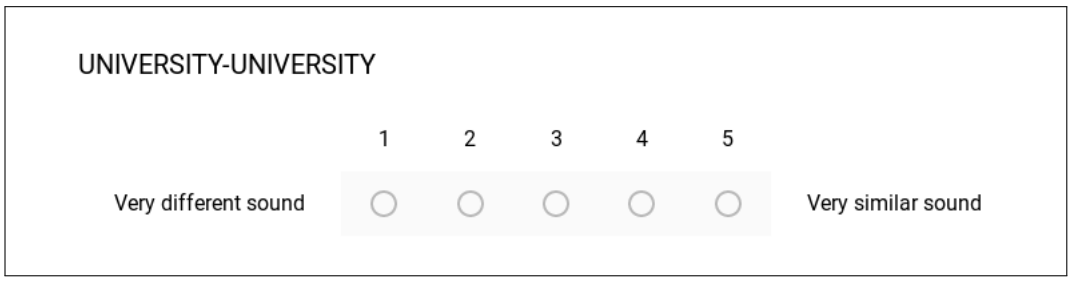

Figure 4.5: Exact experiment attention question

Alongside this, was the addition of two questions comparing exactly the same word. As both words are the same the result should always be a full 5/5 rating, any results without a full rating will be discarded. Figure 4.5 contains one example of the attention questions used to filter inaccurate results.

The final check to ensure the validity of results is a check for native English fluency. Having non-native English speakers complete the experiment has the possibility to introduce inconsistencies into the data and, therefore, needs to remain controlled. To achieve this a preliminary MTurk study will be run to ask users for their perceived English fluency on a scale of 1 (Basic Understanding) to 5 (Native). Workers with an answer of 5 will be provided with a Qualification[4] that tags them as defining themselves as native speakers. This is then added as a prerequisite when running the experiment. This will insure only fully native speakers are being assessed. The possibility of the worker falsely stating their level of fluency has also been considered. However, as the worker does not know the purpose for this initial study, thereby providing inaccurate results has no real valid motive for the worker.

Considerations

The first consideration is the way the words are compared. Due to the channel of authentication being phone based the comparison of words will be a mixture of auditory and textual. This is because a user needs to match a words sound to the one displayed on their device. In the case of this experiment the users are asked to compare the sound of two word that are visually displayed to them.

This visual aspect is the first point of contact between the participant and the question, followed by mental comparison of the phonetics of the word. Therefore, this initial visual contact has the potential to bias the results of the study. Levenshtein has the potential to be most affected by this due to the matches never being more than one character different, therefore, resulting in very similar looking words. The alternative method of running this experiment is to get users to compare audio samples of words, this will force the user to compare just the phonetics of the words and not be influenced by the visual aspects of the pair. This, however, adds a substantial cost onto the operation of the study. Due to the aforementioned lack of resources the chosen design was decided as most feasible as it is a balance between cost and accuracy. Further work, therefore, could improve by producing more conclusive results on the performance of the metrics by implementing the more accurate but costly alternative.

Another consideration is the demographics of people accessible on Mechanical Turk. A comprehensive and still active survey run by D Difallah *et al.* [37] shows that MTurk workers are younger and have a larger household income than that of the US population. This has to be taken into consideration when interpreting the results because it

[4] https://blog.mturk.com/tutorial-understanding-requirements-and-qualifications-99a26069fba2

will inherently introduce a bias. This is due to the assumptions that an increase in age provides a better understanding of more obscure words, and, therefore, a better representation of the metrics performance over the entire dictionary.

4.5.2 Trustword Attacks

The final experiment required to answer the Research Question 6 will simulate attacks on participants. The user will be presented with the same design as presented in the p≡p Android application (See Section 2.3.2).

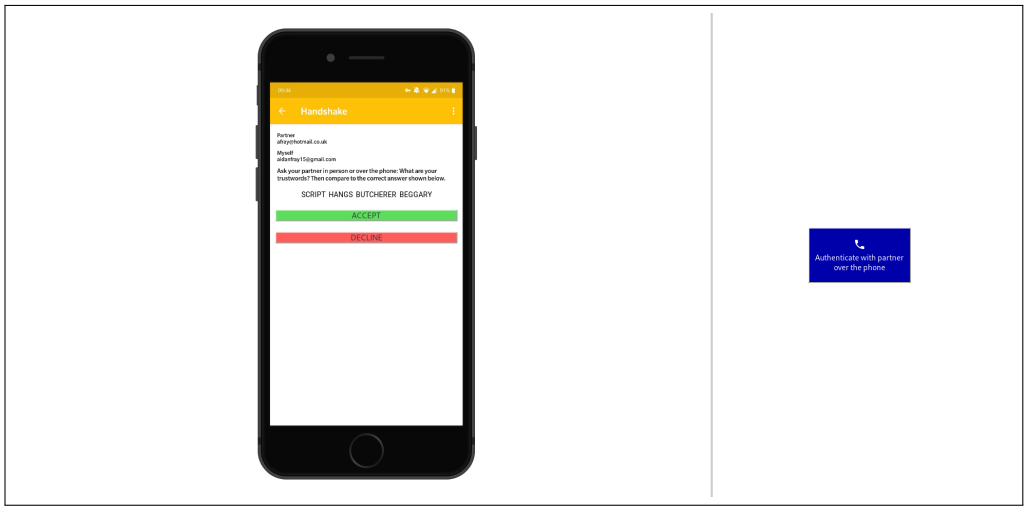

Figure 4.6: Experiment UI

Figure 4.6 shows the design of the experiments front end. As can be seen by comparing it to Figure 2.2 it has been designed to be a close as possible to the actual application. Users will click the blue button to simulate the authentication ceremony over the phone. A text-to-speech system will then read a set of words and the user should accept if they match and decline if they don't.

Design

Due to the scarcity of attacks in a real-world setting users will are often complacent about their occurrence. Therefore, in this experiment attacks only occur 30% of the time. This is a much higher value than in a real-world setting, but it is a balance between resources and realistic design. Alongside, this the first 5 trials of a new experiment will always

be benign. This is to ensure users are lulled into a level of complacency regarding the possibility for an attack.

Certain keys have higher levels of near-collision possibilities than others due to how certain words are deemed similar by the chosen metric. Therefore, this presents the possibility to have keys with a very low number of similar matches. The distribution of these keys will be explained further in Chapter 6. Therefore, if random sets of words were chosen with no restraints there are attacks presented to the user that in a real setting would be infeasible as they are close to a 2^{64} attack (4 Words * 16-bits). Therefore, the experiment will sample from a list of "vulnerable" keys. These are keys that have a number of combinations that allow an attack in a certain timeframe. Furthermore, certain levels of attacks are also simulated. Below are the list of possible attacks:

- **Zero static words** - All words in the set can vary.
- **One static word** - All but the first word can vary.
- **Two static words** - All but the first and last word can vary.

The start and ends were chosen as the highest priority words to keep static. This is due to research highlighting on the common habit of users to only check the start and end of a checksum. [38] shows this by measuring user eye movement and displaying it as a heat map.

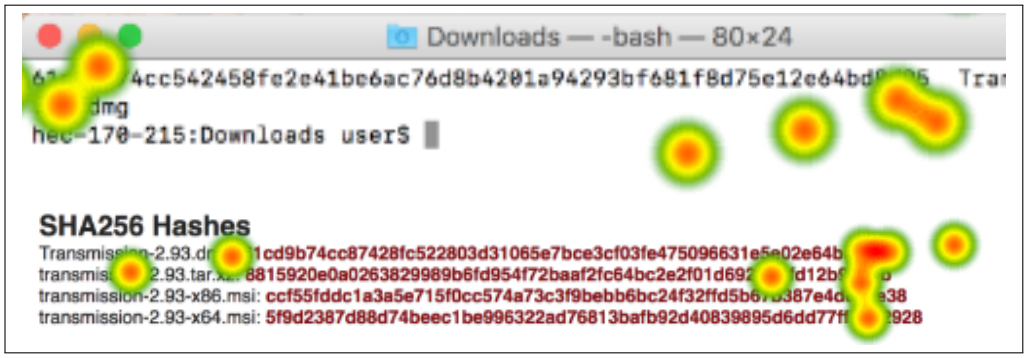

Figure 4.7: Failed verification of an incorrect checksum[38]

Figure 4.7 shows the results of a eye tracking experiment where the user failed to detect a mismatching digest. As it can be clearly seen the users never compared the center of the digest.

However, this may not be applicable in this context due to the user being linearly fed an audio stream that they cannot skip to the start or end. We believe, however, that attention will also play a part in defining the most important areas of comparison. The hypothesis is that the users attention will be lower in the middle sections of comparison where it peak at the start and end. All these aspects will require empirical

evidence to arrive at a conclusion. Therefore, this consideration will need to be made when assessing the results of the experiment.

These attacks are ascending in complexity. The timeframe for finding a match was decided as 7-days with attack strengths of 1 GPU/day, 10 GPU/days and 100 GPU/days on a mid range GPU (AMD RX 480 was the example in experiments). Due to the long life time of PGP keys, 7 days was chosen as a reasonable length of attack, attackers can just leave the tool and have it compute potential keys for the target without any need for user engagement. This is also a maximum time, these attacks also include keys that have a shorter average computation time.

GPU/days	No of combinations required	Attack Type
1	15250	Zero static
10	1525	One static
100	152	Two static

Table 4.6: Summary of attack requirements

Table 4.6 contains a summary of the different level of attacks and their respective restrictions. More combinations makes the attack search quicker as there are more possibilities. Any key that exceeds this criteria is deemed as vulnerable is one of the three attack contexts. A list of these vulnerable keys are sampled from when an attack is simulated. The percentage of vulnerable keys will be explored in Chapter 6.

Quality control

As the previous experiment, participants will be sourced from MTurk. Therefore, again, quality control is and aspect that requires consideration. As with the previous study, initially screened native speakers will be used. The same process will be performed to recruit suitable workers. Two metrics will be used to detect invalid results. Audio button clicks and overall time taken.

Audio buttons clicks is the number of times the blue "*Authenticate with partner over the phone*" button in Figure 4.6 is clicked. If there are rounds without button clicks, this is a sign of non-attentiveness. The design choice was made to keep this as a result filter, as preventing non-clicks would be a trivial task. This provides a way to detect and discard click-throughs[5].

Overall time taken is as the name suggest a recording of the time taken to complete the entire experiment. If users complete the experiment in

[5]Workers that aim to complete the task as fast as possible, with no regard for the quality of responses

an abnormally small amount of time it can be used as another detection for invalid responses. Anything below a certain threshold will be discarded.

Considerations

As with the previous experiment MTurk demographics will result in participants that are not reflective of the general population. Due to the attack context of this experiment is can be assumed that the youth of the participants will contribute to less successful attacks. Therefore, the results generated by this experiment can be loosely assumed as a best case scenario for the scheme. We predict a high attack success rate if assessed over the general population.

5 Implementation

This chapter will discuss implementation details of the project. Instead of explaining all the technical achievements in detail, the most interesting ones will be discussed alongside the respective challenges encountered.

5.1 GreenOnion

General design

This section will briefly discuss the overall technical implementation of GreenOnion alongside a deeper discussion into the unique aspect of the tool.

GreenOnion is written in C++. This languages was chosen due to its efficiency bonuses over other alternative languages. The tool begins by generating a 2048-bit RSA key through GPG[1]. This key is then used to create a hash of all but the final 512-bit block of the key. The final block of the key is left un-hashed due to the presence of the exponent (e). This exponent is incremented to create a new unique key. This is the aspect provides rapid generation of new keys due to the reduction in expensive large prime generation. This prevents issues in such as entropy starvation when generating a large number of valid RSA keys. This produces valid keys but with abnormally large exponents, this was deemed as suitable due to the short term use-case for these keys. 3 bytes are used to represent the exponent giving the potential to create $2^{24} - 1$ extra keys for a single expensive key generation.

All keys plus the intermediate hash are loaded on the GPU via an OpenCL kernel. This kernel is designed to hash the final block, obtain the final fingerprint and checking if the fingerprint is present in the provided list.

[1] GNU Privacy Guard

Bloom filter

Mass checking of fingerprints is the tools main abilities as it allows for millions of keys to be checked with a very small amount of overhead. This is achieved through the use of a bloom filter. A bloom filter is a probabilistic data structure that allows efficient checking for presence of an element in a set. It is effectively a very large array of booleans that state wether an element is present where the position in the array is decided by a pre-defined hashing algorithm. This is repeated a number of times with a number of hashing algorithms (k) to populate the array of length (m). This then means checking the presence of an element in the set will just means hashing the target and checking the indices returned. This data structure therefore has a complexity of $O(k)$ regardless of the number of elements in the set. This is the only data structure that provides this useful characteristic.

However, due to the random distribution of the hashing algorithms, there is the possibility for collisions and, thus, the possibility of false-positives. The data structure, however, does not produce any false-negatives. This, therefore, makes it fully suited to this use-case as any fingerprints tagged as "possibility" being present in the bloom filter can be followed up with a more expensive hash-table check to determine their actual presence. Therefore, if the levels of false-positives are controlled (by altering k and m) the tool can search through huge number of potential keys without any decrease in speed.

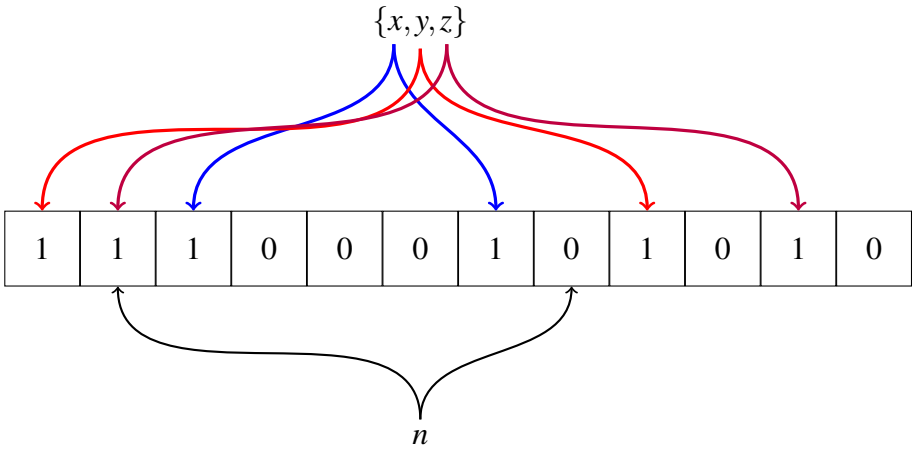

Figure 5.1: Bloom filter example

Figure 5.1 shows the operation of a simple bloom filter. As it can be seen the element n definitely does not exist in the set due to both array indices not being set. The performance boost when comparing large numbers of keys is substantial when compared to the similar tool Scallion (See Section 4.4.) The results from the comparison will be

quantified in Chapter 6.

5.2 First Experiment

In order to assess and collect data from users for the first experiment (See Section 4.5.1) Google Forms[2] was used to host and design the questionnaire. It provided functionality to easily present questions and record their responses. However, as the design requirement of renewing sample from the matches of each metric was not part of the main applications functionality a solution was required. This was solved by Google's App Script[3] service that allows a user to programmatically interact with the Form. This allowed the re-sampling of matches each time a user submitted a response.

Figure 5.2 contains an interesting snippet from the script used in the experiment. Each time a user submitted their response the `updateForm()` endpoint was called, that subsequently updated the questions for the next user. This aimed to keep the samplings fair as they are equality likely to appear when attacking the system alongside an ample exploration of the overall match quality.

5.3 MainExperiment

In order to asses users in a similar scenario to ones experienced when utilizing p≡p in the real-world a custom application was required. The front-end is developed in Javascript with a Python-Flask backend that acts as the webpages functionality. A number of end points are exposed like `get_audio` or `get_words` that are requested and displayed on the front-end.

One of the main problems encountered was the handling of each user's session. Initially during testing it was noted that changes in one endpoint if timed correctly would overwrite the progress of another, this is commonly know as a "Lost update". This was due to the user session being stored in the `session` cookie. When sending off a request the session cookie is included in the header, the code will then utilise the data within the cookie to make decision and alter state, the updated state is then returned as a cookie in the response. If, however, another request is executed before the first has completed, the second request is

[2] https://www.google.com/forms
[3] https://developers.google.com/apps-script/

```
var ALGOS = ["Soundex", "Metaphone", ...];
var ALGO_SIZES = [763777, 412916, ...];

function updateForm(ss, form) {

  // Obtains a list of the items present on the form
  formItems = form.getItems();

  // Updates the new values
  var algo_index = 0;
  var scale_index = 0;
  for (var i = 0; i < formItems.length; i++)
  {
    if (formItems[i].getType() == "SCALE")
    {
      var rnd_index = Math.floor(Math.random() *
      ALGO_SIZES[algo_index]) + 1;

      // Obtains the Google Sheets object
      var sheetname = ALGOS[algo_index];
      var sheet = ss.getSheetByName(sheetname);
      var range = sheet.getRange(rnd_index, 1)
      var values = range.getValues();

      // Re-samples from the connected Google Sheets
      _updateScaleTitle(formItems[i], values[0]);

      scale_index++;

      // Moves on to the next metric when a group size is
      // been completed
      if (scale_index != 0)
      {
        if (scale_index % QUESTION_PER_ALGO == 0) {
           algo_index++;
        }
      }
    }
  }
}
```

Figure 5.2: Section of code from the First Experiment's Google App Script

using the context of the first request and, thus, any alterations made by the first will be completely lost.

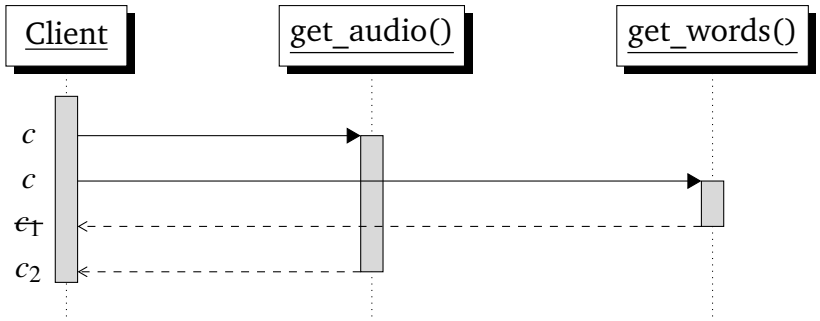

Figure 5.3: Sequence diagram for the 'Lost Update' problem

Figure 5.3 shows this problem visually. As it can be seen the response cookie c_2 is based of the old data of c and, therefore, when returned will remove the progress of c_1.

To keep the complexity of the backend low it was opted to solve this problem by restricting the user via the UI, when a button was clicked all other operations were disabled until a response was received. This stopped the possibility of sending off another request while another is still processing.

I order to collate data for results later in the process state had to be saved. One potential solution would be to format the data and save it as a .csv file. However, this is a static way of data persistence and would require alteration every time the data schema was modified. The chosen solution utilized a python object serialization library known as 'Pickle'[4]. This library saves the an instance of a class as a file. This file then can be used to load the instance of the class back into the program. This allows changes to the object to filter down through the program and keeps the design of the application fluid. For example, after initial feedback gathered about the application it was decided that a inclusion of 'Round times' would be a useful addition. Edits were made to the `experiment` class and the changes propagated through the application, no changes were needed on the save functionality. To distinguish between old and new save formats a version number was used.

[4]https://docs.python.org/2/library/pickle.html

6 Experiments

6.1 Scallion vs GreenOnion

This section will compare Scallion and the newly designed GreenOnion ability to search for a large number of potential keys.

Table 6.2 shows the speed decline as the number of keys increase. Any number of keys over 45 resulted in a crash to the program.

TODO: *Add screenshot of the exception*

Scallion	
OS	Windows 10 (May 2019)
Version	Scallion-2.0[1]
GreenOnion	
OS	Linux Mint (Version: TODO)
Hardware	
GPU	GTX 750 Ti
CPU	AMD FX-6300

Table 6.1: Testing environment

Number of keys	**Speed**
1	1001.3 MH/s
5	784.3 MH/s
10	707.9 MH/s
15	628.2 MH/s
20	526.5 MH/s
25	503.8 MH/s
30	473.1 MH/s
35	432.6 MH/s
40	404.9 MH/s
45+	*N/A*

Table 6.2: Scallion's speed decline

TODO: *Plot scallion speed decline*

TODO: *Show the maximum number of keys the tool can achive*

6.2 Experiment 1 - Metric performance

As discussed in Section 4.5.1 the goal of the experiment was the cull the number of metrics to be assessed in the following experiment. This section will discuss the demographics of participants alongside the subsequent results.

Gender	Male:	46.2%
	Female:	53.8%
Age:	18-24:	10.6%
	25-29:	20.2%
	30-39:	30.8%
	40-49:	22.1%
	50-59:	11.5%
	60-69:	3.8%
	70-79:	1.0%
Highest Education:	GCSE:	15.4%
	A-Level/O-Level:	18.3%
	Bachelor's degree:	51.0%
	Master's degree:	13.5%
	PhD:	1.9%

Table 6.3: Participant demographics

Metric	**Average Rating**	σ
Leven	3.66	1.15
NYSIIS	2.92	1.31
Metaphone	2.56	1.32
Phonetic Vec	2.50	1.35
Soundex	2.08	1.12
Random	1.16	0.46

Table 6.4: Average metric performance

Table 6.3 contains the demographical breakdown of the participants assessed. As it can be seen over 60% of participants can be considered highly educated (Bachelor's and up). This is not fully reflective of the general population and therefore, has to be considered when interpreting the results. All participants were sourced from the US, this again requires consideration due to the large range of dialects present that may bias the results. Further work could investigate the affect of location and dialect on similar results.

Overall, 104 participants were assessed in this study. Five results were discarded from the set due to either failing the attention questions (See

in Section 4.5.1) or having having too low of a fluency rating. This was a necessary process to improve the health of the results.

Figure 6.4 shows the average results for the metrics. It can be seen that Levenshtein came out substantially above the rest. The breakdown of the ratings in Figure 6.6 also shows Levenshtein's dominance. Levenshtein has a much larger proportion of 4 and 5 ratings than the alternatives alongside a very low level of low ratings. This performance may, however, be due to the visual way the comparisons are being performed. (Discussed in detail in Section 4.5.1).

Due to the averages of Metaphone and Phonetic Vectors being so close standard deviation was used as the final decider. As it can be seen Megaphone has a slightly lower σ value of that of Phonetic vectors, thus, contributing to the decision to select Metaphone to be carried over to the next experiment.

Considerations

As discussed in Section 4.5.1 compromises were made in order to fullfil all requirement with a minimized cost. Levenshtein's abnormal performance further backs up concerns around the visual aspect biasing the performance of the metric. This will have to be considered when interpreting the results.

Even with the discussed issues, the aim of the experiment was to promptly reduce the number of metrics for use later in the project. This experiment, therefore, has achieved that goal of providing three metrics for the subsequent experiment while balancing between accuracy and expenditure. The results are also an improvement over informal ad-hoc methods. Further work could aim to reproduce this study with the proposed audio based design.

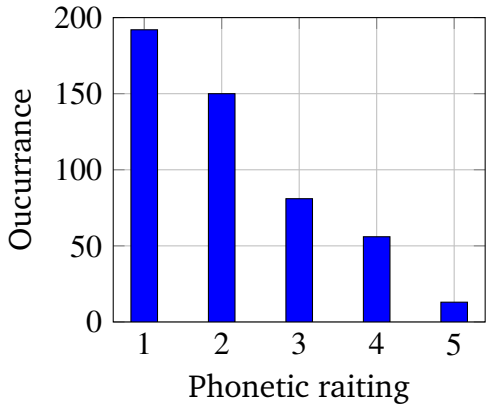

Figure 6.1: Soundex

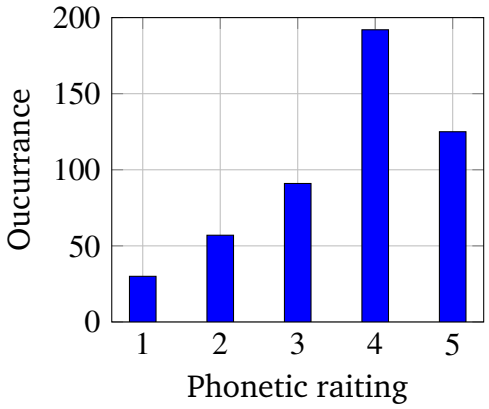

Figure 6.2: Levenshtein

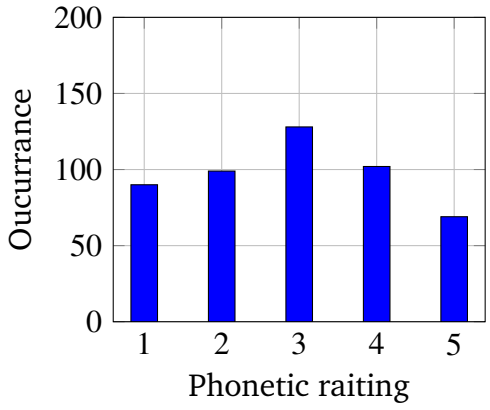

Figure 6.3: NYSIIS

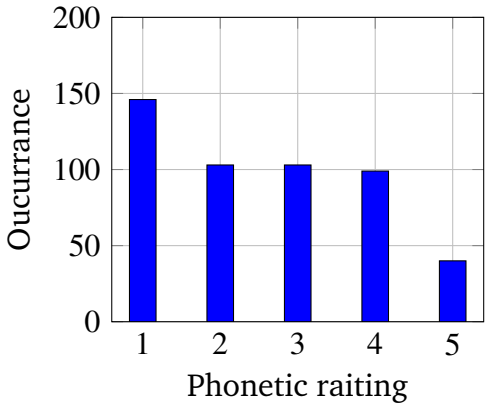

Figure 6.4: Metaphone

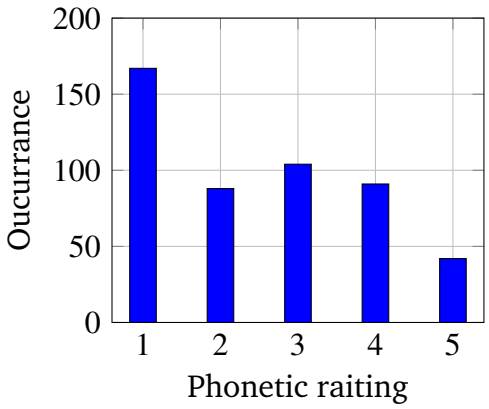

Figure 6.5: Phonetic vector

Figure 6.6: Individual breakdown of results for each metric

6.3 Experiment 2 - Trustword attacks

Section 4.5.2 explains the design of the experiment. This section will present and discuss the results of the experiment alongside a comparison to relevant literature.

Demographics

Overall, 435 paid participants recruited via Amazon's MTurk were assessed in this experiment. We excluded 66 results; 7 due to being non-native speakers and 59 were discarded for failing the attention metrics (Discussed in Section 4.5.2).

Gender	Male:	50.41%
	Female:	49.59%
Age:	18-24:	12.74%
	25-29:	18.16%
	30-39:	37.4%
	40-49:	17.89%
	50-59:	8.67%
	60-69:	4.34%
	70-79:	0.81%
Highest Education:	GCSE:	13.82%
	A-Level/O-Level:	24.12%
	Bachelor's degree:	51.49%
	Master's degree:	8.4%
	PhD:	2.17%

Table 6.5: Participant demographics

The reduced set of 369 participants had an average age of 36.63 ($\sigma = 11.35$) and consisted of an almost equal split of Male (50.41%) to Female (49.59%). Around 62% of participants had completed a single stage of university (Bachelor and up). This makes this set of participants more educated that the general population. All participants were also sourced from the USA and have rated themselves as fully native English speakers.

As user-agent strings were collected for each participant collations of OS and internet browser versions was made possible. Tables 6.6 and 6.7 contain the browser and OS breakdowns respectively. Chrome and Windows 10 seem to dominate the proportion of their respective domains alongside the surprisingly large proportion of workers utilizing older versions of Windows such as 7 or 8 (15.81%).

Browser	Percentage
Chrome	85.09%
Firefox	12.20%
Safari	1.90%
Other	0.81%

Table 6.6: Internet Browser breakdown

Operating System	Percentage
Windows (All)	74.26%
Windows 7	12.74%
Windows 8	4.07%
Windows 10	57.45%
MacOS X	13.82%
ChromeOS	3.79%
Android	4.34%
iPhone	1.08%
iPad	0.54%
Linux	2.17%

Table 6.7: Operating System breakdown

Results

Table 6.8 contains the break down of results for the experiment. It can be seen that the best metric out of the set was Levenshtein with an overall success of 19.80%. The best performer when regarding attack strength, as expected is the 2 static words. Levenshtein's 2 static attack performed the best overall with a success rate of 32.05%. The worst performer was Metaphone with an average of 16.88% over its three levels of attacks. When comparing the performance of the metrics to the previous experiment the ordering remains the same with Levenshtein, NYSIIS and Metaphone all ranking in the exactly same order.

Considerations

As previously mentioned, the set of recruited participants are more highly educated than the general population. This combined with the low average age contributes to an inevitable bias being introduced into the data. An assumption can be made that higher education leads to more skillful interaction with computer based systems, this, therefore, with the previous assumption in mind means the results of this experiment being a 'best case' scenario. The attacks, therefore, would be expected to be more successful on less technical users. This, however, requires further research and possible empirical data to determine the

Metric	Successful Attacks	Total Attacks	Success Rate
Levenshtein	218	1101	19.80%
0 static	34	358	9.50%
1 static	59	353	16.71%
2 static	125	390	32.05%
Metaphone	181	1072	16.88%
0 static	38	345	11.01%
1 static	57	375	15.20%
2 static	86	352	24.43%
NYSIIS	209	1114	18.76%
0 static	36	385	9.35%
1 static	72	375	19.20%
2 static	101	354	28.53%
Overall	608	3287	18.50%

Table 6.8: Success rates for simulated attacks

validity of the assumption.

During this study the main task was to perform the authentication ceremony a large number of times. In the real-world the ceremony is a secondary task due to it not strictly being required. This alongside the potential for users to downplay the possibility of attack leads to the conclusion that the success rates could be much higher in a real-world setting. The assumption is that these results display the success rate for extremely attentive user where they will make sure to double check with their authentication partner multiple times. This is opposed to the assumption that the majority of users will aim to complete the task as quickly as possible.

Comparison to alternative literature

This section will aim to compare the results of the experiment to similar literature.

Work by Kainda *et al.*[7] (previous discussed in Chapter 2) is the most comparable to this experiment. They were the only study to use exactly 4 words, but differed on the size of dictionary (1024 words) and how a near-match was calculated. Near-matches were a difference in a single word but without the consideration for similarity, this is the unique aspect considered with this experiment.

However, with those aspect in mind attacks on words encoding received an overall success rating of 3.3%, considerably lower than that

of this study. In the worst case our simulated attacks achieved almost 3 times as many successes compared to this study.

Other relevant work by Dechand *el al.*[5] also discussed in Chapter 2 assess the success rate of attacks on words. This is less similar to the experiment as 14 words per attack were assessed with each participant, where the comparison was performed visually. However, the success for the PGP Word list was 8.78%. As 10/14 words were kept static more comparable to comparable the results to the highest attack strength "2 static". Therefore, there is another 3 fold improvement in the success rates of our simulated attacks.

The final relevant paper to compare is that of Tan *et al.*. This paper had the highest number of words assessed with 16 overall. With it assessing the same wordlist and attack strength as the work by Dechand *et al.*. The attack success was 14% overall, this again compared to the highest attack assessed in the experiment results in a sizeable difference.

In conclusion, all similar literature had much lower attack success rates that the results presented in this experiment. With all papers using similarly designed words lists it highly suggests that the deficiencies in Trustwords are the cause for the substantial increase in attack success. Alongside this, the consideration for similarity when calculating near-collisions could have also resulted in the higher attack success rates. This aspect is also a unique consideration compared to available literature.

6.4 Average number of near-collision keys for each metric

In order to predict the effectiveness of the attack over a wide variety of keys the number of average near-matches requires computation. Table 6.9 shows the average keys for each metric. The "Compute Time" is the computation time required for a single mid range (2000MH/s) GPU.

As it can be seen metrics like Levenshtein have a low overall average and would require much more attacker resources to become feasible. On the other hand, Metaphone has attacks that could be computed on average in under 2 days. The success rate for this attack type in the second experiment was 11.01% on average. This means on comity hardware commonly found in the home, an attacker could compute a key in less than two days that could succeed more than 10% of the time. This is substantial and highlights on weaknesses in the Trustword system.

Metric		Mean matches	σ	Compute Time
NYSIIS				
	0 static	1963.07	24769.48	27.19 days
	1 static	448.37	4316.57	119.14 days
	2 static	98.07	550.44	1.49 years
Metaphone				
	0 static	27138.78	529229.35	1.97 days
	1 static	2822.74	23932.39	18.91 days
	2 static	339.84	1730.97	157.45 days
Levenshtein				
	0 static	293.92	2158.24	182.17 days
	1 static	102.75	719.28	1.43 years
	2 static	35.53	120.71	4.18 years

Table 6.9: Each metric's average near-collision keys

6.5 Distribution of vulnerable keys

As discussed in the design of the second experiment (See Section 6.3) the attacks sampled from a list of 'vulnerable keys'. These vulnerable keys were created using real keys extracted from PGP key servers. These keys were then randomly sampled and used to create the combined key as discussed in Section 2.3.2. 100,000 random key pair were created and used as the sample. Any vulnerable keys in this set were recorded and used in the experiment.

TODO: *Reference David's work*

Attack Type	Metric	Vulnerable %
2 static (152)		
	Levenshtein	4.295%
	Metaphone	24.095%
	NYSIIS	10.441%
1 static (1525)		
	Levenshtein	0.827%
	Metaphone	15.569%
	NYSIIS	4.341%
0 static (15250)		
	Levenshtein	0.157%
	Metaphone	10.202%
	NYSIIS	1.824%

Table 6.10: Number of vulnerable keys per metric

Table 6.10 contains the breakdown of the number of vulnerable keys.

The takeaways from this table is the extremely low values for Levenshtein in the higher attacks. This is due to the low number of overall matches produced by Levenshtein. This means Levenshtein will not be effective in the vast majority of cases. However, the standout metric is Metaphone with a good proportion of vulnerable keys, it was the worst performer when assessed against real users but only by as little as 2%. If you take into consideration the number of vulnerable keys detected in the set alongside the average amount near-collision matches in Table 6.9 Metaphone appears to be the most usable metric overall.

6.6 Generation of keys

To demonstrate that the the generation of the near collision keys is not only theoretical actual generation of keys was performed using GreenOnion. A random metric was selected from the three carried forward from the first experiment. Alongside this a random uncontrolled key was generated as the simulated attack target (See Appendix A for the armor public key). Then for each attack level (0 static, 1 static, 2 static) a key was computed. NYSIIS was chosen as the selected metric. To expedite the process an initial search was performed for controlled keys that had the highest number of near-collision keys, this was due to the demonstrative purpose of this experiment. Producing near-matching keys from key pairs with a very small number of potential matches, adds very little to this demonstration.

Tables 6.11, 6.12 and 6.13 contain the parameters and results on the computation. Due to varying availability of resources each computation has a varying hashing speed. This, however, has been normalized by the introduction of the "GPU Day" metric. This is simply the computation time required to reach the same conclusion on a single 2000 MH/s GPU.

As it can be seen most notable from Table 6.13 the generation of the actual Trustwords differs in only 3 characters making it arguably very similar. When linking this generated key to results of the second experiment (See Section 6.3) computation of 2.53 days on a single mid-range GPU is able to achieve around a 28% success rate when assessed against actual users. Even with the least restricted attack type presented in Table 6.11 the computation of half a day can provide almost a 10% success rate when attacking users. This means any user with a single commodity GPU could compute enough keys in less than a week to have a very high probability of fooling an actual user.

Potential near-collision keys	142,296
Hashing speed	4000 MH/s
Estimated computation time	9 hours
Actual computation	6.41 hours
GPU Days	0.52
Original Trustwords	BELL GRIND ALGERIA ANNULI
Actual Trustwords	BOIL GRAND ALGER ANNUL

Table 6.11: NYSIIS - 0 static

Potential near-collision keys	133,200
Hashing speed	8000 MH/s
Estimated computation time	4.2 hours
Actual computation	5.12 hours
GPU Days	0.853
Original Trustwords	ASSUMING SONOMA DENS KEENER
Actual Trustwords	ASSUMING SUMMON DENNI CONNOR

Table 6.12: NYSIIS - 1 static

Potential near-collision keys	16,464
Hashing speed	6000 MH/s
Estimated computation time	2.16 days
Actual computation	23.44 hours
GPU Days	2.93
Original Trustwords	VOCATION BORE TANN ANTE
Actual Trustwords	VOCATION BARE TONE ANTE

Table 6.13: NYSIIS - 2 static

7 Evaluation

8 Conclusion

A Randomly generated uncontrolled key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBF0vHaYBCACxOy7RacHIprgaUo66vGmIB2VIaSwToliuaEBxOAloD+wTB/T9
TOKgAMLWMgK2z3/tOZj2l5k9ATUr0nZ9iBgew5Ih3ykOp+OkE9dh4NTD3EJz5UWo
M1r6scPhby92zMBqKi0iPFlPcvk+eg+SPusxIp+Vn16ALoB2pwGQvG+qoFEJfdVP
nwSAWJo1mcd+TCxgIMqe11FXDzxpd4BEpMXsHOB8NH/TUiI57sxqT8A2HnnyWo2i
5vs93VAP8rTTIRfGelW2c3oqhV9XhXbvVpJPmvkTxM/SshV1L5HhYr1TUAkAQw6w
RUZqooysgZEDqQw58lFQ7C9plCHOpXIEhbE7ABEBAAG0HEpvaG4gSm9obnNvbiA8
am9obkB3b3JrLmNvbT6JAVQEEwEIAD4WIQRQZ0ukmi5QUEz7KB3FwHKSVKqpwgUC
XS8dpgIbAwUJA8JnAAULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRDFwHKSVKqp
wg4YB/9knveazSM4yN3EJY7HRZe5PVA5r0GGDlEDMte+surPKLsV/T39SNl8qTH+
DXnOyC4rt1kykTJP+ULcKqGkVRnH2XYZP/W6E9wwBPjoKHaWyn8MP0ix5kgkq6Ld
ZWEQ2pUwCsAhWv8K/ybjQ+Li484JKbrz/GRALp8qU3hZgCo/Fw3sD2KcTPU1v8qU
av9G2N5su10W7G0F1ozszFit+r2iinoZtGSddBPG+pIUEtZoBn4V2FRRSO1UY/TU
jGf2BLsk+ZkNJdx5x6IEU9CQ+ivPuKtupVu9ySKqE2Qfmn5lh9iAjYDN3XxUQfCf
braGoC9JLnL5QmT1JaQXfxkIHilyuQENBF0vHaYBCADDp/u8D/TG5d5HmabmgbBF
3R2d3J00vt7VQQ5sIRMLyP0cx6qy2bTIy9mgfU3/WcrhRp728gBo+NnUfxyiTb5R
u7AbntlkerF3dh0JLm5eIossPW37QXn3Ce4Tv4ixGRLbsGKTu6h0lP41FM5VcqV9
99NE1iml3WyngnsFdKeUYqPW4QY+ydyjpSyblbQTMptIKG2llCmsd9k+hb8nnGx/d
GvyGKaZ270iVGY2im/9VP2VwzqXlFxxx6Dux4egJJMMAy5xkzBKf+Szmm1pMJkt2
4sBrwmLnuUObgsAkq8K+MqEpWeFi6aWsh7RnY8T0SEb/3QUwG12V77ePhrarBwmf
ABEBAAGJATwEGAEIACYWIQRQZ0ukmi5QUEz7KB3FwHKSVKqpwgUCXS8dpgIbDAUJ
A8JnAAAKCRDFwHKSVKqpwoW3CACeKIcQFuVv2vpUHq/4ATH0atPYqR1NbJTegE+v
XZyrNHtmRF4V+kdx/1KX9jsIY2TjyxbhTsrtMFkMZMfAVjXVoS4y8OkwX5IJWgHC
0JQIHQrvaIGfI1yuVrNvWYyPXFxNtbntrgTd8JKYGKYaTqXNZ2r1ozbZElp7fQtR
rWNUGJYYo+aHHLM+Aot5k+Z3YqHifuPSnCEYt/GxnMhYq1IL3LSZ96hsgwqeFoLP
+v4RsvY52Yb5tiOzxZwikALOJdM2aAumxq8R1nYHiswWxotw2WkKFlB7gfLpbMWs
Y+o4J7QKSqpVTrUYBTPF1F4kBWOMY10kpeCDbZ1YwJOZJf8P
=aHBu
-----END PGP PUBLIC KEY BLOCK-----
```

B Computed Attack Keys

B.1 NYSIIS - 0 Static

B.1.1 Controlled Key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mI0EXS8lIAEEAMWEJ+NGB/ukiUijzdT+DvzVwouWV/j4Rw78X9YdgJP0ExT4ncHG
TJbOKjEMEI9bb/oYMLF/434ufScYNci+rL/3RVY8CcjEv5l0UGX+udZ78c7Z/vf1
zFr2MQAglr9M8599JwUY25LiNKzDYc87B16hqRV8hRaXHfAFFHBxRsgBABEBAAG0
HUF1dG9nZW5lcmF0ZWQgS2V5IDx1c2VyQENWRUQ+iNQEEwEKAD4WIQSlBU278MbT
iVM5h6GNohojnp5YQgUCXS8lIAIbLwUJAeJfkAULCQgHAgYVCgkICwIEFgIDAQIe
AQIXgAAKCRCNohojnp5YQvxSA/0bjL+4me3hLjrM+8IckKABCRdRYYRG7svbln9q
ULUngsXMpbIMoHFg1ePpT2lniiHtQhDC88ZW1J5yeVZN1cfJhRsByyKlkzG3Dcy7
5wKonbVz1LSYpIDq/DNNP0eINzuVNc1rxqRXPGAh6NHnmZN5MqxLNyu0DSX9oJHm
KbbtLQ==
=FQCu
-----END PGP PUBLIC KEY BLOCK-----
```

B.1.2 Computed key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQEOBF0zhx4BCAD0iC55GShgArPzKZdn6we2mrkxS8o8fI7HdWXnLoRAta1Q1j9z
U+mYU9tGYb0yjqIARjjUK2qnq0CsGZJpB65AhbrOsDLv6dmI68tuiT4xjmzcM6+6
6M/EY5E5hYpZeurDJ8rSYrfu/v9P0bV78VnM17uUl4PTyEBmbJsE9kszq4MXd5yz
G+ibBPBvnBWSlRQzyCFMgTL8gQH36XJUj7jgxTKoJWzli/QtNcMOjwSQtEZxFWVK
+qsJjYUClpg9Kkow1LqJmEc1Tj98EipypDtdwNQhn4wHQVgu7SliaNNmrR8MpP9o
gL9hhfafx5FsE7v4zHhzu+ubR+c7KnUlwJyfABkBhI5atQAeSm9obiBEb2UgPGNo
YW5nZV9tZUBlbWFpbC5jb20+
=
-----END PGP PUBLIC KEY BLOCK-----
```

B.2 NYSIIS - 1 Static

B.2.1 Controlled Key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mI0EXS8o2AEEAKQmxot2Z5XFrNjg4zXBfvLsTzoO3vZeEI9l8+U2X8aq5KVQJpPU
BRSfqQuMZvA1zh3vH3w4I/vR4nXCNw/hPCi0B/DckUe6IJPZJfMiQfT1Xt16ZEms
W53CBiNqnt5xruo4wKwoKGuHELtPcD4K/iBIFt6n3UTi+R8xU4WMc+PjABEBAAG0
HUF1dG9nZW5lcmF0ZWQgS2V5IDx1c2VyQENWRUQ+iNQEEwEKAD4WIQTFXc8VXOqe
13ib81dcbeH3Z8A13wUCXS8o2AIbLwUJAeJb2AULCQgHAgYVCgkICwIEFgIDAQIe
AQIXgAAKCRBcbeH3Z8A13wItA/kBbfc6kxtLumDQiGpcIzgWmM9UhCn0V8ny9iB2
gt4V8bImpC3qicj6+KywsEsvFIgsJO0UcNeoREKanpXxPf6T7tvBUTLQjPJFedsn
J57996JiZ9COQ4lEg2AvfAKy1alKrMs0Xy4hWrHY9cJWs1a7KpbrsioHd4vDRgMS
2yRn/w==
=krrv
-----END PGP PUBLIC KEY BLOCK-----
```

B.2.2 Computed key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQEOBF0yA4QBCAC78Lk2k9BdUEufpqll4D60iAuYVrxVs3lpjwP2hpztBFFw8vY4
26uW1C7MPo9l4Oteks5626hwAAgI1/YHFwWpBDdTWYEnVGrjv5v0Juz5IZc7kiwp
Ui2XWJ7I3VpE5nljFyzaQMGaS5Cvmtco5cjin1Tx486jieGBt0loQQdtiJ3gEAFT
XlUvwIQSdYWY2F8xWKg5aJoDFEZN74KfwQqB7hx/53VHyzOJGygnvvpMcJaX3c8E
V3t7n90WvXJWND7bn6HIJrnFKyRJsKHneL2n0wGd/N0NLCuLTvNK+uQ7sXGNZox1
Qa4a1guiSdaq9NO5mX+tOYox3/onzpshJgVDABkBvckntQAeSm9obiBEb2UgPGNo
YW5nZV9tZUBlbWFpbC5jb20+
=
-----END PGP PUBLIC KEY BLOCK-----
```

B.3 NYSIIS - 2 Static

B.3.1 Controlled Key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mI0EXS8s+gEEANkph53UuQkNVFhYimrzH0bXBp/Y3InV9p5MP09TZxtTOf44JBPp
MCFMvgFJpVK3D5PQ2h7fW1lNb41THkiGC7Jk+F7gd+beqH5kaaKmC+RNAF0aZ5av
2al5JvtHT0Z8/1r8bxfp33WLmQXK0/9EwL4HvERRGh7wZ5WemvCrvZPXABEBAAG0
HUF1dG9nZW5lcmF0ZWQgS2V5IDx1c2VyQENWRUQ+iNQEEwEKAD4WIQSjUV9vxlLP
LPE78Z2TulybLqhYHAUCXS8s+gIbLwUJAeJXtgULCQgHAgYVCgkICwIEFgIDAQIe
AQIXgAAKCRCTulybLqhYHONrBAC8uh/K2Adq/o/Qe/EVM5JpT6ETE1WlAj2iUMWB
MAmdjgiQlRyBCTd4Lt9xFsbnvQG9pj4GVHosXjVpuA/1NxyS+DE4EkBDzXA3NuR8
0djhyD27VSsn2q1j8bB1wxO7sLbjo30SIBG2wUaft5DVJiBGG69kh4pt+aKKOcjb
LLPldw==
=dCl/
-----END PGP PUBLIC KEY BLOCK-----
```

B.3.2 Computed key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQEOBF0wxGEBCACsqQN92G3IizweQHw42OhUM5ZySKTvzPqVhAeCb2ETSBDZFbGN
lHYPa6dytXwLbU6kdlt8kgxbk75iPJsCSPLAUzKvlVd2rkTu5xtKPFqDTRvbjOjf
Ik8zJimXJGZNYVrMUqkfPR2FtVC4VUJXtML8voj/1lcLxTWRcYjuNQuSRNhJQtUP
ViZMTj97IO894lVg+ciZXEZkzHzlriqeLqwTupQxvzqjmObmeH1NCPSfGcTSNIBu
cNG/DrXTXsVpZPUilIRsXTtc2FM9iY49Zjv01KdzsJlV97JVyghYFFCEe1xq7Twp
U9TOzypXD4NdTKivFZUNT4jdG2HoJE4IcZ3NABkBh0qYtB5Kb2huIERvZSA8Y2hh
bmdlX21lQGVtYWlsLmNvbT4=
=0nU6
-----END PGP PUBLIC KEY BLOCK-----
```

Bibliography

[1] G. A. Miller, 'The magical number seven, plus or minus two: Some limits on our capacity for processing information.', *Psychological review*, vol. 63, no. 2, p. 81, 1956.

[2] J. Callas, L. Donnerhacke, H. Finney and R. Thayer, 'Openpgp message format', RFC 2440, November, Tech. Rep., 1998.

[3] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau and A. Narayanan, 'An empirical study of namecoin and lessons for decentralized namespace design.', in *WEIS*, Citeseer, 2015.

[4] 'Whatsapp encryption overview - technical white paper', Dec. 2017.

[5] S. Dechand, D. Schürmann, K. Busse, Y. Acar, S. Fahl and M. Smith, 'An empirical study of textual key-fingerprint representations', in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 193–208.

[6] J. Tan, L. Bauer, J. Bonneau, L. F. Cranor, J. Thomas and B. Ur, 'Can unicorns help users compare crypto key fingerprints?', in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ACM, 2017, pp. 3787–3798.

[7] R. Kainda, I. Flechais and A. Roscoe, 'Usability and security of out-of-band channels in secure device pairing protocols', in *Proceedings of the 5th Symposium on Usable Privacy and Security*, ACM, 2009, p. 11.

[8] H.-C. Hsiao, Y.-H. Lin, A. Studer, C. Studer, K.-H. Wang, H. Kikuchi, A. Perrig, H.-M. Sun and B.-Y. Yang, 'A study of user-friendly hash comparison schemes', in *2009 Annual Computer Security Applications Conference*, IEEE, 2009, pp. 105–114.

[9] A. Perrig and D. Song, 'Hash visualization: A new technique to improve real-world security', in *International Workshop on Cryptographic Techniques and E-Commerce*, 1999, pp. 131–138.

[10] C. Ellison and S. Dohrmann, 'Public-key support for group collaboration', *ACM Transactions on Information and System Security (TISSEC)*, vol. 6, no. 4, pp. 547–565, 2003.

[11] Y.-H. Lin, A. Studer, Y.-H. Chen, H.-C. Hsiao, L.-H. Kuo, J. M. McCune, K.-H. Wang, M. Krohn, A. Perrig, B.-Y. Yang *et al.*, 'Spate: Small-group pki-less authenticated trust establishment', *IEEE Transactions on Mobile Computing*, vol. 9, no. 12, pp. 1666–1681, 2010.

[12] M. Shirvanian, N. Saxena and J. J. George, 'On the pitfalls of end-to-end encrypted communications: A study of remote key-fingerprint verification', in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACM, 2017, pp. 499–511.

[13] E. Uzun, K. Karvonen and N. Asokan, 'Usability analysis of secure pairing methods', in *International Conference on Financial Cryptography and Data Security*, Springer, 2007, pp. 307–324.

[14] M. M. Olembo, T. Kilian, S. Stockhardt, A. Hülsing and M. Volkamer, 'Developing and testing a visual hash scheme.', in *EISMC*, 2013, pp. 91–100.

[15] D. Loss, T. Limmer and A. von Gernler, *The drunken bishop: An analysis of the openssh fingerprint visualization algorithm*, 2009.

[16] A. Karole and N. Saxena, 'Improving the robustness of wireless device pairing using hyphen-delimited numeric comparison', in *2009 International Conference on Network-Based Information Systems*, IEEE, 2009, pp. 273–278.

[17] P. Juola, 'Whole-word phonetic distances and the pgpfone alphabet', in *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, IEEE, vol. 1, 1996, pp. 98–101.

[18] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik and E. Uzun, 'Loud and clear: Human-verifiable authentication based on audio', in *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, IEEE, 2006, pp. 10–10.

[19] N. Haller, 'The s/key one-time password system', 1995.

[20] K. Rieck, 'Fuzzy fingerprints attacking vulnerabilities in the human brain', *Online publication at http://freeworld. thc. org/papers/ffp. pdf*, 2002.

[21] M. Shirvanian and N. Saxena, 'Wiretapping via mimicry: Short voice imitation man-in-the-middle attacks on crypto phones', in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2014, pp. 868–879.

[22] D. Mukhopadhyay, M. Shirvanian and N. Saxena, 'All your voices are belong to us: Stealing voices to fool humans and machines', in *European Symposium on Research in Computer Security*, Springer, 2015, pp. 599–621.

[23] S. Chen, K. Ren, S. Piao, C. Wang, Q. Wang, J. Weng, L. Su and A. Mohaisen, 'You can hear but you cannot steal: Defending against voice impersonation attacks on smartphones', in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2017, pp. 183–195.

[24] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre and H. Li, 'Spoofing and countermeasures for speaker verification: A survey', *speech communication*, vol. 66, pp. 130–153, 2015.

[25] R. L. Peterson and B. F. Pennington, 'Developmental dyslexia', *The Lancet*, vol. 379, no. 9830, pp. 1997–2007, 2012.

[26] *Mysql 5.5 reference manual :: 12.5 string functions and operators*. [Online]. Available: https://dev.mysql.com/doc/refman/5.5/en/string-functions.html#function_soundex.

[27] *Oracle - database sql reference*, Jul. 2005. [Online]. Available: https://docs.oracle.com/cd/B19306_01/server.102/b14200/functions148.htm.

[28] *F.15. fuzzystrmatch*. [Online]. Available: https://www.postgresql.org/docs/9.1/fuzzystrmatch.html.

[29] F. Patman and L. Shaefer, 'Is soundex good enough for you? on the hidden risks of soundex-based name searching', *Language Analysis Systems, Inc., Herndon*, 2001.

[30] L. Philips, 'Hanging on the metaphone', *Computer Language*, vol. 7, no. 12, pp. 39–43, 1990.

[31] *Metaphone*. [Online]. Available: https://www.php.net/manual/en/function.metaphone.php.

[32] G. P. Hettiarachchi and D. Attygalle, 'Sparcl: An improved approach for matching sinhalese words and names in record clustering and linkage', in *2012 IEEE Global Humanitarian Technology Conference*, IEEE, 2012, pp. 423–428.

[33] A. Parrish, 'Poetic sound similarity vectors using phonetic features', in *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2017.

[34] N. Chomsky and M. Halle, 'The sound pattern of english.', 1968.

[35] P. Ladefoged, 'The measurement of phonetic similarity', in *INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS COLING 1969: Preprint No. 57*, 1969.

[36] A. Bradlow, C. Clopper, R. Smiljanic and M. A. Walter, 'A perceptual phonetic similarity space for languages: Evidence from five native language listener groups', *Speech Communication*, vol. 52, no. 11-12, pp. 930–942, 2010.

[37] D. Difallah, E. Filatova and P. Ipeirotis, 'Demographics and dynamics of mechanical turk workers', in *Proceedings of the eleventh acm international conference on web search and data mining*, ACM, 2018, pp. 135–143.

[38] M. Cherubini, A. Meylan, B. Chapuis, M. Humbert, I. Bilogrevic and K. Huguenin, 'Towards usable checksums: Automating the integrity verification of web downloads for the masses', in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2018, pp. 1256–1271.