Investigating the Security of p≡p's Trustwords

Aidan Fray

University of York

September 2019

***Abstract* – Many encrypted connections require the comparison of a fingerprint to protect against eavesdropping. A substantial amount of past work has aimed to propose fingerprint representations that work better with hu- man limitations. "Trustwords" proposed by pEp is an example of such a scheme, where the fingerprint is encoded as words in an attempt to improve usability.**

This works main aim is to assess if Trustword's recommended minimum number of four words is sufficient. In order to achieve this goal, this work implements an attack on Trustwords and quantifies its effectiveness on more than 400 participants. A tool called GreenOnion was designed to assist in quantifying attack feasibility. GreenOnion improved substantially on a similar tool's ability to search for matches concurrently where it was able to search for over 1.5 million concurrent keys. Our findings show a substantial increase in attack success compared to related literature. We believe this increase is due to levering the design flaws in the Trustword scheme. Therefore, we believe a minimum of four Trustwords is insufficient to provide even a basic level of security.

1 Introduction

The increasing use of public-key cryptography by instant messaging and secure email means ensuring confidentiality 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 entirely circumvent the encryption as it allows an attacker to read all of the encrypted data. A countermeasure for the threat of MiTM attacks is the verification of each parties' fingerprint. A fingerprint is a small string of characters that is unique to each key and, thus, can be used to identify.

Fingerprints can come in several 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 (± 2) worth of data in their working memory[1]. Consequently, this makes the designing of user-friendly schemes a task of utmost importance.

Humans are commonly considered the most vulnerable part of any computer system. Solutions, therefore, have been proposed to remove this 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. Manual verification is, consequently, left in a difficult position, due in part to proposed solutions needing to sacrifice either usability or security. Therefore, research into improving or creating a more secure and user-friendly fingerprint encoding remains an important task.

One proposed user-friendly scheme is Pretty Easy Privacy's (p≡p) "Trustwords". Trustwords is an implementation of word fingerprint mapping with a emphasis on usability. This increase in usability is achieved by the user comparing a reduced number of words. This usability boost, however, comes at a cost of a much larger word list.

This report evaluates the security and feasibility of attacking Trustwords. The motivation for this is due in part to the shortage of justification behind the size and features of the chosen wordlist.

2 Background

2.1 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. Hence, this means that sharing the keys required to encrypt can be performed across insecure channels.

An E2EE connection uses the key types discussed previously to ex- change messages between verified parties securely. However, this hinges on the verification of the initial parties, if one party impersonates another and receives sensitive communication, the secure encryption is completely circumvented. Therefore, the correct identification of parties is crucial to maintaining security. The exploitation of this is known as a Man-in-the-middle (MiTM) attack. This attack is commonly bidirectional, and if executed correctly, there is often no discerning change to the user experience.

2.2 Similarity Metrics

A similarity metric is an algorithm designed to determine if two words are phonetically a match. For example, the words ”THEIR” and ”THERE” is a match, whereas the words ”DARK” and ”PRINCIPLE” are not phonetically matching. This section provides a base explanation of important phonetic algorithms relevant in this project.

Soundex

One of the earliest examples of a phonetic algorithm is known as Soundex. It was initially designed for phonetically indexing names alongside detection of transposed letters in spelling mistakes. Soundex is one of the most famous example of a phonetic. This is due in part to its implementation into major database clients like MySQL[4], Oracle[5] and PostgreSQL[6].

NYSIIS

The New York State Identification and Intelligence System (NYSIIS) phonetic code was created for the phonetic matching of American names. The motivation for its inception was mostly due to the presence of Hispanic names in the American based databases (this was an aspect Soundex was known to have low accuracy with).

Metaphone

Metaphone was invented by Lawrence Philips in 1990[7] 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.

Levenshtein Distance

Levenshtein distance is a string metric designed to measure the ‘distance’ between two strings. It is merely the number of single-character edits (insertions, deletions or substitutions) required to reach the other string.

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.

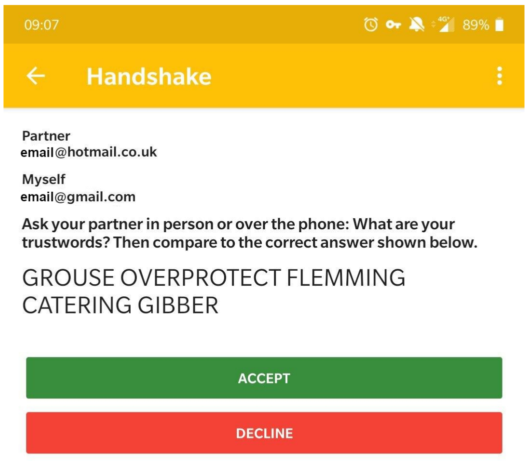
Phonetic Vectors

Phonetic Vectors is a unique addition to the chosen set. Created by Allison Parrish in 2017[8], Phonetic vectors is as the name suggests the vectorisation of the phonetics of a word. Phonetic features are used in this work as a way to compare the similarity of phonemes. Phonemes are the phonetic elements that construct a word. For example, the word ”RING” translated into the phonemes `/R IH NG/`.

2.3 Pretty Easy Privacy

A very recent implementation of a word list can be found in Pretty Easy Privacy (p≡p) implementation of TrustWords[9]. 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 embedded design principles state that above all the systems should be 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 1 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 accept or decline respectively.



Partner
email@hotmail.co.uk

Myself
email@gmail.com

Ask your partner in person or over the phone: What are your trustwords? Then compare to the correct answer shown below.

GROUSE OVERPROTECT FLEMMING CATERING GIBBER

Figure 1: Trustword fingerprint verification

The unique aspect of TrustWords is its mapping of a single word to 16-bits. If compared to alternative 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. This number of words is arguably higher than most users' vocabulary. This deviation from the norm has not been currently backed up by research.

3 Design

3.1 Attack Design

The attack on Trustwords involves generating "near-collision" keys. Near-collision keys are keys composed of a set of words that are deemed a match by the similarity metric.

As each combined key in Trustwords is an exclusive-or of both sides' public-key the attack is designed to target a single pair of users and requires recomputation for each attack target. Each pair is split into an "Uncontrolled" or "Controlled" key. Uncontrolled is the receiver of the communication, and, thus, the key cannot be altered. The Controlled key is the one we are attempting to impersonate. It is assumed that there is the ability to replace the Controlled key with the malicious option. The uncontrolled and controlled keys can be swapped around, thus, resulting in the possibility to intercept both directions of communication. This, however, requires performing the attack twice.

When attacking, a similarity metric is used to compute a list of possibilities for each position in the target fingerprint. Completing these steps produces a list of fingerprints that can be inserted into a tool designed to hash a large number of keys and search for matches. This aspect of using an extensive 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 near-collision key fingerprints.
4. Use a list of near-collision keys in the mass computation of keys to find near-collision keys.

3.2 GreenOnion Design

The inspiration for the design of this tool was taken from a tool called Scallion[1]. Scallion was designed by Richard Klafter and Eric Swanson and was used to demonstrate that 32-bit PGP key IDs were insufficient. To keep with the onion-based theme, the

[1] `https://github.com/lachesis/scallion`

proposed tool is called 'GreenOnion' and is a rewrite of Scallion 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 are 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 key fingerprints. This list is then used as a search criterion when searching for keys. To utilise the parallel nature of the GPU to compute the hash of a large number of keys, the tool utilises 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 provides a massive speed increase compared to the sequential nature of the CPU.

4 Experimental Design

4.1 Metric performance

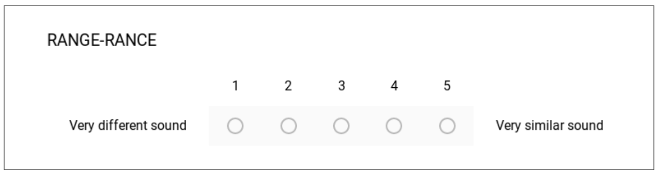
This experiment aims to reduce the number of metrics to assess in the later rounds. The thinning out of metrics is required due to limited resources. Therefore, possible further work could involve repeating this work with a much more varied selection of metrics. The chosen metrics up for assessment are 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 (Very different sound) to 5 (Very similar sound).

Comparisons

Each comparison contains a match-pair. These matches are generated by running the similarity metric over all the pairings in the Trustword dictionary; if the pair meets the criteria discussed in the previous section, it is determined a match. Potential matches are then sampled from their respective lists.

Each participant is then asked to rate the similarity of a match on a scale of 1 (*'Very different sound'*) and 5 (*'Very similar sound'*). Figure 2 shows an example match and the connected scale. Users are provided with five of these comparisons per similarity metric.


RANGE-RANCE

	1	2	3	4	5	
Very different sound	○	○	○	○	○	Very similar sound

Figure 2: Example experiment question

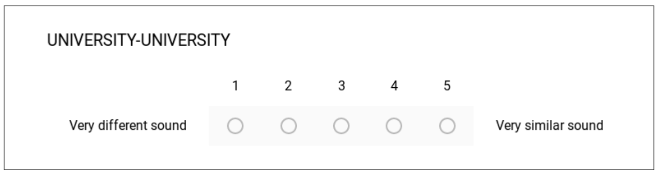
The experiment randomises the order of these comparisons per session alongside a complete refreshing of matches once per submission. This design makes sure selections from the samples are fair and consistent as each user has a different selection of matches.

Quality control

As the study was being outsourced to Amazon's Mechanical Turk, the requirement to check the quality of results is essential. 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 one. This design allows for a simple check for valid results. If their average rating for Random matches is too high, their results are discarded.

Alongside this, was the addition of two questions comparing precisely 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 are discarded. Figure 3 contains one example of the attention question used to filter inaccurate results.


UNIVERSITY-UNIVERSITY

	1	2	3	4	5	
Very different sound	○	○	○	○	○	Very similar sound

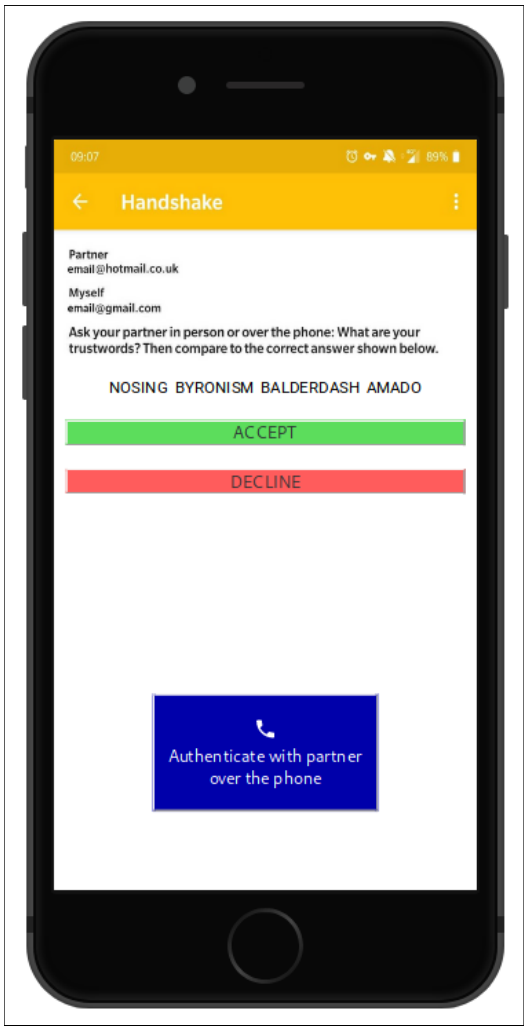
Figure 3: Exact experiment attention question

The final check to ensure the validity of results is a check for native English fluency. Having non-native English speakers complete the experiment can introduce inconsistencies into the data and, therefore, needs to remain controlled. To achieve this, a preliminary MTurk study has been created to ask users for their perceived English fluency on a scale of 1 (Basic Understanding) to 5 (Native). Workers with an answer of 5 are provided with a Qualification that tags them as defining themselves as native speakers. This qualification is then added as a prerequisite when running the experiment. This prerequisite ensures 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 of this initial study, thereby providing inaccurate results has no real valid motive for the worker.

4.2 Trustword Attacks

This experiment is designed to be a simulation of the attack proposed in Section 3.1. The experiment tests the simulated matches of three similarity metrics decided by the previous experiment. The overall aim of the experiment is to quantify the success rate of the attack on live participants. Figure 4 contains a view of the user interface used in the experiment. It has been designed to be as similar as the p≡p Android application.

A live version of the application is avaliable[2]. The blue button is clicked to simulate the authentication ceremony over the phone. A text-to-speech system then reads a set of words. The user should accept if they match and decline if they don't.



Partner
email@hotmail.co.uk

Myself
email@gmail.com

Ask your partner in person or over the phone: What are your trustwords? Then compare to the correct answer shown below.

NOSING BYRONISM BALDERDASH AMADO

Figure 4: Trustword user interface

4.2.1 Design

Due to the scarcity of attacks in a real-world setting, users are often complacent about their occurrence. Therefore, in this experiment, attacks only occur 30% of the time. This occurrence is a much higher value than in a real-world setting but is a balance between resources and realistic design. If the chosen attack rate is too low, many more trials are required to collate a useful number of attack instances, thus, requiring more resources, but with more realism. If, however, the attack rate is too high, less total rounds are required to obtain the target number of attack trails. However, the user's complacency is lost as an attack is expected. This loss of complacency is undesirable as it does not realistically reflect real life. Furthermore, the first 5 trials of a new experiment are always benign. This inclusion is to ensure users are consistently lulled into complacency.

Certain keys have higher levels of potential near-collision keys than others due to how certain words are deemed similar by the similarity metric. Therefore, if random sets of words are 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 is designed to sample from a list of "vulnerable" keys. These are keys that have a number of potential near-collision keys that allow computation in a specified timeframe.

Furthermore, certain levels of attacks are also simulated. Below is the list of attacks considered in

[2]https://afray.pythonanywhere.com/

this experiment:

- (`OOOO`) All words in the set can vary
- (`XOOO`) All but the first word can vary
- (`XOOX`) All but the first and last word can vary

Where :

`X`: is a static word

`O`: is a changeable word

The start and ends were chosen as the highest priority words to keep static. This design choice is due to research highlighting the common habit of users only to check the start and end of a checksum.

These attacks are ascending in complexity. The timeframe for computing a near-collision was decided as one week with attack strengths of 7 GPU-days *(1 x GPU)*, 70 GPU-days *(10 x GPUs)* and 700 GPU-days *(100 x GPUs)* on a mid-range GPU (AMD RX 480). Due to the long lifetime of PGP keys, 7 days was chosen as a reasonable length of the attack. The tool requires no user interaction once started, therefore, it can be left for an extended period to compute potential keys. This timeframe is also a maximum time, as these attacks also include keys that have a shorter average computation time.

Table **??** contains a summary of the different level of attacks and their respective restrictions. Any key that exceeds the defined numbers of potential near-collision key for attack level is deemed as vulnerable. A list of these vulnerable keys is sampled from when an attack is simulated.

4.2.2 Quality control

Like the previous experiment, participants are sourced from MTurk. Therefore, again, quality control is an aspect that requires consideration. As with the previous study, initially screened native speakers are used. Alongside this, two metrics are used to detect invalid results: audio-button-clicks and the overall-time.

Audio-buttons-clicks is the number of times the blue *"Authenticate with partner over the phone"* button in Figure 4 is clicked. If there are rounds without button clicks, this is a sign of non-attentiveness. The design choice was made to allow non-clicks. This design provides a way to detect and discard click-throughs[3].

Overall-time is as the name suggest a recording of the time taken to complete the entire experiment. If users complete the experiment in an abnormally small amount of time, it can be used as another detection for invalid responses. A benchmark for the time is set by completing the experiment as fast as possible. Anything quicker was discarded as anything faster risks invalid and rushed results.

5 Results

5.1 Scallion vs GreenOnion

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

Figure 5.1 shows the speed decline as the number of concurrently checked potential keys increases. Alongside the results for base version of Scallion is the speed for *Scallion Improved.* This is the same Scallion code but with a fix that reduces the severity of the speed decline. Scallion contained a call in its print statement that became become increasingly expensive as the number of concurrently checked keys increases. Removing this line from the print statement results in a sizeable improvement in speeds as can be seen in Figure 5.1. This was necessary to include as we believe the improved versions show a better comparison between the design of the two tools.

GreenOnion's speed is almost perfectly consistent. It is slower up until around 800 keys. This initial lower speed is due to the overhead of the bloom filter; however, as the number of keys increase, the bloom filter's benefits become apparent. In this test, the bloom filter size was kept consistent at exactly 10,000 elements.

Scallion was unable to handle anything over 2513 concurrent keys. This was due to the search strings being passed via a command-line argument as a regex string. This number of keys hit the character limit of a Powershell command. GreenOnion, however, has been tested to handle more than 1.5 million keys with a slight reduction in performance.

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

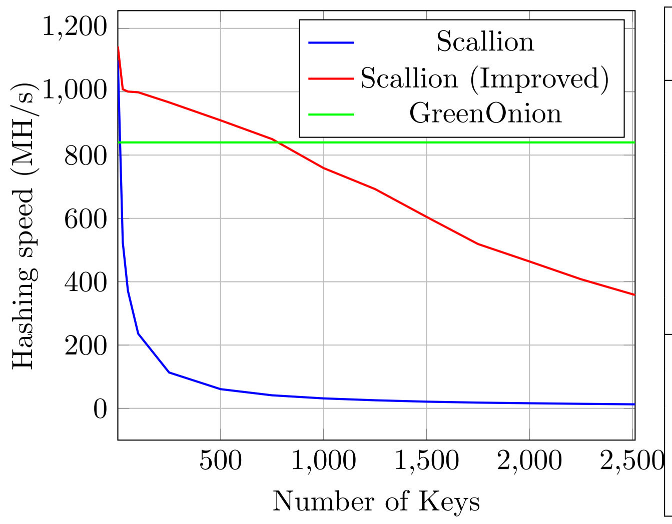


Figure 5: Speed comparison between Scallion and GreenOnion

This improvement over Scallion is, therefore, substantial. GreenOnion's code has been made publicly available and can be viewed on GitHub[4].

5.2 Experiment 1 - Metric performance

The goal of this experiment was to select a set of metrics to be assessed in the subsequent experiment. This section discusses the demographics of participants alongside the subsequent 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 4.1) or having too low of a fluency rating. This dismissal was a necessary process to improve the health of the results.

Table 1 contains the demographical breakdown of the reduced set of participants. The average ages of the participants were 37.3 years ($\sigma = 11.67$) with a split of 44.4% of Males to 55.6% of Females. As can be seen, over 60% of participants can be considered highly educated (Bachelor's and up). This level of education is not sufficiently 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 broad range of dialects present that may bias the results. Further work could investigate the effect of location and dialect on similar results.

[4]https://github.com/AidanFray/GreenOnion

Gender	Male:	44.4%
	Female:	55.6%
Age:	18-24:	10.1%
	25-29:	20.2%
	30-39:	31.3%
	40-49:	21.2%
	50-59:	12.1%
	60-69:	4.0%
	70-79:	1.0%
Highest Education:	GCSE:	13.1%
	A-Level/O-Level:	19.2%
	Bachelor's degree:	52.5%
	Master's degree:	13.1%
	PhD:	2.0%

Table 1: Participant demographics

Figure 2 shows the average results for the selected matches.. It can be seen that Levenshtein came out substantially above the rest. The breakdown of the ratings in Figure 11 also shows Levenshtein's dominance. Levenshtein has a much more significant proportion of 4 and 5 ratings than the alternatives.

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

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 2: Average metric performance

6 Experiment 2 - Trustword attacks

This experiment aims to quantify the success rate of the proposed attack. Design details for this experiment can be seen in Section 4.2. This section presents and discusses the results of the experiment alongside a comparison to relevant literature.

Overall, 435 paid participants recruited via Amazon's MTurk were assessed in this experiment. We

Metric	**Successful Attacks**	**Total Attacks**	**Success Rate**
Levenshtein	218	1101	19.8%
`OOOO`	34	358	9.5%
`XOOO`	59	353	16.7%
`XOOX`	125	390	32.1%
Metaphone	181	1072	16.9%
`OOOO`	38	345	11.0%
`XOOO`	57	375	15.2%
`XOOX`	86	352	24.4%
NYSIIS	209	1114	18.8%
`OOOO`	36	385	9.3%
`XOOO`	72	375	19.2%
`XOOX`	101	354	28.5%
Overall	608	3287	18.5%

Table 3: Success rates for simulated attacks

excluded 66 results; 7 due to being non-native speakers and 59 were discarded for failing the attention metrics (Discussed in Section 4.2.2)

Gender	Male:	50.4%
	Female:	49.6%
Age:	18-24:	12.7%
	25-29:	18.2%
	30-39:	37.4%
	40-49:	17.9%
	50-59:	8.7%
	60-69:	4.3%
	70-79:	0.8%
Highest Education:	GCSE:	13.8%
	A-Level/O-Level:	24.1%
	Bachelor's degree:	51.5%
	Master's degree:	8.4%
	PhD:	2.2%

Table 4: Participant demographics

The reduced set of 369 participants had an average age of 36.6 ($\sigma = 11.35$) and consisted of an almost equal split of Male (50.4%) to Female (49.6%). Around 62% of participants had completed a single stage of university (Bachelor and up). This aspect makes this set of participants more educated than the general population. All participants were also sourced from the USA and have rated themselves as fully native English speakers.

Table 3 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 19.8%. The best performer, when regarding attack strength, as expected, is the `XOOX` attack. Levenshtein's `XOOX` attack performed the best overall with a success rate of 32.1%. The worst performer was Metaphone, with an average of 16.9% 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 same order.

Comparison to alternative literature

This section compares the results of the experiment to similar literature.

Work by **R. Kainda, I. Flechais** and **A. Roscoe**[10] is the most comparable to this experiment. They were the only study to use exactly 4 words but differed on the size of the dictionary (1024 words) and how a near-match were 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 aspects 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 **S. Dechand *el al.***[11] assess the success rate of attacks on words. Their experiment is less similar as 14 words per attack were assessed with each participant, where the comparison was performed visually. However, the success was 8.78%. As 10/14 words were kept static, this result is more comparable to the highest attack strength (`XOOX`). This, therefore, leads to another 3 fold improvement in the success rates of our simulated attacks.

The final relevant paper to compare is that of **J. Tan *et al.***[12]. 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 **S. Dechand *et al.***[11], the attack success was 14% overall. This success rate compared to the highest attack assessed in the experiment results in another sizeable difference.

In conclusion, all related literature had much lower attack success rates that the results presented in this experiment. With all papers using similarly designed wordlists, 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 the available literature.

7 Discussion and Further Work

7.1 Further work

Trustword improvements

The first area of proposed work could be recommendations into how to improve Trustwords backed by empirical evidence. We feel the most promising avenue would be the utilisation of Phonetic Vectors unique qualities to identify words of low quality by measuring very low vector distances. For example, present in the dictionary are the words `THERE` and `THEIR`, these could be massively improved upon and could result in a better quality word list. Moreover, utilising the quantification of dissimilarity could be used to create a dictionary of maximized phonetic distance. This phonetically distinct wordlist could then be assessed in a similar way with its success rate quantified against users.

Similar metrics performance

Another area of further work is the comprehensive assessment of algorithms used to assess phonetic similarity. The algorithms assessed in this work are a very small proportion of potential options. Thus, further work could assess the performance of metrics against each other to determine the one that works best with human models. A conclusion could be reached by running the improved experiment discussed in Section **??**, where it could be repeated with an increased number of metrics. Other elements for consideration could be age, location and dialect as variables that affect a metrics performance.

GreenOnion optimizations

A minimal amount of work was allocated to improving the performance of GreenOnion as low-level optimizations are very time-consuming. A project, therefore, could aim to improve on the performance already recorded in this paper. This project could improve the performance of the attack and allow for more effective keys to be computed in less time.

7.2 Conclusion

Overall, the project has demonstrated a potential attack on p≡p 's implementation of Trustwords. An attack was proposed that utilised phonetic similarity algorithms to exploit the weaknesses in the Trustword dictionary to generate near-collision keys. To achieve this, a tool was created to compute a massive number of keys concurrently. It was based on a known tool but improved substantially on its key searching performance. The performance of similarity metrics was compared alongside an experiment assessing participants fallibility to the proposed attack. Results showed as much as a 32.05% success rate for the best attack. The main project aim was to show that the recommend minimum number of four Trustwords was insufficient to provide a basic level of security. We believe we have demonstrated that four Trustwords is too low to provide enough security for general use-case and the minimum provided words should, therefore, be increased.

References

[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] "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

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

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

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

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

[9] "Iana registration of trustword lists: Guide, template and iana considerations." [Online]. Available: https://tools.ietf.org/html/draft-birk-pep-trustwords-03

[10] 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.

[11] 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.

[12] 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.

A Experiment 1 - Results

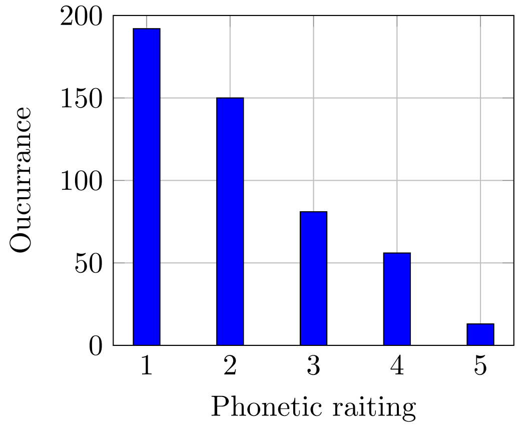


Figure 6: Soundex

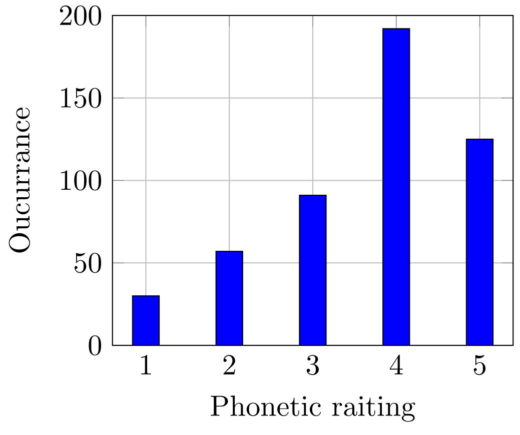


Figure 7: Levenshtein

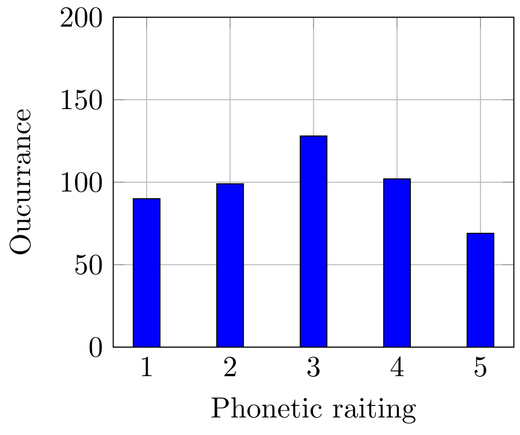


Figure 8: NYSIIS

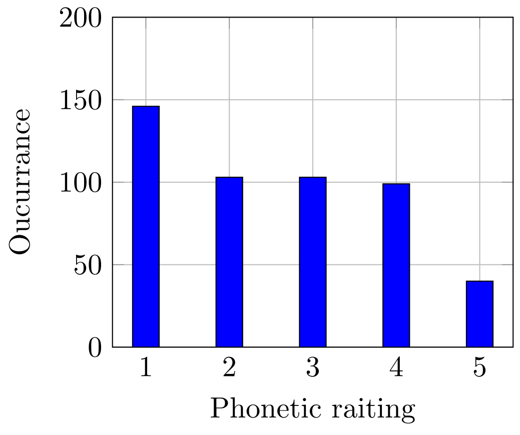


Figure 9: Metaphone

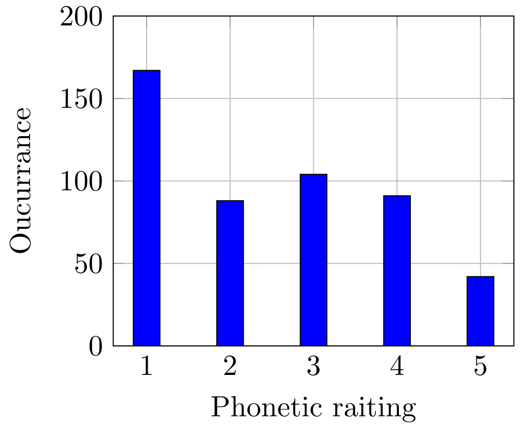


Figure 10: Phonetic vector

Figure 11: Individual breakdown of results for each metric