An Empirical Study of Textual Key-Fingerprint Representations

Sergej Dechand
USECAP, University of Bonn

Dominik Schürmann
IBR, TU Braunschweig

Karoline Busse
USECAP, University of Bonn

Yasemin Acar
CISPA, Saarland University

Sascha Fahl
CISPA, Saarland University

Matthew Smith
USECAP, University of Bonn

Abstract

Many security protocols still rely on manual fingerprint comparisons for authentication. The most well-known and widely used key-fingerprint representation are hexadecimal strings as used in various security tools. With the introduction of end-to-end security in WhatsApp and other messengers, the discussion on how to best represent key-fingerprints for users is receiving a lot of interest.

We conduct a 1047 participant study evaluating six different textual key-fingerprint representations with regards to their performance and usability. We focus on textual fingerprints as the most robust and deployable representation.

Our findings show that the currently used hexadecimal representation is more prone to partial preimage attacks in comparison to others. Based on our findings, we make the recommendation that two alternative representations should be adopted. The highest attack detection rate and best usability perception is achieved with a sentence-based encoding. If language-based representations are not acceptable, a simple numeric approach still outperforms the hexadecimal representation.

1 Introduction

Public key cryptography is a common method for authentication in secure end-to-end communication and has been a part of the Internet throughout the last two decades [7, 11]. While security breaches have shown that systems based on centralized trusted third parties such as Certificate Authorities and Identity Based Private Key Generators are prone to targeted attacks [42], decentralized approaches such as Web of Trust and Namecoin struggle with beeing adopted in practice due to usability issues [7, 13, 30]. Certificate transparency systems, such as CONIKS and others [24, 39, 27], aim to solve a subset of these issues by providing an auditable directory of all user keys. Still, manual key verification, i. e., the link between public keys and the entities, such as hostnames or people, remains a challenging subject, especially in decentralized systems without pre-defined authorities, such as SSH, OpenPGP, and secure messaging [12, 41].

Many traditional authentication systems still rely on manual key-fingerprint comparisons [17]. Here, key-fingerprints are generated by encoding the (hashed) public key material into a human readable format, usually encoded in hexadecimal representation. A variety of alternatives such as QR Codes, visual fingerprints, Near Field Communication (NFC), and Short Authentication Strings (SAS) have been proposed. Most of these systems offer specific benefits, e. g., QR codes and NFC do not require users to compare strings, but they also come with specific disadvantages, e. g., they require hardware and software support on all devices. While advances are being made in these areas, the text-based representation is still the dominant form in most applications.

However, due to the recent boom of secure messaging tools, the debate of how to best represent and evaluate textual fingerprints has opened up again and there are many very active discussions among security experts [28, 33]. In April 2016, WhatsApp serving over one billion users enabled end-to-end encryption as default by implementing the Signal protocol. Key verification is optional and can be done by using QR codes or comparing numeric representations, in their case 60-digit numbers [43]. However, it is not clear whether their solution is more usable than traditional representations.

In this paper, we present an evaluation of different textual key-fingerprint representation schemes to aid in the secure messenger discussion. The requirements posed to the developers are as follows:

- The fingerprint representation scheme should provide offline support and work asynchronously. One reason for this is that fingerprints are often printed on business cards or exchanged by third parties.

- The fingerprint should be transferable via audio channels, e. g., it should be possible to compare fingerprint over the phone.

- The representation scheme should be as technically inclusive as possible. No special hardware or software should be required to verify the fingerprints: both require a concerted and coordinated effort between many actors to get enough coverage for a comparison mechanism to be worthwhile for users to adopt.

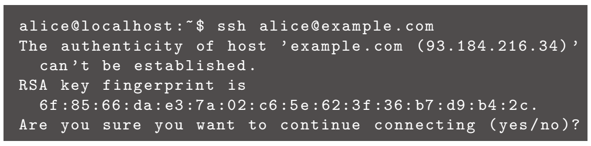
- The representation should be as inclusive as possible, i. e., excluding as few people with sensory impairments (visual, color, audio, etc.) as possible.

The above requirements exclude many proposed representation schemes and offer an explanation why they have not seen any adoption outside of academia. For this reason, we focus exclusively on textual fingerprint representations in our study. Textual key-fingerprints do not require hardware support and work in synchronous and asynchronous scenarios, i. e., they can be compared via voice or printed on business cards. Depending on the scheme, they even could be recalled from memory and exchanged over a voice channel.

This paper presents our study testing the usability of various textual key-fingerprint representation schemes. Our study consists of two parts: (1) an experiment where we measured how fast and accurate participants perform for different schemes, and (2) a survey about their perception and sentiment. These also contained a direct comparison between the representations.

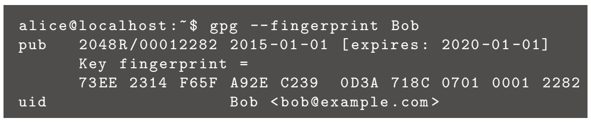
Our findings suggest that the most adopted alphanumeric approaches such as the Hexadecimal and Base32 scheme perform worse than other alternatives: under a realistic threat model, more than 10% of the users failed to detect attacks targeting Hexadecimal representations, whereas our best system had failure rates of less than 3%. While the best system for accuracy is not the fastest, it is the system which received the highest usability rating and is preferred by users.

In the following sections, we discuss related work followed by an analysis of current implementations deploying in-persona key-fingerprint representation techniques and discuss our evaluated representation schemes. Then, we describe our experiment evaluating text-based key-fingerprint verification techniques with regards to their attack-detection accuracy and speed. Our experiment was conducted as an online study with 1047 participants recruited via the Amazon Mechanical Turk (MTurk) platform. We consider the scenario outlined above, where a user compares two key-fingerprint strings encoded by the different representation schemes. In addition to the implicit measurements of accuracy and speed, we also



```
alice@localhost:~$ ssh alice@example.com
The authenticity of host 'example.com (93.184.216.34)'
  can't be established.
RSA key fingerprint is
  6f:85:66:da:e3:7a:02:c6:5e:62:3f:36:b7:d9:b4:2c.
Are you sure you want to continue connecting (yes/no)?
```

(a) OpenSSH: Lowercase Hexadecimal with Colons



```
alice@localhost:~$ gpg --fingerprint Bob
pub   2048R/00012282 2015-01-01 [expires: 2020-01-01]
      Key fingerprint =
      73EE 2314 F65F A92E C239  0D3A 718C 0701 0001 2282
uid                  Bob <bob@example.com>
```

(b) GnuPG: Uppercase Hexadecimal with Spaces

Figure 1: Alphanumeric Fingerprints Used in Practice

evaluate the self-reported user perception to get feedback about which systems are preferred by end users. Finally, we present our results, discuss their implications and takeaways, and conclude our work.

2 Related Work

Various key-fingerprint representations have been proposed in academia and industry. Various cryptographic protocol implementations still rely on manual fingerprint comparisons, while the hexadecimal representation is used in most of them. However, previous work suggests that fingerprint verifications are seldom done in practice [17, 37].

2.1 Key-Fingerprint Representations

Previous work has shown that users struggle with comparing long and seemingly "meaningless" fingerprints and it is suspected that they even might perform poorly in this task [19]. While most previous work has focused on the family of visual fingerprints [35, 32, 19, 10], to our knowledge, none of those focused on the differences between various different textual fingerprint representations.

Hsiao et al. have conducted a study with some textual and visual representation methods for hash verification [19]. They compared Base32 and simple word list representations with various algorithms for visual fingerprints and hash representation with Asian character sets (a subset of Chinese, Japanese Hiragana, and Korean Hangul, respectively). A within-subjects online study with 436 participants revealed that visual fingerprints score very well in both accuracy and speed, together with the Base32 text representation. Hsiao et al. conclude that depending on the available computation power and display size, either Base32 or one of the visual fingerprinting schemes should be used. They explicitly did not evaluate hexadecimal representation or digits "because that scheme is similar to Base32 and known to be error-prone" [19]. However, our work shows that numeric representations actually perform significantly better than Base32 and is less error prone. In addition, our results suggest that language-based schemes, e. g., generated sentences achieve excellent results comparable to visual schemes. At the same time, textual approaches are more flexible (can be read out loud) and do not exclude people with sensory impairments.

Another study by Olembo et al. also focused mainly on the topic of visual fingerprints [32]. They developed a new family of visual fingerprints and compared them against a Base32 representation. The Base32 strings were twelve characters long and displayed without chunking. The participants performed better with the visual fingerprints than with Base32, regarding both accuracy and speed. Olembo et al. conclude that the Base32 representation is far away from optimal when it comes to manual key-fingerprint verification. We test this claim by comparing Base32 representation with other textual key-fingerprint representation and eventually prove it wrong.

Regarding chunking, Miller et al. have published *The magical number seven* and succeeding work that shows that most people can recall 7 ± 2 items from their memory span [29]. It has been shown that although there are slight differences between numbers, letters and words (numbers perform slightly better than letters, and letters slightly better than words), they perform similar in studies. More recent studies have shown that human working memory easily remembers up to 6 digits, 5.6 letters and 5.2 words [1, 6, 8]. Adjusting chunk sizes to these numbers can help users when comparing hashes.

While all of the above studies offer interesting insights into different (mainly visual) fingerprint representations, to the best of our knowledge there is not work focusing on which textual representation performs the best. However, this knowledge would be extremely important to help in the current debate in the secure messaging community. The representations currently being put forward and implemented are far from optimal and the results of our study can help improve the accuracy and usability of fingerprint representations. Unlike the above studies we conduct our study with a more realistic attacker strenth, as presented in subsection 4.1).

2.2 Passwords and Passphrases

A passphrase is basically a password consisting of a series of words rather than characters. In academic literature, passphrases are often considered as a potentially more memorable and more secure alternative to passwords and are often recommended by system administrators [23, 40]. In contrast to most passphrase-based systems, key-fingerprints cannot be chosen by the end-user and thus are more related to the system-assigned passphrases field: Bonneau et al. have shown that users are able to memorize 56-bit passwords [4]. miniLock[1] and its commercial successor Peerio[2] use system-assigned passphrases to generate cryptographic key pairs easing key backup and synchronization among multiple devices.

Scheme	Example

Hexadecimal	`18e2 55fd b51b c808`
	`601b ee5c 2d69`

Base32	`ddrf l7nv dpea`
	`qya3 5zoc 22i`

Numeric	`2016 507 6420 1070 394`
	`1136 2973 991 70`

PGP	locale voyager waffle disable Belfast performance slingshot Ohio spearhead coherence hamlet liberty reform hamburger
Peerio	bates talking duke rummy slurps iced farce pound day
Sentences	Your line works for this kind power cruelly. That lazy snow agrees upon our tall offer.

Table 1: Examples for different textual key-fingerprint representations for the same hash value

Contrary to widespread expectations, Shay et al. were not able to find any significant recall differences between system-assigned passphrases and system-assigned passwords [40]. However, they reported reduced usability due to longer submission times due to typing.

Similar to passphrases, the usage of language-based key-fingerprint representations is claimed to provide better memorability than just an arbitrary series of character strings despite the lack of empirical evidence. In our study, we measure the performance of the different approaches and also collect perception and feedback from end users.

3 Background

In the past years, various textual key-fingerprint representations have been proposed. In this section, we analyze currently practised in-persona key verification techniques in well-known applications. For comparison, Table 1 lists the approaches we used in our evaluation generated from the same hash value.

Only applications requiring manual key-fingerprint

[1] `https://minilock.io`
[2] `https://peerio.com`

verification are considered. In mechanisms like S/MIME or X.509, fingerprints play only a secondary role because certificates are verified via certificate chains.

In the following, $\mathit{SHA\text{-}1}(x)^{16}$ defines the execution of 16 rounds of nested *SHA-1* on x, a truncation to the left-most 16 bits is defined by $x[0,\ldots,16]$, and pk is used as an abbreviation for the values of a public key (differs for RSA, DSA, or ECC).

3.1 Numeric

Numeric representation describes the notation of data using only numeric digits (0-9). The primary advantage of a such system is that Arabic numerals are universally understood, and in addition, numeric key-fingerprints show a similarity to phone numbers. The encoding is achieved by splitting a binary hash into chunks of equal length and expressing each chunk as a decimal number, e. g., by simply switching the representation base from 2 to 10.

The messaging and data exchange application SafeSlinger[3] implements this as a fallback scheme for unsupported languages [14]. A 24 bit SAS in SafeSlinger (cf. Figure 2a) can be expressed by three decimal encoded 8-bit numbers.

In the messaging platform WhatsApp, a fingerprint is calculated by $\mathit{SHA\text{-}256}(pk)^{5200}[0,\ldots,240]$. This fingerprint is split up into six chunks, where each chunk is represented by a five digits long number modulo 100,000 [43]. Concatenating this fingerprint with the fingerprint of the communication partner results in the displayed representation, e. g.,

```
77658 87428 72099 51303
34908 23247 95615 27317
09725 59699 62543 54320
```

3.2 Alphanumeric

Alphanumeric approaches use numbers and letters to represent data. Depending on the representation type and its parameters, the letters can be presented either in lower-case or in upper-case. The string can be chunked into groups of characters, which are usually of equal length. Chunking does not alter the information contained, while changing lower-case letters to upper-case letters (and vice versa) may does, depending on the coding scheme. Commonly used representations are Hexadecimal, Base32, and Base64.

3.2.1 Hexadecimal

Hexadecimal digits use the letters A-F in addition to numerical digits and are a common representation for key-fingerprints and primarily used in SSH and OpenPGP. Note that the case of the letters do not make any difference. Regarding chunking, both spaces (cf. Figure 1b) and colons (cf. Figure 1a) are commonly used as separation characters.

[3]`https://www.cylab.cmu.edu/safeslinger`

Key fingerprints in OpenPGP version 4 are defined in RFC 4880 [7] by

$$\mathit{Hex}(\mathit{SHA\text{-}1}(0x99 \,\|\, \mathit{len} \,\|\, \mathit{4} \,\|\, \mathit{creation\_time} \,\|\, \mathit{algo} \,\|\, \mathit{pk}))$$

where *len* is the length of the packet, *creation_time* is the time the key has been created and *algo* is unique identifier for the public-key algorithm. While the inclusion of *creation_time* makes sure that even two keys with the same key material have different fingerprints, it allows an attacker to iterate through possible past times to generate similar fingerprints skipping the key generation step [5]. The actual representation of OpenPGP fingerprints is not defined in RFC 4880, but most implementations chose to encode them in hexadecimal form, e. g., GnuPG displays them uppercase in 16 bit blocks separated by whitespaces with an additional whitespace after 5 blocks (cf. Figure 1b), e. g.,

```
73EE 2314 F65F A92E C239  0D3A 718C 0701 0001 2282
```

Other implementations, such as OpenKeychain, deviate only slightly, for example by displaying them lowercase or with colored letters to ease comparison but still provide compatibility with GnuPG.

SSH fingerprint strings, as defined in RFC 4716 and RFC 4253 [15, 44], are calculated by

$$\mathit{Hex}(\mathit{MD5}(\mathit{Base}64(\mathit{algo} \,\|\, \mathit{pk})))$$

where *algo* is a string indicating the algorithm, for example “ssh-rsa”. Fingerprints are displayed as “hexadecimal with lowercase letters and separated by colons” [15] (cf. Figure 1a), e. g.,

```
6f:85:66:da:e3:7a:02:c6:5e:62:3f:36:b7:d9:b4:2c
```

3.2.2 Base32

Base32 uses the Latin alphabet (A-Z) without the letters O and I (due to the confusion with numbers 1 and 0). There is no difference between lower-case letters and upper-case letters. In addition, a special padding character “=” is used, since the conversion algorithm processes blocks of 40 bit (5 Byte) in size. The source string is padded with zeroes to achieve a compatible length and sections containing only zeroes are represented by “=” [20, 21].

The ZRTP key exchange scheme for real-time applications is based on a Diffie-Hellman key exchange extended by a preceding hash commitment that allows for very short fingerprints, called Short Authentication

Strings (SAS) without compromising security [45]. The Base32 encoding used in ZRTP uses a special alphabet to produce strings that are easier to read out loud. VoIP applications such as CSipSimple[4] use this Base32 option, usually named "B32" inside the protocol. Here, the leftmost 20 bits of the 32 bit SAS value are encoded as Base32. , e. g.,

```
5 e m g
```

3.2.3 Base64

There exist a number of specifications for encoding data into the Base64 format, which uses the Latin alphabet in both lower-case and upper-case (a-z, A-Z) as well as the digits 0-9 and the characters "+", "/", and "=" to represent text data. Again, the character "=" is used to encode padded input [20]. Starting with OpenSSH 6.8 a new fingerprint format has been introduced that uses SHA-256 instead of MD5 and Base64 instead of hexadecimal representation. In addition the utilized hash algorithm is prepended, e. g.,

```
SHA256:mVPwvezndPv/ARoIadVY98vAC0g+P/5633yTC4d/wXE
```

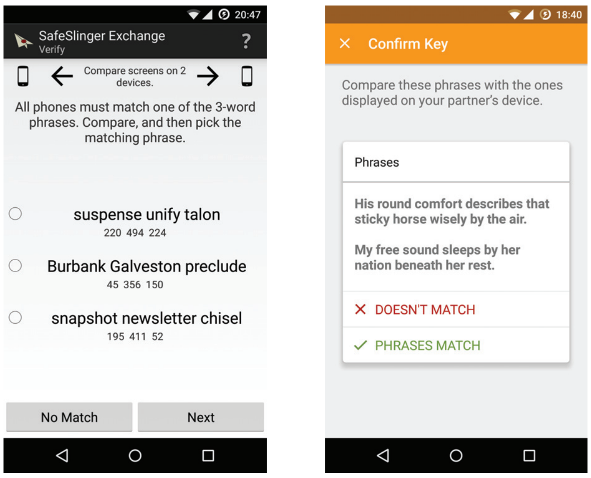
3.3 Unrelated Words

Instead of (alpha)numeric representation, fingerprints can be mapped to lists of words. Here, the binary representation is split into chunks, where each possible value of a chunk is assigned to a word in a dictionary. To increase readability, such a dictionary usually contains no pronouns, articles, prepositions and such. Word lists, such as the PGP Word List [22] and the Basic English word list compiled by K.C. Ogden [31], are primarily used for verification mechanisms based on SAS. Key-Fingerprints represented by words have been implemented for VoIP applications based on the ZRTP key exchange and other real-time communication protocols. Examples are Signal[5], and the messaging and contact sharing application SafeSlinger [14] (cf. Figure 2). Besides their use in SAS based mechanisms, miniLock and Peerio utilize unrelated words for passphrase generation.

An example for a modern VoIP implementation that utilizes ZRTP for key exchange over Secure Real-Time Transport Protocol (SRTP) is Signal's private calling feature, previously distributed as Redphone. The developers chose to implement only a specific subset of the ZRTP specification [45], namely Diffie-Hellmann key exchange via P-256 elliptic curves using "B256" SASs, i. e., Base256 encoding that maps to the leftmost 16 bits of the 32 bit SAS values to the previously introduced PGP Word List [22], e. g.,

[4]https://github.com/r3gis3r/CSipSimple

[5]https://github.com/WhisperSystems/Signal-Android



All phones must match one of the 3-word phrases. Compare, and then pick the matching phrase.

suspense unify talon
220 494 224

Burbank Galveston preclude
45 356 150

snapshot newsletter chisel
195 411 52

Compare these phrases with the ones displayed on your partner's device.

Phrases

His round comfort describes that sticky horse wisely by the air.

My free sound sleeps by her nation beneath her rest.

(a) SafeSlinger: List of words (b) OpenKeychain: Sentences

Figure 2: Language-based fingerprint representations

```
quota holiness
```

The messaging application SafeSlinger is based on a Group Diffie-Hellman protocol [14] implementing a key verification with SASs for up to 10 participants. In SafeSlinger the leftmost 24 bits of a SHA-1 hash is used to select 3 words from the PGP Word List, e. g.,

```
suspense unify talon.
```

Besides this, two other 3 word triples are selected to force users to make a selection before proceeding (cf. Figure 2a).

In contrast to Signal and SafeSlinger, Peerio (based on miniLock) does not use any SAS based verification mechanism. It uses pictures for verification and word lists for code generation. The word list is generated from most occurring words in movie subtitles. Besides key verification, these are also used to generate so called passphrases, which are used to derive their ECC private keys.

3.4 Generated Sentences

The words from the previous dictionaries can also be used to generate syntactically correct sentences as proposed by previous research: Goodrich et al. proposed to use a "syntactically-correct English-like sentence" representation for exchanging hash-derived fingerprints over audio by using *text-to-speech* (TTS) [16]. Michael Rogers et al. implemented a simple deterministic sentence generator [16, 38][6] Though the sentences from both approaches rarely make sense in a semantic fashion, they are syntactically correct and are claimed to pro-

[6]https://github.com/akwizgran/basic-english

vide good memorability. In our study, we used Michael Roger's approach for our sentence generator.

We implemented this method for PGP fingerprints in OpenKeychain 3.6[7] (cf. Figure 2b). To the best of the authors' knowledge, to this date, it is the first integration of key verification via sentences although other projects are considering to change their fingerprint encoding scheme [38, 36].

4 Methodology

In order to evaluate the effect and perception of the different textual key-fingerprint representations, we conducted an online study on Amazon's Mechanical Turk (MTurk) crowdsourcing service. Our Universities do not have an IRB, but the study conformed to the strict data protection law of Germany and informed consent was gathered from all participants. Our online study is divided into two parts: The experiment for performance evaluation followed by a survey extracting self-reported data from users. The survey ended with demographic questions.

4.1 Security Assumptions

In this section, we define the underlying security assumptions of our study, such as fingerprint method, length, and strength against an adversary. The fingerprint method and parameters are utilized consistently for all experiments in our study to offer comparability between all possible fingerprint representations. This attack model is important for the usability since an unrealistically strong or weak attacker could skew the results. Obviously, if the fingerprint strength is not kept equal between the systems this would also skew the results.

4.1.1 Fingerprint Method

To decide upon a fingerprint method for humanly verifiable fingerprints in our study, we first have to differentiate between human and machine verification to illustrate their differences. While a full fingerprint comparison can be implemented for machine verification, humans can fall for fingerprints that match only partially. Additionally, machine comparison can work with long values, whereas for human verification the length must be kept short enough to fit on business cards and to keep the time needed for comparison low.

For machine comparison, full SHA-256 hashes should be calculated binding a unique *ID* to the public key material. The probability of finding a preimage or collision attack is obviously negligible, but the fingerprints can still be computed fast in an ad-hoc manner when needed.

[7]`https://www.openkeychain.org`

It is important to note that collision resistance is not required for our scenarios. It is required for infrastructure-based trust models such as X.509, where certificates are verified by machines and trust is established by authority. In these schemes, a signature generated by a trusted authority can be requested for a certificate by proving the control over a domain, but then reused maliciously for a different certificate/domain. This is already possible with a collision attack, without targeting a full preimage. In contrast, the direct human-based trust schemes considered in this study only need to be protected against preimage attacks, because no inherently trusted authority is involved here.

While machine comparison needs to be done fast, e. g., on key import, manual fingerprint verification by humans is done asynchronously in person or via voice. Thus, we can use a key derivation function to provide a proof-of-work, effectively trading calculation time for a shorter fingerprint length. Secure messaging applications such as Signal or OpenPGP-based ones could pre-calculate the fingerprints after import and cache these before displaying them for verification later.

Thus, modern memory-hard key derivation functions such as *scrypt* [34] or *Argon2* [3] can be utilized to shorten the fingerprint length. These key derivation functions are parametrized to allow for different work factors. Suitable parameters need to be chosen by implementations based on their targeted devices and protocol.

As discussed in Section 3.2.1, while the generation of new fingerprints consists of the creation of a new key pair and the key derivation step, an attacker can potentially skip the key creation. Thus, in the following we only consider the key derivation performance as the limiting factor for brute force attacks.

When utilizing a properly parametrized key derivation function for bit stretching, the security of a 112 bit long fingerprint can be increased to require a brute force attack comparable to a classical 2^{128} brute force attacker. Consequently, a fingerprint length of 112 bit is assumed throughout our study.

4.1.2 Attacker Strength for Partial Preimages

In our user study, we assume an average attacker trying to impersonate an existing *ID* using our fingerprint method. Thus, an attacker would need to find a 112 bit preimage for this existing fingerprint using a brute force search executing the deployed key derivation function in each step. Due to the work factor, we consider this to be infeasible and instead concentrate on partial preimages. For comparability and to narrow the scope of our study, an attacker is assumed that can control up to 80 bits of the full 112 bit fingerprint.

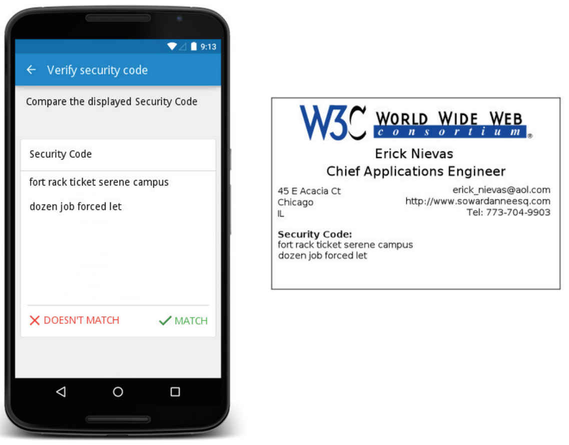
Attackers might aim to find partial preimages where the uncontrolled bits occur at positions that are more easily missed by inattentive users. First, the bits at the beginning and the end should be fixed as users often begin their comparison with these bits. Thus, we assume that, for any representation method, the first 24 and last 24 bits are controlled by the attacker and thus the same as in the existing fingerprint. Based on the feedback from our pre-study participants and reports from related work, this can be considered best-practice [17, 37]. Second, of the remaining 64 bits in the middle of our 112 bit fingerprint, we assume that 32 bits are controlled by the attacker in addition to the first 24 and last 24 bits. In total, we assume that 80 bits are controlled by the attacker, i. e., are the same as in the existing fingerprint, and 32 bit are uncontrolled.

The probability of finding such a partial preimage for a fingerprint when executing 2^{49} brute force steps is calculated approximately by

$$1-\left(\frac{2^{112}-\sum_{k=1}^{32}\binom{64}{k}}{2^{112}}\right)^{2^{49}}\approx 0.66.$$

The inner parentheses of this equation define the probability that no partial preimage exists for one specific bit permutation. Instead of using $\binom{64}{32}$, a sum over 32 variations has been inserted to include permutations with more than the uncontrolled 32 bit that are also valid partial preimages. Finally, the probability to find a partial preimage is defined by the inverse of the exponentiation. Assuming the scrypt key derivation function parametrized with $(N,r,p)=(2^{20},8,1)$, Percival calculates the computational costs of a brute force attack against 2^{38} ($\approx 26^8$) hashed passwords with \$610k and 2^{53} ($\approx 95^8$) with \$16B [34]. These costs can be considered a lower and upper bound for our attacker, which we assume to have average capabilities and resources. While 2^{38} has a probability of finding a partial preimage of only 0.05%, with 2^{42} the probability reaches nearly 1%, and with 2^{49}, as in our example, a partial preimage is found with over 50%.

In our study, we simulate attacks by inverting the bits from the existing fingerprint which are uncontrolled by the attacker, while the controlled bits are unchanged. For our theoretical approximation, we assume that the first 24 and last 24 bits should be controlled as well as 32 bits from the middle. In our study, we simulate an even more careful selection of appropriate fingerprints from the ones that an attacker would brute force. A general criteria here is to minimize the influence of uncontrolled bits on the entire fingerprint: For numeric and alphanumeric representations all bits affecting a character or digit are inverted together. For unrelated words, all bits affecting a word are changed. Sentences are never changed in a way that would alter the sentence structure.


9:13
Verify security code
Compare the displayed Security Code
Security Code
DOESN'T MATCH
MATCH
W3C WORLD WIDE WEB consortium

fort rack ticket serene campus
dozen job forced let

Erick Nievas
Chief Applications Engineer
45 E Acacia Ct
Chicago
IL
erick_nievas@aol.com
http://www.sowardanneesq.com
Tel: 773-704-9903

Security Code:
fort rack ticket serene campus
dozen job forced let

Figure 3: A screenshot of the actual task a user had to perform in the experiment. A user rates whether the *security codes* match, in this case with the Peerio word list approach, by clicking on the corresponding buttons shown on the phone.

4.2 Pre-Study

To get additional feedback from participants and evaluate our study design for flaws and misunderstandings, we conducted two small pre-studies: A lab study with 15 participants and an MTurk experiment with 200 participants, all required to perform 10 comparisons for each representation scheme (totally 60 comparisons in a randomized order). In our lab-study, we mainly focused on qualitative feedback, whereas the main goal of the MTurk pre-study was to find flaws in the presentation and task descriptions, as well as to check whether our proposed methodology is received as expected.

The biggest problem we found regarding the study design was that participants were uncertain if they should check for spelling mistakes in the words and sentence-based representation or if the all attacks would change entire words. To clarify this, a speech bubble was included in the task description that the participants do not have to look for spelling mistakes for language-based approaches.

We tested different rates of attack during the pre-study. The results showed that participants who were exposed to frequently occurring attacks were more aware and had a much higher attack detection rate. For our main study, we reduced the number of attacks to 40 comparisons with 4 attacks to have a good balance between true positives and false negatives. We received feedback that attacks on anchor parts of the strings, i. e., in the beginning, end, and at line breaks could be easily detected. Many users had problems with distinguishing the hexadecimal from the Base32 representation as well as distinguishing different word list approaches (Peerio vs. OpenPGP word list). Thus, we opted for a mixed factorial study design where users test *only one* scheme of each type. We grouped the hexadecimal and Base32 scheme for the alphanumeric type and the PGP and Peerio for the word-list type together. These two groups were tested between-subjects in a split-plot design, i. e., the participants test either hexadecimal or Base32 for the alphanumeric type. See Table 2 for a graphical representation of our condition assignment design.

4.3 Experiment Design

The main part of our online study is the experiment part where users perform actual fingerprint comparisons. Here, we conducted two separate experiments with a distinct set of participants: (1) our main experiment testing different textual high-level representation schemes against each other and (2) a secondary experiment testing different chunk sizes for the hexadecimal representation. We opted for two distinct experiments due to the exponential growth of experiment conditions, as described in Section 4.3.1.

Before letting the participants start our experiment, we explained the scenario:

> "With this HIT, we are conducting an academic usability study followed by a short survey about different types of security codes used in the IT world. Security codes are often used in encrypted communications to identify the participants in a communication. If the security codes match, you are communicating securely. If they don't match, an eavesdropper may be intercepting your communication".

On MTurk, the term *Human Intelligence Task*, or *HIT* stands for a self-contained task that a worker can work on, submit answers, and get a reward for completing. Since our participants might not be familiar with the *key-fingerprint representation* term, we replaced it with *security codes* for the sake of the study.

We opted not to obfuscate the goal of the study since our research aims at finding the best possible representation for the comparison of key-fingerprints in a security context. This is closest to how users interact with fingerprints in the real world — their secure messaging applications also ask them to compare the strings for security purposes. The question how to motivate users to compare fingerprints is an entirely different research question. So in our case, we believe it was not necessary or desirable to use deception and since deception should be used as sparingly as possible we opted for the "honest" approach.

After agreeing the terms, participants are shown a fictitious business card next to a mobile phone, both displaying a security code (as shown in Figure 3). To become more familiar with the task, the experiment is

Type (Within-Group)	**Scheme (Between-Group)**
Alphanumeric	Hexadecimal `XOR` Base32
Numeric	Numeric
Unrelated Words	PGP `XOR` Peerio
Generated Sentences	Generated Sentences

Table 2: To avoid confusion between too similar approaches (cf. Section 4.2), in our condition assignment, scheme types (left column) can consist of multiple representation schemes (right column). Each participant tests *only one* randomly assigned scheme of each type in a randomized order. .

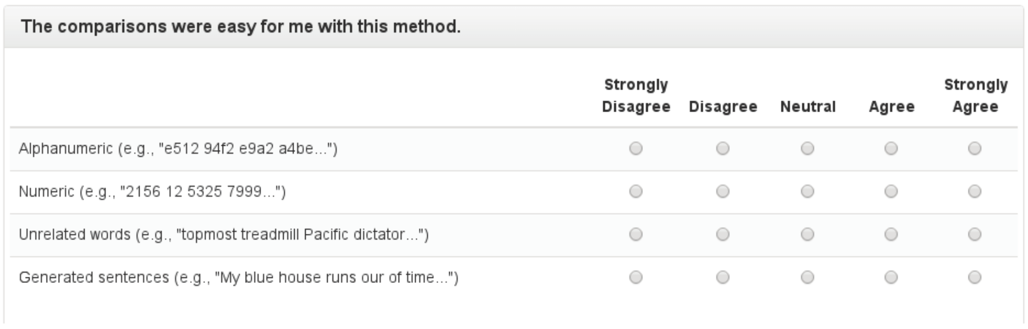
started with 4 training tasks (each method once) not considered in the evaluation. The user's only task is to rate whether the shown fingerprints match by clicking on *Match* or *Doesn't Match* on the phone. Based on the condition assignment, participants see different approaches in a *randomized order*. We measure whether their answer was correct and their speed, i. e., the amount of time spent on the comparison. The experiment is concluded with a survey collecting feedback on the used approaches and the tasks and demographic information discussed in the "Results" section.

4.3.1 Variables and Conditions

In the main experiment, the used *representation scheme* is our controlled independent variable whereas its values define our experiment conditions. In our additional chunking experiment, the *chunking size* is our controlled independent variable instead of the representation algorithm. During all tasks, we measure how fast participants perform with their given conditions and whether they are able to detect attacks by rating "incorrect" (*speed* and *accuracy* as our measured dependent variables).

In both experiments, each user had to perform 46 comparisons in total. To detect users clicking randomly, 2 obviously distinct comparisons were added to test a participant's attention. Training comparisons and attention tests are not included in the evaluation. Based on the feedback in our pre-study, we added tooltips during the training comparisons giving hints for language-based approaches telling the user that spelling attacks would not occur. We set the number of attacks to six: two obvious attacks where all bits are altered serving as control questions and 4 actual attacks with partial 80-bit preimages (one for each representation scheme). Participants failing at the control attacks are not considered in the evaluation but still received a payment if finishing all tasks. The major challenge in the study design is a high attack detection rate in general: most users perform comparisons correctly for the given attacker strength.

To avoid side effects, we chose fixed font size, color



The comparisons were easy for me with this method.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Alphanumeric (e.g., "e512 94f2 e9a2 a4be...")	○	○	○	○	○
Numeric (e.g., "2156 12 5325 7999...")	○	○	○	○	○
Unrelated words (e.g., "topmost treadmill Pacific dictator...")	○	○	○	○	○
Generated sentences (e.g., "My blue house runs our of time...")	○	○	○	○	○

Figure 4: A screenshot showing a statement rating in the post-experiment survey. Since the participants might not distinguish the different types, we have provided an example from their previous task.

and style, i. e., the same typeface for all fingerprint representations. In addition, we set fixed line breaks for sentences and word lists. In the main experiment, the same chunking style was used for all representations: For (alpha)numeric approaches a chunk consists of four characters separated by spaces. For word lists, we opted for a line break every four words. In the generated sentences representation, one sentence per line is displayed. We are aware that all these design decisions can have an effect on the comparison of the representations. However, our pre-study results show a significantly lower effect size. More importantly, we are mainly interested in comparing the concepts, therefore we did not vary any of the visual attributes like font size or style. In particular, differences resulting from the font's typeface have not been evaluated. Lund showed in his meta-analysis that there are no significant legibility differences between serif and sans serif typefaces [25].

Chunk-Size Testing A question was raised whether the chunking of a hexadecimal string plays a greater role in comparison to the different approaches. Thus, in addition to the main experiment testing different representation types, we conducted a second experiment with new participants testing different chunk sizes for the hexadecimal representation. Here, we used chunk-sizes ranging from 2 to 8 in addition to "zero-chunk size" (8 cases). The zero-chunk size means that no spaces have been included. To make the results more comparable, we opted for a similar design as done in the major experiment, i. e., we required the same amount of comparisons, used the same font settings, and had the same amount of attacks. For each participant, we assigned 4 out of 8 different chunk-sized randomly. Same as in the major experiment, all participants had to compare 46 fingerprints whereas the first 4 are considered as training comparisons, 4 attacks (one for each chunk size), and 2 control attacks with obviously distinct fingerprints.

The major experiment is followed by a survey focusing on self-reported user perception and opinions about the different approaches. This is the main reason we opted to compare as much as possible in a within-groups fashion and only selected a small number of conditions in total. Since users might not notice the difference between the various dictionary or alphabet approaches, we designed a mixed factorial design where the users would only get one of the alphabets/dictionaries (between-subjects) but they would test all different high-level systems (within-group) as depicted in Table 2. The between-group conditions have been assigned randomly with a uniform distribution. Since participants from our pre-study had difficulties to distinguish the different chunking approaches, we skipped the survey part in the chunk-size experiment.

4.3.2 Online Survey

The experiment was followed by an online survey gathering self-reported data and demographics from participants. To measure perception, we asked the participants whether they agreed with statements discussed in subsection 5.2 on a 5 point Likert scale: from strongly disagree to neural strongly agree as shown in Figure 4. Participants had to rate each representation type for all statements. Since users might not distinguish the different representation schemes, we provide an example from their previously finished task.

4.3.3 Statistical Testing

We opted for the common significance level of $\alpha = 0.05$. To counteract the multiple comparisons problem, we use the Holm-Bonferronicorrection for our statistical significance tests [18]. Consequently, all our p-values are reported in the corrected version.

We test the comparison duration with the Mann-Whitney-Wilcoxon (MWW) test (two-tailed). We opt for this significance test due to a few outliers, consequently a

Scheme	Speed				Accuracy			Total		
	mean [s]	med [s]	stdev	p-val	fail-rate	p-val	f-pos	fails	attacks	tests
Hexadecimal	11.2	10.0	6.4		10.44		0.49	50	479	4765
Hexadecimal – Base32	1.0	1.1	0.0	<0.001	−1.94	0.690	−2.09	12	32	269
Hexadecimal – Numeric	0.6	0.5	0.6	<0.001	−4.10	0.048	0.21	−9	−452	−4527
Hexadecimal – PGP	−1.8	−1.2	−1.0	<0.001	−1.65	0.690	−0.01	11	35	340
Hexadecimal – Peerio	2.5	2.7	0.8	<0.001	−4.69	0.048	0.08	22	−8	−91
Hexadecimal – Sentences	−1.1	−0.7	−0.6	<0.001	−7.45	<0.001	−0.99	22	−457	−4518
Base32	10.2	8.9	6.4		8.50		2.58	38	447	4496
Base32 – Hexadecimal	−1.0	−1.1	−0.0	<0.001	1.94	0.690	2.09	−12	−32	−269
Base32 – Numeric	−0.4	−0.6	0.6	<0.001	−2.16	0.404	2.30	−21	−484	−4796
Base32 – PGP	−2.8	−2.3	−1.0	<0.001	0.28	0.714	2.08	−1	3	71
Base32 – Peerio	1.5	1.6	0.8	<0.001	−2.75	0.404	2.17	10	−40	−360
Base32 – Sentences	−2.1	−1.8	−0.6	<0.001	−5.51	<0.001	1.10	10	−489	−4787
Numeric	10.6	9.5	5.8		6.34		0.28	59	931	9292
Numeric – Hexadecimal	−0.6	−0.5	−0.6	<0.001	4.10	0.048	−0.21	9	452	4527
Numeric – Base32	0.4	0.6	−0.6	<0.001	2.16	0.404	−2.30	21	484	4796
Numeric – PGP	−2.4	−1.7	−1.6	<0.001	2.45	0.404	−0.22	20	487	4867
Numeric – Peerio	1.9	2.2	0.2	<0.001	−0.59	0.714	−0.13	31	444	4436
Numeric – Sentences	−1.7	−1.2	−1.2	<0.001	−3.35	0.004	−1.20	31	−5	9
PGP	13.0	11.2	7.4		8.78		0.50	39	444	4425
PGP – Hexadecimal	1.8	1.2	1.0	<0.001	1.65	0.690	0.01	−11	−35	−340
PGP – Base32	2.8	2.3	1.0	<0.001	−0.28	0.714	−2.08	1	−3	−71
PGP – Numeric	2.4	1.7	1.6	<0.001	−2.45	0.404	0.22	−20	−487	−4867
PGP – Peerio	4.3	3.9	1.8	<0.001	−3.03	0.337	0.09	11	−43	−431
PGP – Sentences	0.7	0.5	0.4	<0.001	−5.79	<0.001	−0.98	11	−492	−4858
Peerio	8.7	7.3	5.6		5.75		0.41	28	487	4856
Peerio – Hexadecimal	−2.5	−2.7	−0.8	<0.001	4.69	0.048	−0.08	−22	8	91
Peerio – Base32	−1.5	−1.6	−0.8	<0.001	2.75	0.404	−2.17	−10	40	360
Peerio – Numeric	−1.9	−2.2	−0.2	<0.001	0.59	0.714	0.13	−31	−444	−4436
Peerio – PGP	−4.3	−3.9	−1.8	<0.001	3.03	0.337	−0.09	−11	43	431
Peerio – Sentences	−3.6	−3.4	−1.4	<0.001	−2.76	0.075	−1.07	0	−449	−4427
Sentences	12.3	10.7	7.0		2.99		1.48	28	936	9283
Sentences – Hexadecimal	1.1	0.7	0.6	<0.001	7.45	<0.001	0.99	−22	457	4518
Sentences – Base32	2.1	1.8	0.6	<0.001	5.51	<0.001	−1.10	−10	489	4787
Sentences – Numeric	1.7	1.2	1.2	<0.001	3.35	0.004	1.20	−31	5	−9
Sentences – PGP	−0.7	−0.5	−0.4	<0.001	5.79	<0.001	0.98	−11	492	4858
Sentences – Peerio	3.6	3.4	1.4	<0.001	2.76	0.075	1.07	0	449	4427

Table 3: Our experiment results showing the differences between the representation schemes. The top rows of each row group separated by a rule, show the raw performance of a baseline scheme, followed by italic rows showing a direct comparison delta. Greyed-out values are not backed by statistical significance. The columns *fail-rate* (undetected attacks) and *false-pos* (same string rated as an attack) display percentage values.

slightly skewed normal distribution, and a large amount of collected data. The *common language effect size* is shown by mean and median comparisons [26].

The attack detection rate is tested with a pairwise Holm-Bonferroni-corrected Barnard's exact test (`Exakt` package in `R`) achieving one of highest statistical power for 2x2 contingency tables [2].

Survey ratings are, again, tested by using the MWW significance test (two-tailed test). As has been shown in previous research [9], it is most suitable for 5-point Likert scales, especially if not multimodal distributed as in our survey results. In case two fingerprint representation schemes are statistically tested against each other, only participants encountering both schemes were considered.

5 Results

In this section, we present our results: our online study with 1047 participants has been conducted in August and September 2015. The study for testing the chunk size has been conducted in February 2016 with 400 participants. Starting with our online experiment evaluation showing the raw performance of users, we then present user perception results from the follow-up survey. Finally, we discuss the demographics of our participants.

5.1 Online Experiment

Participants who have not finished all comparisons or failed the attention tests were excluded from our eval-

Scheme	Speed			Accuracy			Total		
	mean [s]	med [s]	p-val	fail-rate	p-val	false-pos	fails	*attacks*	*tests*
Hexadecimal (4)	12.3	10.4		6.78		0.38	16	236	2360
hex (4) – hex (0)	−2.4	−2.6	<0.001	0.33	1.000	−0.28	−2	−17	−170
hex (4) – hex (2)	−0.3	−0.9	<0.001	1.37	1.000	0.00	−3	3	30
hex (4) – hex (3)	−0.3	0.1	0.362	−0.64	1.000	0.09	2	8	80
hex (4) – hex (5)	−1.4	−1.2	<0.001	1.01	1.000	−0.40	−2	5	50
hex (4) – hex (6)	−1.9	−1.8	<0.001	2.43	1.000	0.09	−5	8	80
hex (4) – hex (7)	−1.7	−1.8	<0.001	3.35	1.000	0.19	−8	−1	−10
hex (4) – hex (8)	−2.8	−3.2	<0.001	1.35	1.000	−0.12	−4	−10	−100

Table 4: Comparison of the chunking experiment results showing the differences between the representation schemes. The top row shows the raw performance of the hexadecimal scheme with a four-character chunking, followed by italic rows showing a direct comparison delta. Greyed-out values are not backed by statistical significance. The columns *fail-rate* (undetected attacks) and *false-pos* (same string rated as an attack) display percentage values.

uation: all participant compared 46 security codes in a randomized order, whereas 40 (10 of each scheme) were considered in the evaluation. The four training samples and the control questions are excluded. Few comparisons done in less than 2 seconds and more than one minute have been excluded. The reason for such can either be multiple clicks during the page load, or external interruptions of the participants. None of the attack could be successfully detected in under 4 seconds.

Our experiment results, summarized in Table 3, show the raw performance of all schemes regarding their speed, accuracy and false-positive rate. The top rows of each row group, separated by a rule, show the raw performance of a representation scheme as baseline (negative values indicate lower values than the baseline). The following rows show a direct comparison delta between between two schemes. The speed column group consists of the mean and median (in seconds), the standard deviation and the according p-values for a direct comparison. The fail-rate column shows the rate of the undetected attacks with the according p-values for a direct comparison. The total column group simply shows the total numbers of tests, attacks and undetected attacks.

The results show that the average time spent on comparisons plays only a minor role among the schemes: 4.3 s difference between the best and the worst scheme. Note that the Peerio word-list scheme performed best with 8.7 s mean whereas the PGP word list performed worst with 13 s mean ($p < 0.001$).

However, there is a clear effect regarding the attack detection rate (see Table 3). All alternative key-fingerprint representations performed better than the state-of-the-art hexadecimal representation, where 10.1% of attacks have not been detected by the users. Previous work shows similar numbers for Base32 [19]. To our surprise, the numeric approach performs better in both categories: it features an attack detection rate of 93.57% ($p < 0.01$) and an average speed of 10.6s ($p < 0.001$). Generated sentences achieved the highest attack detection rate of 97.97% with a similar average speed as the hexadecimal scheme. On the downside, this scheme has produced a slightly higher false-positive rate. We found that the false positives occurred mostly with longer sentences where there has been a line break on the phone mock-up due to portrait orientation. This is a realistic problem of this system if used with portrait orientation and not a problem with our mock-up in itself. Improvements on making the sentences shorter could mitigate this situation.

Chunk-Size Experiment

Table 4 summarizes the results of our secondary chunk-size experiment. As can be seen, no statistically significant results have been achieved for the *attack detection fail-rate* (undetected attacks by end users). However, we observed that the chunk sizes with 3 and 4 characters performed best in speed, even though the effect sizes were minor: only 3.3 seconds difference with similar standard deviations between the best and worst chunk size setting.

Firstly, we notice that despite the same attack strength as in our major experiment, participants were able to detect more attacks. We suspect that the higher attack detection rate is based on (1) a higher learning effect due to the same scheme for all comparisons and (2) in contrast to our major study, participants had a slightly higher drop-out rate and thus only more motivated participants were considered. This is supported by the numbers in the total tests column of Table 4: here, we can see that for the zero-chunking and chunking with 8 characters less tests have been performed. This is based on the fact that although the chunk sizes have been assigned almost uniformly, participants assigned with harder chunk settings often dropped out before even finishing their entire task.

More importantly, our results also support the claim from our pre-study: The chunking parameter in hexadecimal strings plays only a minor role in the *attack detection fail-rate*.

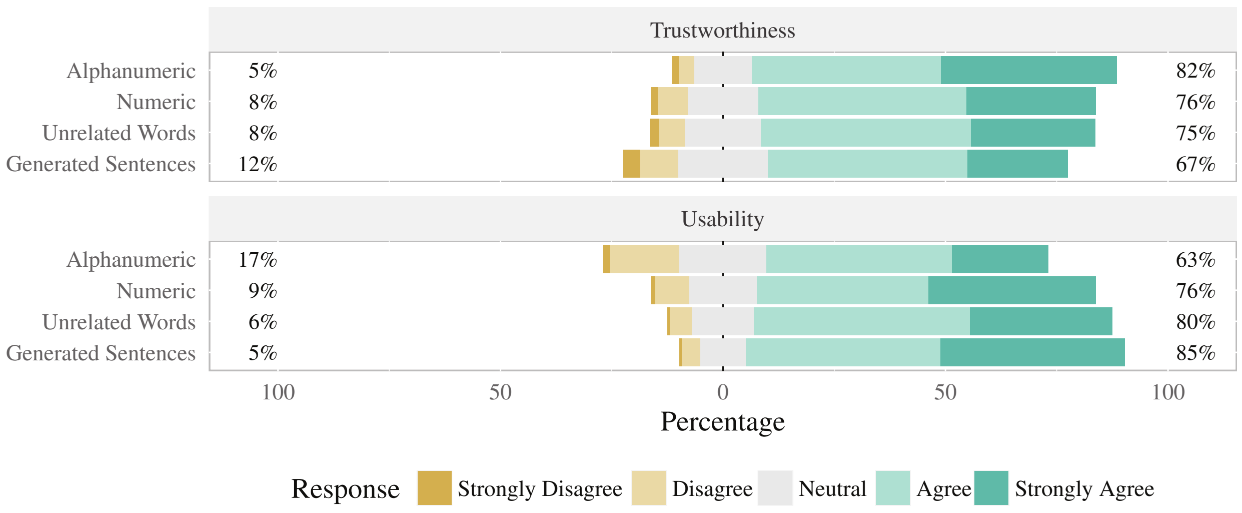


Figure 5: Aggregated survey results for statement rating regarding the usability and trustworthiness.

5.2 Online Survey

To measure the usability and trustworthiness of all representation schemes, we asked our participants whether they agreed with the following statements:

S_1 The comparisons were easy for me with this method

S_2 I am confident that I can make comparisons using this method without making mistakes

S_3 I think making comparisons using this method would help me keep my communications secure

S_4 I was able to do the comparisons very quickly with this method

S_5 I found this method difficult to use

S_6 Overall, I liked this method

We mixed positive and negative statements, e. g., S_1 and S_5, to create a more robust measure. S_6 is used to calculate the overall ranking of the different representation schemes.

Figure 5 shows the aggregated results where the usability statements are grouped to one usability feature and the trustworthiness derived from the rating on the statement S_3. Negative statement ratings have been inverted for a better comparison. Figure 6 shows the rating results for each specific statement in the survey. The order of the tested schemes has been chosen randomly, but was kept consistent across all statements. Same as in our online experiment evaluation, the pairwise statistical tests are Holm-Bonferroni corrected. In case of a direct statistical test between two schemes, only users encountering both schemes have been considered. All in all, the usability perception of the participants is almost consistent with the performance results from the experiment.

To measure the perception of the task difficulty, we asked the participants whether they agreed with the statements S_1, S_2 and S_3 respectively. As illustrated in Figure 6 in the Appendix A, the effect size between the different approaches is low. However, the participants were more likely to agree that language-based representation schemes are easier to use. For instance, we see that in comparison to the alphanumeric schemes (average rating of 3.4), word list (average rating of 3.9, $p < 0.001$) and generated sentence schemes (average rating of 4.2, $p < 0.001$) are rated to be easier by our participants (S_1, S_5). While the experiment results of the sentence generators clearly outperformed all other approaches, they also were rated better by the participants. Same applies for the low-performing hexadecimal and Base32 schemes which clearly received lower ratings. Consistently with the surprising performance results in the experiment, the numeric scheme is also considered to be easier by many participants: average rating of 3.9 and $p < 0.001$.

The sentence generator scheme achieved the highest user confidence rating "making comparisons without any mistakes" (S_2, $p < 0.001$ for all pairwise comparisons). The participants' perception is consistent with the experiment results where the word-list-based and sentence generator schemes lead to higher attack detection rates.

The ratings for S_4 illustrate that more complex representation schemes from the user's point of view, such as hexadecimal and Base32, are considered to be more secure by participants, even though all approaches provide the same level of security.

5.3 Demographics

A total of 1047 users participated in the online study while only 1001 have been considered in the evaluation due to our two control questions. Out of the evaluated participants, 534 participants were male, 453 were female, 4 chose other while the rest opted to not give any information. No significant difference between genders could be found, with a subtle trend of a higher accuracy for women and higher speed among men. The median age was 34 (34.4 average) years, while 34 participants chose not to answer (no statistically significant differences between ages).

A total of 39 people reported to have "medical conditions that complicated the security code comparisons (e. g., reading disorders, ADHD, visual impairments, etc.)" with a slightly higher undetected attack rate (statistically insignificant due to small sample size and thus low statistical power).

The majority of the participants stated to have a Bachelor's degree (399 of 1047) as their highest education whereas 34% chose not to answer. 931 participants have started our HIT but stopped early during the experiment (mostly after the first few comparisons). 160 users reported the general task to be annoying.

6 Discussion

The results of our study show that while there are subtle speed variations among all approaches, the attack detection rate and user perception for the current state-of-the-art hexadecimal key-fingerprint representation is significantly lower than those of most alternative representation schemes. Language-based representations (with the exception of the PGP word list) show improved user behaviour leading to a higher detection rate of attacks. To improve the usability of key-fingerprints, we propose the following takeaways based on our study results.

6.1 Takeaways

Our results show that all representation schemes achieve a high accuracy (high attack detection rate) and can be performed quickly by users. As expected, language-based fingerprint representations are more resilient against attacks (higher attack detection rate) and achieve better usability scores. Among all conditions, alphanumeric approaches performed worse and have been outperformed. For instance, the *numeric* representation was more suitable than hexadecimal and Base32. The raw performance results suggest a similar speed for the numeric representation with a higher attack detection rate, and it also has received better usability ratings from end-users.

Our chunking experiment has shown that chunk-sizes play only a minor role in improving attack detection rates (we could not find statistically significant differences). However, if a hexadecimal representation is used chunks of 3 and 4 characters perform best.

As shown by the word list representations, the comparison speed can be increased by larger dictionaries leaving room for improvement in this area. Even though all representation schemes provide the same level of security, exotic looking solutions are considered to be more secure by end users.

6.2 Limitations

Most importantly, our study design *does not test* whether end users *are actually willing to compare any fingerprints* in practice. We only aim to study how easy different representations are to compare from the users' point of view.

As with any user study conducted with MTurk, there is concern about the external validity of the results: users in the real world might show different behaviour. This is mainly because of two reasons: (1) in practice fingerprint comparisons will seldom occur in a such frequency, and (2) when performed in practice play a more important role than just participating in an anonymous online study. Additionally, MTurkers have been shown to be more tech-savvy and are better in solving textual and visual tasks in comparison to the average population. Thus, they could have performed better in most of the comparison conditions than the average population. It is also known that some MTurkers just "click through" studies to get the fee and thus distort study results. Our counterbalanced study design with included control questions and statistical significance tests mitigate this effect. For instance, we excluded 46 out of 1047 participants from our main study part based on these questions being answered incorrectly.

Due to the within-group part of our factorial design, many parameter choices such as different fonts, font sizes, attack rates, etc. could not be considered. These are, however, interesting avenues for future work. As shown in our additional chunking experiment, another challenge in testing different parameters is the high attack detection rate, where subtle changes would require a high amount of users to produce statistically significant results.

Due to the anonymous nature of online studies, it is also impossible to reliably tell which languages a participant is fluent in. We specified that we only wanted participants from English-speaking countries, however we had no way of checking compliance except by relying on self-reported data. Language-based representation approaches might induce additional barriers for non-native speakers, e. g., due to unknown or unfamiliar words.

7 Conclusion and Future Work

We evaluated six different key-fingerprint representation types with regards to their comparison speed, attack detection accuracy and usability, which encompasses attack detection but also resilience against human errors in short-term memory. An online study with 1047 participants was conducted to compare numeric, alphanumeric (Hexadecimal and Base32), word lists (PGP and Peerio), as well as generated sentences representation schemes for key-fingerprint verification. All fingerprint representations were configured to offer the same level of security with the same attacker strength.

Our results show that usage of the large word lists (as used in Peerio) lead to the fastest comparison performance, while generated sentences achieved highest attack detection rates. In addition, we found that additional parameters such as chunking of characters plays only a minor role in the overall performance. The widely-used hexadecimal representation scheme performed worst in all tested categories which indicates that it should be replaced by more usable schemes. Unlike proposals which call for radically new fingerprint representations, we studied only textual fingerprint representations, which means that the results of our work can be directly applied to various encryption applications with minimal changes needed. Specifically, no new hardware or complex software is required: applications merely need to replace the strings they output to achieve a significant improvement in both attack-detection accuracy and usability.

There are various interesting areas of future work. Firstly, we chose to study only a selected sample from the design space of fingerprint representations in a within-subjects design, so we could facilitate a direct comparison between the different classes of fingerprints. Further work exploring line breaks, font settings, dictionaries, different attacker strengths, etc. will likely lead to further improvement possibilities.

While this work shows that there are better ways to represent key-fingerprints than currently being used, it does not explore what can be done to motivate more users to actually compare the fingerprints in the first place. Follow-up studies to research this important question are naturally an interesting and vital area of research.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments, and Trevor Perrin, Jake McGinty, Tom Ritter and Skylar Nagao for their discussion and excellent feedback.

References

[1] BADDELEY, A. Working memory. *Science 255*, 5044 (1992), 556–559.

[2] BARNARD, G. Significance tests for 2×2 tables. *Biometrika* (1947), 123–138.

[3] BIRYUKOV, A., DINU, D., AND KHOVRATOVICH, D. Argon2: the memory-hard function for password hashing and other applications. Tech. rep., Password Hashing Competition (PHC), December 2015.

[4] BONNEAU, J., AND SCHECHTER, S. Towards reliable storage of 56-bit secrets in human memory. In *Proceedings of the 23rd USENIX Security Symposium* (August 2014).

[5] BREITMOSER, V. pgp-vanity-keygen. `https://github.com/Valodim/pgp-vanity-keygen`, 2014.

[6] BUCKNER, R. L., PETERSEN, S. E., OJEMANN, J. G., MIEZIN, F. M., SQUIRE, L., AND RAICHLE, M. Functional anatomical studies of explicit and implicit memory retrieval tasks. *The Journal of Neuroscience 15*, 1 (1995), 12–29.

[7] CALLAS, J., DONNERHACKE, L., FINNEY, H., SHAW, D., AND THAYER, R. OpenPGP Message Format. RFC 4880 (Proposed Standard), Nov. 2007. Updated by RFC 5581.

[8] CRANNELL, C., AND PARRISH, J. A comparison of immediate memory span for digits, letters, and words. *The Journal of Psychology 44*, 2 (1957), 319–327.

[9] DE WINTER, J. C., AND DODOU, D. Five-point Likert items: t test versus Mann-Whitney-Wilcoxon. *Practical Assessment, Research & Evaluation 15*, 11 (2010), 1–12.

[10] DHAMIJA, R. Hash visualization in user authentication. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2000), CHI EA '00, ACM, pp. 279–280.

[11] DIERKS, T., AND RESCORLA, E. The Transport Layer Security Protocol Version 1.2. RFC 5246, Aug. 2008. Updated by RFCs 5746, 5878, 6176.

[12] ELECTRONIC FRONTIER FOUNDATION. Secure Messaging Scorecard. `https://www.eff.org/secure-messaging-scorecard`, 2014.

[13] ELLISON, C., ET AL. Establishing identity without certification authorities. In *USENIX Security Symposium* (1996), pp. 67–76.

[14] FARB, M., LIN, Y.-H., KIM, T. H.-J., MCCUNE, J., AND PERRIG, A. Safeslinger: easy-to-use and secure public-key exchange. In *Proceedings of the 19th annual international conference on Mobile computing & networking* (2013), ACM, pp. 417–428.

[15] GALBRAITH, J., AND THAYER, R. The Secure Shell (SSH) Public Key File Format. RFC 4716 (Informational), Nov. 2006.

[16] GOODRICH, M. T., SIRIVIANOS, M., SOLIS, J., TSUDIK, G., AND UZUN, E. Loud and clear: Human-verifiable authentication based on audio. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on* (2006), IEEE, pp. 10–10.

[17] GUTMANN, P. Do users verify SSH keys? *USENIX;login: 36*, 4 (2011).

[18] HOLM, S. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* (1979), 65–70.

[19] HSIAO, H.-C., LIN, Y.-H., STUDER, A., STUDER, C., WANG, K.-H., KIKUCHI, H., PERRIG, A., SUN, H.-M., AND YANG, B.-Y. A study of user-friendly hash comparison schemes. In *Computer Security Applications Conference, 2009. ACSAC'09. Annual* (2009), IEEE, pp. 105–114.

[20] JOSEFSSON, S. The Base16, Base32, and Base64 Data Encodings. RFC 3548 (Informational), July 2003.

[21] JOSEFSSON, S. The Base16, Base32, and Base64 Data Encodings. RFC 4648, Oct. 2006.

[22] JUOLA, P. Whole-word phonetic distances and the PGPFone alphabet. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on* (1996), vol. 1, IEEE, pp. 98–101.

[23] KEITH, M., SHAO, B., AND STEINBART, P. A behavioral analysis of passphrase design and effectiveness. *Journal of the Association for Information Systems 10*, 2 (2009), 2.

[24] LAURIE, B., LANGLEY, A., AND KASPER, E. Certificate Transparency. RFC 6962 (Experimental), June 2013.

[25] LUND, O. *Knowledge construction in typography: the case of legibility research and the legibility of sans serif typefaces.* PhD thesis, The University of Reading, Department of Typography & Graphic Communication., 1999.

[26] MCGRAW, K. O., AND WONG, S. A common language effect size statistic. *Psychological bulletin 111*, 2 (1992), 361.

[27] MELARA, M. S., BLANKSTEIN, A., BONNEAU, J., FREEDMAN, M. J., AND FELTEN, E. W. CONIKS: A Privacy-Preserving Consistent Key Service for Secure End-to-End Communication. Tech. Rep. 2014/1004, Cryptology ePrint Archive, December 2014.

[28] [MESSAGING] MAILING-LIST ARCHIVE. Usability of Public-Key Fingerprints. `https://moderncrypto.org/mail-archive/messaging/2014/000004.html`, 2014.

[29] MILLER, G. A. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review 63*, 2 (1956), 81.

[30] NAMECOIN PROJECT. Namecoin. `http://namecoin.info`, Nov. 2014.

[31] OGDEN, C. K., ET AL. System of Basic English. *Self-published* (1934).

[32] OLEMBO, M. M., KILIAN, T., STOCKHARDT, S., HÜLSING, A., AND VOLKAMER, M. Developing and testing a visual hash scheme. In *EISMC* (2013), pp. 91–100.

[33] [OPENPGP] IETF MAIL ARCHIVE. Fingerprints. `https://mailarchive.ietf.org/arch/msg/openpgp/2C9gTsxTgh29W8VX8x7OOYZqfUY`, 2015.

[34] PERCIVAL, C. Stronger key derivation via sequential memory-hard functions. *Self-published* (2009).

[35] PERRIG, A., AND SONG, D. Hash visualization: A new technique to improve real-world security. In *International Workshop on Cryptographic Techniques and E-Commerce* (1999), pp. 131–138.

[36] PINGEL, I., IRVING, A., GENERALMANAGER, WIKINAUT, TINLOAF, FARB, M., AND JPOPPLEWELL. Fingeprint exchange - issue #826 - whispersystems/textsecure - github.

[37] PLASMOID. Fuzzy Fingerprints: Attacking Vulnerabilities in the Human Brain. `http://www.thc.org/papers/ffp.html`, Oct. 2003.

[38] ROGERS, M., AND PERRIN, T. Key-Fingerprint Poems. `https://moderncrypto.org/mail-archive/messaging/2014/000125.html`, 2014.

[39] RYAN, M. D. Enhanced certificate transparency and end-to-end encrypted mail. In *NDSS* (2014), NDSS.

[40] SHAY, R., KELLEY, P. G., KOMANDURI, S., MAZUREK, M. L., UR, B., VIDAS, T., BAUER, L., CHRISTIN, N., AND CRANOR, L. F. Correct horse battery staple: Exploring the usability of system-assigned passphrases. In *Proceedings of the Eighth Symposium on Usable Privacy and Security* (2012), ACM, p. 7.

[41] UNGER, N., DECHAND, S., BONNEAU, J., FAHL, S., PERL, H., GOLDBERG, I., AND SMITH, M. Sok: Secure messaging. In *Security and Privacy (SP), 2015 IEEE Symposium on* (2015), IEEE, pp. 232–249.

[42] VASCO.COM. `http://www.vasco.com/company/about_vasco/press_room/news_archive/2011/news_diginotar_reports_security_incident.aspx`, Sept. 2011.

[43] WHATSAPP. Encryption Overview. `https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf`, Apr. 2016.

[44] YLONEN, T., AND LONVICK, C. The Secure Shell Transport Layer Protocol. RFC 4253, Jan. 2006. Updated by RFC 6668.

[45] ZIMMERMANN, P., JOHNSTON, A., AND CALLAS, J. ZRTP: Media Path Key Agreement for Unicast Secure RTP. RFC 6189 (Informational), Apr. 2011.

A Appendix

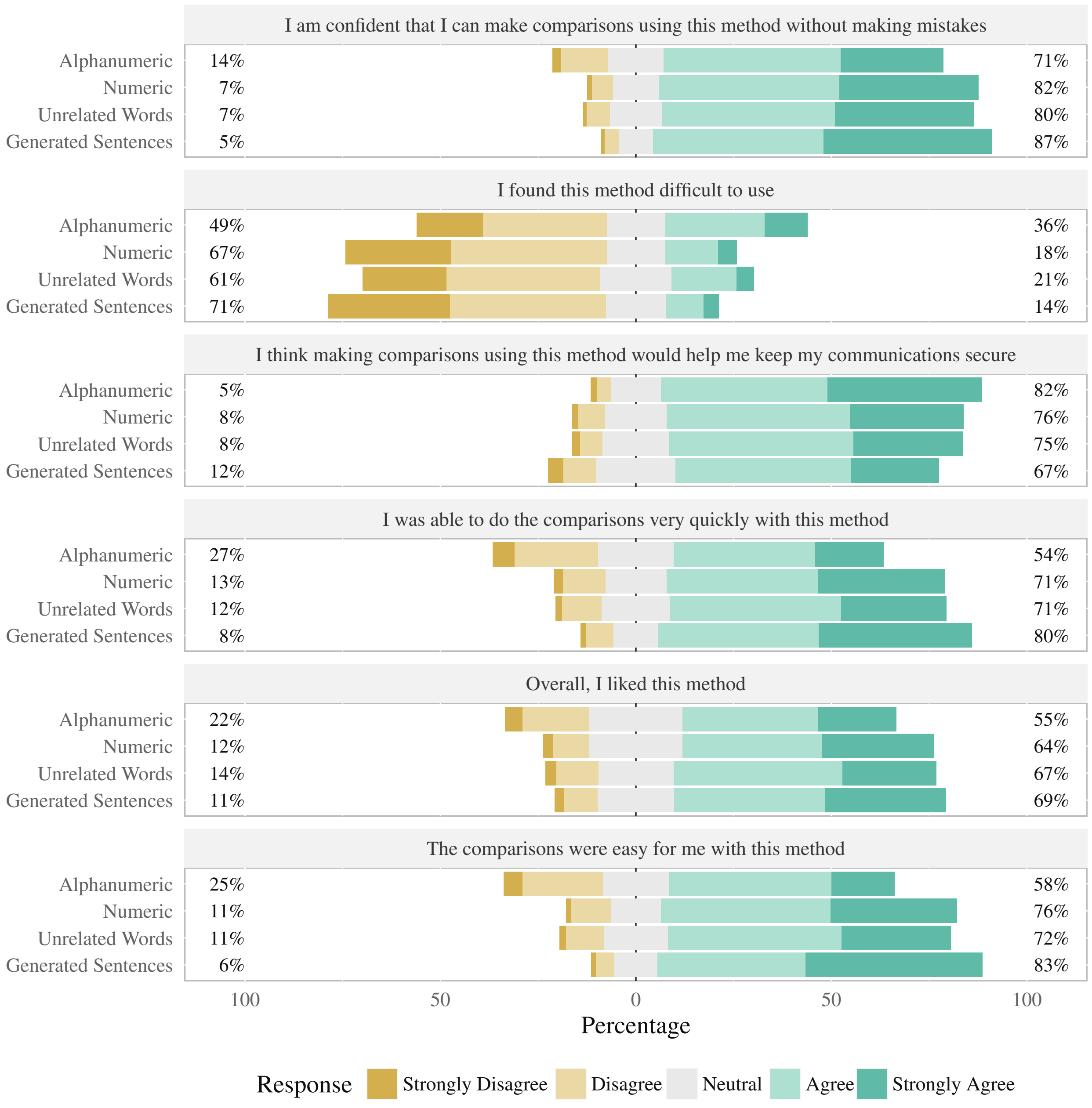


Figure 6: Survey results for all statement ratings