Can Unicorns Help Users Compare Crypto Key Fingerprints?

Joshua Tan, Lujo Bauer, Joseph Bonneau†, Lorrie Faith Cranor, Jeremy Thomas, Blase Ur*

Carnegie Mellon University, {jstan, lbauer, lorrie, thomasjm}@cmu.edu
† Stanford University, jbonneau@cs.stanford.edu
* University of Chicago, blase@uchicago.edu

ABSTRACT

Many authentication schemes ask users to manually compare compact representations of cryptographic keys, known as fingerprints. If the fingerprints do not match, that may signal a man-in-the-middle attack. An adversary performing an attack may use a fingerprint that is similar to the target fingerprint, but not an exact match, to try to fool inattentive users. Fingerprint representations should thus be both usable and secure.

We tested the usability and security of eight fingerprint representations under different configurations. In a 661-participant between-subjects experiment, participants compared fingerprints under realistic conditions and were subjected to a simulated attack. The best configuration allowed attacks to succeed 6% of the time; the worst 72%. We find the seemingly effective compare-and-select approach performs poorly for key fingerprints and that graphical fingerprint representations, while intuitive and fast, vary in performance. We identify some fingerprint representations as particularly promising.

ACM Classification Keywords

K.6.5 Security and Protection: Authentication; H.5.2 User Interfaces: Evaluation/methodology

Author Keywords

usability; key fingerprints; authentication; secure messaging

INTRODUCTION

To protect the privacy of communications like email and instant messaging, users can encrypt messages using public-key encryption. For Alice to send a message to Bob that only Bob can read, she needs to encrypt the message with Bob's public key. Bob will use his private key to decrypt the message.

While this method of securing communication is believed to be technically sound, it hinges on Alice knowing Bob's public key. To learn Bob's key, Alice would typically look it up on a web site (e.g., a public key server) that publishes such information. Unfortunately, an attacker seeking to intercept Alice's communications to Bob might try to add his own key to the key server under Bob's name. When trying to find Bob's public key, Alice would then unwittingly download the attacker's key. Any messages she composed for Bob would then be readable by the attacker, and not by Bob.

A more reliable method would be for Bob to deliver his public key to Alice in person. Because public keys are long strings of arbitrary bits, this approach is unfortunately unwieldy and impractical. A common alternative is for Bob to give Alice a *fingerprint* of his key, which is a short digest (hash) of the key. Alice then manually compares (e.g., looks at them side by side) the fingerprint received from Bob to the fingerprint computed from the key she downloaded from the key server. Fingerprints are by design long enough for it to be exceedingly unlikely that two different keys will have the same fingerprint, yet short enough for manual comparison to be feasible.

Fingerprint verification is only useful, however, if Alice is able to determine easily and successfully whether the fingerprint she obtained from Bob matches the one she computed. If Alice is only comparing the first part of the two fingerprints, for example, this opens the door to attackers who try to create a public key whose fingerprint will be *similar* to Bob's key's fingerprint, in the hope that Alice's (cursory) examination will not distinguish it from the real fingerprint.

Fingerprints can be represented in many ways, which may impact the efficiency and accuracy with which users compare them. Besides the commonly used hexadecimal format [2, 8], other representations used in practice include ASCII art [22], numbers [34], pronouncable strings [13], and avatars [20]. Additional representations have been proposed, including abstract art [23], sentences [1], snowflakes [18], and fractal flames [25].

In this paper we report on the results of a 661-participant online study through which we compare the usability and efficacy of a range of fingerprint representations and configurations. We test eight different representations and examine how likely users are to notice fingerprint mismatches caused by an attacker who creates public keys whose fingerprints are similar to the authentic key's fingerprint. We include fingerprint rep-

CHI 2017 May 06-11, 2017, Denver, CO, USA

ACM ISBN 978-1-4503-4655-9/17/05.
DOI: http://dx.doi.org/10.1145/3025453.3025733

resentations used in practice (e.g., hexadecimal strings, ASCII art) as well as ones hypothesized to be good alternatives (e.g., randomly generated images of unicorns). We also examine different approaches used to compare fingerprints (seeing two fingerprints side by side versus selecting the correct fingerprint from a list) and the effect on security against differently powerful attackers. We pay special attention to our experiment's realism, simulating time pressure that users may feel in practice when performing security tasks, accounting for the fact that attacks are rare rather than the typical case, and simulating other practical challenges (e.g., comparing a fingerprint on a business card to one on a screen).

Our results include findings important for designing systems that involve comparing fingerprints. We find that graphical representations, thought promising [11], have mixed success. Although all allowed quick comparisons, some were much more susceptible to attack than more standard representations. We also find that the compare-and-select method of comparing fingerprints resulted in many users failing to detect mismatches, to the point where its use for fingerprint comparison should be strongly discouraged. Echoing prior work [6], we find that non-hexadecimal textual representations have promise, especially for usability. The traditional hexadecimal representation, however, overall fares surprisingly well.

These findings lead to a set of suggestions for when a fingerprint representation may be appropriate. When security is paramount, none of the representations tested seem adequate. When the risk and impact of attack is low but usability is paramount, visual representations excel. For casual use, when security and usability need to be balanced, textual representations, including hexadecimal, seem to be the most appropriate.

BACKGROUND AND RELATED WORK

In this section we discuss the uses of fingerprints and prior attempts to improve their usability. We also explain how to use entropy to quantify the security afforded by fingerprint representations. Finally, we relate attacker strengths to real-world costs for performing brute-force attacks on fingerprints.

Fingerprint Applications

One type of attack in communication systems is a man-in-the-middle (MitM) attack, in which an attacker inserts himself or herself between two communicating parties in such a way that neither party is aware of the attacker. Once inserted, the attacker can eavesdrop or actively manipulate communications. Fingerprint comparisons enable a user to detect MitM attacks.

Well-known applications that make use of fingerprints include GnuPG [2], a tool for encrypting communications, and OpenSSH [8], commonly used for remote access to servers. Off-the-Record (OTR) Messaging applications provide multiple ways to authenticate message recipients, including fingerprint verification [21]. Many popular secure chat smartphone apps also use fingerprints, usually in a layered approach in which fingerprint comparison is optional [34, 36].

A variety of fingerprint representations and formats are used, the majority of which are textual. Examples of textual representations used in real systems are shown in Table 1. Graphical representations have seen limited use; examples include Peerio [20], which uses an avatar representation, and OpenSSH Visual Host Key, which resembles ASCII art (Figure 2).

GnuPG	3A70 F9A0 4ECD B5D7 8A89 D32C EDA0 A352 66E2 C53D
OpenSSH	ef:6d:bb:4c:25:3a:6d:f8:79:d3:a7:90:db:c9: b4:25
bubblebabble	xucef-masiv-zihyl-bicyr-zalot-cevyt-lusob- negul-biros-zuhal-cixex
OTR	4206EA15 1E029807 C8BA9366 B972A136 C6033804
WhatsApp	54040 65258 71972 73974 10879 55897 71430 75600 25372 60226 27738 71523

Table 1: Examples of textual fingerprint representations used in actual applications.

Usability of Fingerprints

Public key cryptography has long had usability and adoption issues [3, 9, 10, 26, 35]. Fingerprints have been part of the problem. In a user study on OTR messaging, participants were confused by fingerprints and struggled to verify them correctly [28]. In order to make fingerprint verification more usable, researchers have explored a variety of approaches.

One way to make comparison easier is to shorten fingerprints. This approach is used in Short Authentication Strings (SAS). SAS can provide reasonable security using only a 15-bit string [33], though they are primarily useful in synchronous environments. Alternatively, the computational security of small fingerprints can be increased by slowing down the hashing algorithm using a stretching function. In our study, we focus on fingerprint comparisons that apply to public-key fingerprint verification, which is traditionally asynchronous.

Adding structure to a fingerprint representation may make it easier to compare. Textual fingerprint representations can be separated into smaller chunks. Representing fingerprints as pronounceable words or sentences may also facilitate comparison. For graphical representations, structured images resembling abstract art have been suggested for improving usability. These include Random Art [23] and OpenSSH Visual Host Key, which was inspired by Random Art [22]. Another way to add structure is to represent fingerprints as avatars, such as unicorns [32] or robots [5]. In our study, we examine textual and graphical representations with varying degrees of structure.

Different comparison modes have also been proposed, primarily for SAS-based device pairing or synchronous authentication [7, 27]. These include compare-and-confirm (compare two strings and indicate if they match), compare-and-select (compare one string to a set of others and select the matching option), and copy-and-enter (copy a string from one device to another and let the device itself perform the check). Compare-and-select has also been used for anti-phishing tools that ask users to select the website they want to visit from a list, rather than ask for a yes/no answer to whether users would like to proceed to a given website [37]. In our study, we test both compare-and-confirm and compare-and-select. In particular, we explore compare-and-select due to its potential benefits for inattentive users. Prior work has postulated that compare-and-

select may help prevent users from "verifying" fingerprints without actually comparing them [7, 31].

A number of usability studies on device pairing have explored representations and comparison modes similar to those we test [14, 16, 17]. They considered representations such as numbers, images of visual patterns, and phrases, as well as various comparison modes. In each of these studies, participants performed comparison tasks similar to ours in a lab setting. In some cases, participants were also subjected to a simulated MitM attack [14, 17]. These studies had mixed findings; for example, Kainda et al. conclude that compare-and-confirm and compare-and-select should not be used because it is subject to security failures [14], while Kumar et al. recommend that compare-and-confirm with numbers be used due to its low error rate and comparison speed [17]. Although these studies were useful for informing our selection of fingerprint representations and comparison modes, a number of differences limit the extent to which their findings might extend to verifying public key fingerprints. These differences include the size of fingerprints tested (15-20 bits, compared to at least 128 bits in our study) and the methodology used (lab studies are often too small to perform statistical testing).

In a 400-participant online study, Hsiao et al. examined the speed and accuracy of fingerprint comparisons under different representations [11]. In contrast to lab studies, their methodology enabled statistical testing. However, with one exception, all fingerprints were sized between 22-28 bits, so it is unclear how many of their findings might translate to our setting. The exception to this caveat is that they tested Random Art [23], which can encode bit sizes comparable to those we explore. They found Random Art performed well in both accuracy and speed, recommending its use for color-display devices with sufficient computational power. We examine Vash, an open-source implementation of abstract art fingerprints [4, 30]. Although the two representations are similar, Hsiao et al.'s findings for Random Art may not extend to the attacks we consider; the similar-looking pairs they used were selected from only 2000 Random Art images, whereas we consider attackers who can generate 2^{60} candidate images.

Other than recent work by Dechand et al. [6], scant research has tested representations suitable for key fingerprints. That study involved a large-scale, within-subjects experiment on Mechanical Turk (MTurk) in which participants compared fingerprints displayed in different textual representations, including hexadecimal, numbers, words, and sentences. They measured comparison speed and accuracy, and they also recorded whether participants correctly compared fingerprints for multiple simulated 2^{80} attacks. They found that hexadecimal performed significantly worse than numbers and sentences in both attack detection rates and usability ratings. In addition, they found that sentences had a significantly higher attack detection rate than numbers, while also being rated as more usable.

Our work shares many similarities with Dechand et al.'s study [6], particularly in the textual representations tested. Similar to their study, we also subject participants to simulated attacks for textual representations to determine the usability and security of those representations. Unfortunately, direct comparison of our work to theirs is difficult due to parameter differences in the textual representations we test, in particular the chosen security level for fingerprints.[1] The other primary differences between the previous study and ours are: we evaluate fingerprint comparisons under realistic conditions of habituation and distraction using a between-subjects design; we explore graphical representations; we perform an initial investigation on compare-and-select for cryptographic key fingerprints; and we test additional attacker strengths.

Entropy as a Security Metric

Entropy can be used to measure the computational security afforded by different fingerprint representations. If users compare fingerprints fully, then the entropy of a fingerprint representation quantifies the average work needed to find a key whose fingerprint collides with the target fingerprint. If users do not compare fingerprints completely, but only compare certain aspects, then an intelligent attacker may attempt to only match the aspects he expects will be actually compared. This type of attack, in which the adversary attempts to find a visually similar fingerprint to the target fingerprint, has been explored for both the hexadecimal and graphical fingerprints used in OpenSSH [19, 24]. More recently, the previously mentioned study by Dechand et al. investigated users' ability to detect such attacks for different textual representations [6].

The reduction in entropy of the original representation (after fixing the matched aspects) can be used to quantify the work an attacker must spend to produce a key whose fingerprint matches specific aspects of the target fingerprint. This approach to quantifying attacker work has the added benefit of being independent of the particular binary encoding used for a fingerprint. A 2^{60} attack corresponds to an attacker that generates 2^{60} keys, computes the fingerprint for each, and then (manually or programmatically) selects the key whose fingerprint maximizes similarity according to some metric. Stevens et al. estimate the cost of renting CPU/GPU time from EC2 to find a SHA-1 collision, which requires resouces similar to those to perform a 2^{60} attack on fingerprints. They estimate this cost to be between 75K and 120K USD, which they note is within the budget of criminal organizations [29].

METHODOLOGY

We conducted a between-subjects experiment to evaluate and compare the usability and security of fingerprint representations and configurations. We recruited participants from MTurk in August 2016. We required participants be 18 years or older and live in the United States. Our protocol was approved by our university's IRB.

We advertised the study as a "role-playing activity involving technology and communication in the workplace" that would take about 20 minutes. We compensated participants $3, with the opportunity to earn a $1 bonus. As our activity was not designed for use on tablets or smartphones, we asked that participants use a desktop or laptop computer. Participants

[1] We began our study prior to publication of their work, limiting our ability to choose parameters consistent with their study.

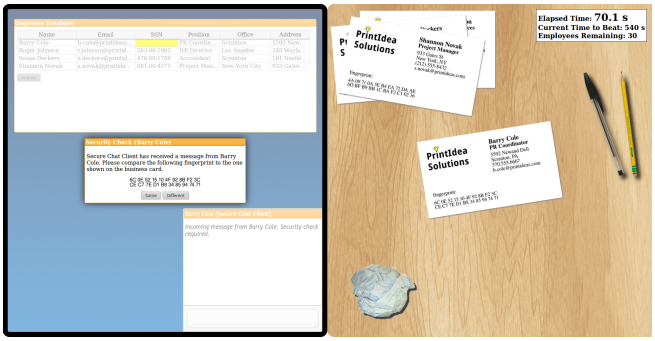


Figure 1: Screenshot of the task. This example is the compare-and-confirm, simultaneously visible, hexadecimal condition.

were randomly assigned to a condition, which determined the fingerprint representation and configuration they saw.

We asked participants to imagine they worked as an accountant at a company that was updating its employee database. To perform this update, participants had to retrieve the social security numbers (SSNs) for 30 employees and enter them into a database. We chose SSNs to motivate the need for secure communication. In the U.S., SSNs are highly sensitive because knowing an individual's SSN can enable identity theft and have other financial ramifications. Our activity web page divided the browser window into two sections that mimicked the appearance of a computer screen and a desk (Figure 1). The computer screen section displayed a spreadsheet-like database where participants would need to fill in missing SSN details for employees. Several business cards were sitting on the desk. A stopwatch and information about the participant's progress completing the task appeared in the top right corner.

We provided the instructions for the activity via an interactive tutorial. Participants could repeat the tutorial any number of times until they were comfortable proceeding, and the on-screen stopwatch did not begin until after the tutorial ended.

At the start of a task, a chat window appeared on the simulated computer screen informing participants of an incoming message from one of the 30 employees. To proceed, they had to perform a security check. Shortly afterward, a dialog box was displayed on the computer screen containing a fingerprint and instructions to compare it to the fingerprint on the employee's business card, which appeared simultaneously on the desk.

For our baseline configuration, a security check involved comparing two fingerprints (one in the security check dialog box and one on the business card) and pressing a button to indicate if they were the same. We informed participants that this check was needed to ensure a secure chat session and avoid potential eavesdroppers. Depending on which fingerprint representation was shown, we provided guidance on what differences participants should look for when comparing fingerprints.

If the participant indicated that the fingerprints matched, the chat window displayed a message from the employee with her SSN. The participant was instructed to type the SSN into the database. If the participant instead indicated that fingerprints did not match, the chat message instructed the participant to instead enter "ERROR" in the database. Each participant repeated this task for 30 employees.

We instrumented our activity to record participants' database entries, security-task decisions, and detailed timing information. We also recorded their browser user agent strings to determine whether participants used a tablet or smartphone. For one condition in which users had to toggle between two views, we recorded the number and timing of toggles.

Afterwards, participants filled out a survey. In this survey, we told participants whether they had missed our attack and asked them to explain why they thought they missed or detected that attack. To aid in memory, we showed the fingerprint pair corresponding to the attack alongside these questions. We asked participants to describe their strategy for comparing fingerprints, respond on a Likert scale to statements about the fingerprints they saw, and provide general demographic data.

Security Task Design Considerations

Security tasks are rarely performed for their own sake. Fingerprint comparisons are secondary to a primary purpose, such as communicating with someone. Combined with the pressures and stresses of everyday life, this state of affairs often results in users performing security tasks while distracted or otherwise not fully attentive. In addition, few users will have previously been the target of a MitM attack and might have little reason to believe they would become such a target. Users asked to compare fingerprints might only encounter mismatching fingerprints due to device misconfiguration, the acquisition of a new device, or security software re-installation. Many design decisions for our activity reflect these real-world factors.

To increase distraction and stress, we incentivized participants to perform the task both quickly and correctly by informing them that the "15% fastest participants with the fewest mistakes" would receive an additional $1 bonus. We considered a mistake to be entering an incorrect SSN into the database or failing the security check for an employee. The interface contained both a persistent reminder of this bonus and a timer showing the elapsed time, a target time participants should try to beat, and the number of employees remaining. We also highlighted this box in the tutorial. To avoid cases where participants felt under-pressured due to exceeding the target time, we noted that beating the target time did not guarantee the bonus as future participants could lower or raise it.

We expected most participants would have little to no experience comparing key fingerprints and thus would lack the expectations typically held by users who frequently compare fingerprints. In an effort to ingrain these expectations quickly, we sacrificed realism with respect to the role-playing activity. To habituate participants to benign situations, 28 of the 30 comparison tasks involved matching fingerprints. One fingerprint comparison task involved obviously different fingerprints, reflecting what would commonly be seen by users in benign situations (e.g., misconfigurations). This task was shown in a randomly determined position between the second and fifth pairs, inclusive. We also used this task to filter out participants who mindlessly clicked through the entire activity.

Threat Model and Simulated Attack

In our threat model, an adversary attempts a MitM attack on a specific user in the context of a fingerprint comparison task. We assume preimage attacks on the user's fingerprint are infeasible. Instead, an adversary attempts to present a key whose fingerprint is similar to the target user's fingerprint. We assume the adversary has finite resources, limiting how similar the attack fingerprint can be to the target. Participants were shown a single simulated attack according to this threat model. The comparison task in which the attack appeared was randomly chosen from between the 25th and 29th pairs, inclusive. To minimize bias, we generated three attack instances for each configuration and randomly selected one of these three for each participant. Since we tested attack strengths requiring computational resources not available to us, we simulated these attacks; we generated a 2^{60} attack by creating a version of the target fingerprint whose similarity was such that it would take 2^{60} bruteforce attempts on average to produce.

Applications can optionally use key stretching to increase resistance to brute-force attacks. WhatsApp implements this approach using iterated hashing [34]. The simulated attacks in our study assume that hash strengthening techniques are not applied. Thus, our attack detection results directly apply to applications such as GnuPG and OpenSSH, which do not currently implement such defenses. However, the attack strengths we test can be translated to applications that do use hash strengthening. Specifically, if a fingerprint scheme employs hash strengthening to require an additional 2^{20} work per generated fingerprint (reasonable even on mobile devices), then our results for 2^{60} attacks on fingerprints without strengthening translate to 2^{80} attacks on fingerprints with strengthening.

Experimental Factors

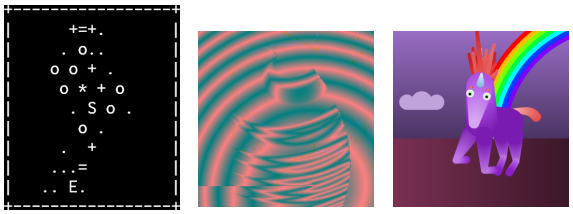
Representations

For textual representations, we chose to target a fingerprint security level of 160 bits. This is the same fingerprint security provided by current implementations of GnuPG, which uses SHA-1 for its hash function. Where applicable, textual representations were chunked in groups of four, with chunks separated by spaces. This chunk size performed well in prior work [6]. For our graphical representations, we evaluate the representation implementation in its original form, leaving the fingerprint security as is. Table 2 demonstrates our textual representations and Figure 2 our graphical representations. As we introduce the representations, we note the number of bits representing the space of possibilities that can be generated using that representation with those parameters. For all representations, both textual and graphical, the number of possibilities that a human can distinguish can only be determined empirically, which is implicit in the rate at which participants in our study detect attacks.

For textual representations, we tested hexadecimal (uppercase and lowercase), alternating vowels/consonants, words, numbers, and sentences. Hexadecimal was 40 characters long (160 bits), numbers was 48 digits long (159.5 bits), alternating was 48 characters long (161.1 bits), and words was 16 words long (155.7 bits). We selected words from Ogden's Basic English word list.[2] For sentences, we used an implementation based on a deterministic sentence generator [1] (159.8 bits). For this representation, the average number of sentences is 3.6 (max: 7) and the average length of the longest sentence is 9 words (max: 12). Each of hex, numbers, and alternating vowels/consonants was equally spread over two lines. Words were spread over 4 lines. For the sentences representation, each sentence began on a separate line, wrapping where necessary.

Hexadecimal	BAAA 9AE6 7B8B 0D41 BD83 05E7 5209 8EDF 1058 41F6
Alt vow./cons.	bunu difu tura wefi wiwe haqe tano haco qevu cori qife nufi
Words	learning equal education bent collar religion new shelf angle table train sad keep meal thing punishment
Numbers	7748 5689 7453 6977 5604 5939 2765 8791 5022 4957 3805 0309
Sentences	The basket ends your right cat on his linen. Her range repeats her nerve. The smile tells secretly. My clean cake pulls your waiting pocket.

Table 2: Textual fingerprint representations used for experiments. For hexadecimal, we tested both uppercase and lowercase variations.



(a) OpenSSH Visual Host Key (b) Vash (c) Unicorn

Figure 2: Visual fingerprint representations.

For graphical representations, we test OpenSSH Visual Host Key ($\leq$ 128 bits), Vash ($\approx$ 5,438 bits), and unicorns [32] ($\approx$ 2,854 bits).[3] Visual Host Key was included because it is widely deployed in SSH software. Prior work explored Random Art fingerprints [23], of which Vash is an open-source implementation. We included unicorns to test fingerprints that use avatar-like representations. We limited consideration to those representations whose entropy was large enough for use as cryptographic fingerprints in asynchronous settings (i.e., at least 128 bits). Unicorn fingerprints were generated by a program in which unicorn attributes were set according to numbers drawn from a pseudorandom number generator. The key to be hashed served as the seed to the generator. For Vash fingerprints, which can viewed abstractly as a graph, a similar process was used to determine graph node types and properties, which uniquely determine the fingerprint appearance.

[2] http://ogden.basic-english.org/words.html

[3] Given the computational difficulty of an exact calculation, for our graphical representations we roughly estimate security.

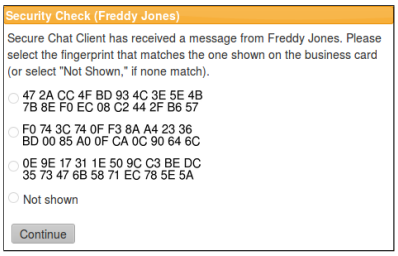


Figure 3: Security task dialog for participants in the compare-and-select, hexadecimal condition.

For all representations, we allowed the fingerprint to take up a maximum of 50% of the card, horizontal or vertical. For sentences, this required formatting the fingerprint using a slightly smaller font size than for the other textual formats. For textual representations, we advised participants that they should ignore differences in font size or type. In addition, for words and sentences, we informed participants that differences in fingerprints due to misspelled words would not occur.

For each representation, we thus needed to pick the specific aspects that would be matched and ensure that the reduction in entropy matched the assumed attacker strength. Our overall strategy was to match the aspects we expected users would focus on during comparisons. For textual representations, we primarily matched the beginning and end of lines. For graphical representations, we mainly attempted to match "big picture" elements, such as overall pattern and color. For Vash, we matched node types up to a certain depth. For unicorns, we chose to roughly match many aspects, such as background hue and horn length. Images of each attack used in our experiment are included as supplementary material.

Comparison Mode

We tested both compare-and-confirm and compare-and-select, as previously described. While compare-and-confirm is the traditional method of fingerprint comparison, we were interested in the performance of compare-and-select, given its potential benefits for inattentive users. An example of compare-and-select is shown in Figure 3. Our implementation is similar to that used in SafeSlinger [7]; three fingerprints are shown in a random order, with two fingerprints randomly generated and one fingerprint corresponding to the received fingerprint.

Visibility Mode

In most cases, participants tasked with comparing fingerprints are able to see both fingerprints simultaneously. For example, to compare a fingerprint on a business card to one on a computer screen, the user can simply hold the business card up to the screen to place the fingerprints side-by-side. However, in certain use cases, it may not be possible or easy to view both fingerprints simultaneously in order to compare. As an example, many versions of Android do not have any easy way to view two applications in a split-screen view. In this case, if the user needs to compare fingerprints shown in two applications (say, a fingerprint shown in a secure chat app and one shown on a website), the user will need to toggle back and forth in order to compare.

We tested situations in which fingerprints are both visible simultaneously as well as situations in which the user must toggle between them. We expected the need to toggle to affect representations differently, as it may be easier to place certain representations in short-term memory than others.

Attack Strength

We considered three different attack strengths: 2^{40}, 2^{60}, and 2^{80}. These strengths correspond to the estimated capabilities of an attack performed on commodity hardware, an attack performed by a well-funded criminal organization, and an attack performed by a state-sponsored actor.

Other Factors

We included two other experimental factors. For hexadecimal, we tested the impact of letter case on performance. We were interested in this because both types have been used for real security applications (e.g., lowercase in OpenSSH and uppercase in GnuPG). We also tested a variation of our activity in which the target time-to-beat was doubled, from 540 seconds to 1080 seconds. We tested this to provide insight into the extent to which our results depend on the specific target time that was used.

Experimental Conditions

Participants were randomly assigned to one of 17 experimental conditions, which determined the specific fingerprint representation, configuration, and attacker strength used for that participant in the activity. Eight of our conditions assumed an attacker strength of 2^{60} and varied only according to fingerprint representation. To explore the effect of different attacker strength assumptions, we tested four additional conditions: 2^{40} attacks on uppercase hexadecimal and unicorns and 2^{80} attacks on uppercase hexadecimal and Visual Host Key.

The four remaining experimental conditions were designed to be compared with our baseline hexadecimal condition, namely uppercase hex using the compare-and-confirm configuration in which both fingerprints to be compared were simultaneously visible, with an assumed attacker strength of 2^{60}. These conditions varied from the baseline condition with respect to only one factor: compare-and-select (comparison mode), toggle (visibility mode), lowercase (hex letter case), and twice the target time-to-beat.

We performed some modifications to test fingerprint comparisons under adverse conditions. In all conditions, fingerprints were displayed in different font types and sizes. The business cards were presented randomly tilted up to 10 degrees. For the Vash and unicorn representations, we applied a random gamma correction to the fingerprint shown on the computer screen section in the range [0.8, 1.2], in order to simulate the effect of an improperly calibrated display. For sentences, we showed fingerprint pairs formatted such that line breaks occurred in different places.

Statistical Analysis

We performed hypothesis tests for each metric we measured. To test for significant effects with respect to the proportion of security failures across conditions, we used Pearson's chi-squared test (for omnibus tests) and Fisher's exact test (to

compare two conditions). To test for significant effects for median comparison time, number of false positives, and Likert ratings, we used the Kruskal-Wallis test (for omnibus tests) and the Mann-Whitney-Wilcoxon test (to compare two conditions). Given the large number of comparisons we made, we applied the Holm-Bonferonni method and report corrected p-values. All hypothesis tests used a significance value of $\alpha = 0.05$.

Limitations

For each representation, we used an attack strategy to maximize similarity to the target fingerprint for particular aspects (or in particular locations) of that fingerprint. Our goal was to find the most effective attack given a fixed computational budget. Nonetheless, it is unlikely that we selected the best possible attack for each representation. Given this limitation, we believe that our statistical power, sufficient to detect only large differences in attack detection rate, is appropriate. Such large differences, if they exist, may indicate weakness intrinsic to a representation or configuration, not only in our attack strategy.

Our participants were recruited from MTurk, which is not representative of the general U.S. population; MTurk workers have been found to be younger and better educated [15].

To habituate participants to benign security scenarios in a short amount of time, we had to sacrifice some realism. However, feedback from participants indicates that we achieved our goal of recreating the types of conditions under which people would likely perform fingerprint comparisons as a security task.

Since the primary goal of most MTurk users is to get paid, and since we tied the bonus payment to participants' performance on the security tasks in our activity, the security tasks were not secondary to some other task, as would be the case in the real world. However, as previously explained, we believe we captured the desirable characteristics intrinsic to security as a secondary task.

Participants

A total of 677 participants completed our MTurk HIT. We excluded 16: 3 participants used an Android or iOS device, 3 encountered technical issues, and 10 failed an attention check. The average time spent on our HIT was 14 minutes and 12 seconds. We paid all workers that accepted our HIT.

Our reduced sample consisted of 661 participants. Participants' ages ranged from 18 to 74 with an average of 33 years ($\sigma =$ 9.7). Participants were 44% female and 55% male, with 1% choosing not to specify. The two most common education levels were a four-year college degree (35%) and some years of college without finishing (27%). The most frequently reported occupations were service (16%); business, management, or financial (13%); and computer engineering or IT professional (12%). We considered participants as technical if two out of three of the following were true: they listed their occupation type as computer engineering or IT professional, they knew a programming language, or they indicated that people often asked them for computer-related advice. According to this definition, 18% of participants qualified as technical.

Condition	Frac. Missed	M(comp)	# Part.
hex, confirm, bothvis, 2^60	0.21	8.03	42
num, confirm, bothvis, 2^60	0.35	8.09	43
alt, confirm, bothvis, 2^60	0.17	8.71	40
word, confirm, bothvis, 2^60	0.14	6.70	42
sent, confirm, bothvis, 2^60	0.06	7.57	33
ssh, confirm, bothvis, 2^60	0.10	5.51	42
uni, confirm, bothvis, 2^60	0.54	2.04	39
vash, confirm, bothvis, 2^60	0.12	2.17	33
hex, **select**, bothvis, 2^60	0.72	5.21	47
hex, confirm, **toggle**, 2^60	0.30	9.79	40
vash, confirm, **toggle**, 2^60	0.27	5.18	33
hex, confirm, bothvis, **2^40**	0.06	9.20	31
uni, confirm, bothvis, **2^40**	0.67	1.84	42
hex, confirm, bothvis, **2^80**	0.37	8.80	43
ssh, confirm, bothvis, **2^80**	0.25	4.10	32
hex, confirm, bothvis, 2^60, **2x time**	0.10	11.04	40
hex (low), confirm, bothvis, 2^60	0.21	8.22	39

Table 3: Summary statistics by condition, including median comparison time (*M(comp)*), fraction of participants that missed the attack, and total number of participants.

RESULTS

We first describe the performance of different fingerprint representations. We then describe the effects of different ways of eliciting confirmation (compare-and-confirm vs. compare-and-select) and varying whether users could see the two fingerprints they were comparing one at a time or both at once. Finally, we discuss participants' self-reported strategies for comparing fingerprints. An overview of our results is provided in Table 3.

Representations

Attack Detection Rate

The fraction of participants who failed to notice a simulated 2^{60} attack varied significantly by condition ($\chi^2 = 131.93$, $df = 16$, $p < .001$). The best performing representation was sentences, causing participants to miss just 6% of attacks; unicorns, surprisingly, were worst, with participants missing 54% of attacks. Our baseline, the hexadecimal representation, was roughly in the middle, with 21% of participants missing the attack. The uppercase and lowercase variants of hexadecimal had an attack success rate within 1% of each other. Figure 4 summarizes these results.

The difference in attack success rate between hexadecimal (21%) and unicorns (54%) was borderline significant ($p =$ .052). Unicorns performed significantly worse than both Vash ($p = .003$) and Visual Host Key ($p < .001$). In contrast to related work by Dechand et al. [6], we did not observe any statistically significant differences in attack success rate between textual representations.

We also tested whether participants who fell under our definition of technical were more successful at detecting attacks than those who did not. Technical users were better at detecting attacks (80% to 69%), but this difference was not statistically significant after correction ($p = .08$). Similarly, for the three representations in which we varied attack strength (Visual

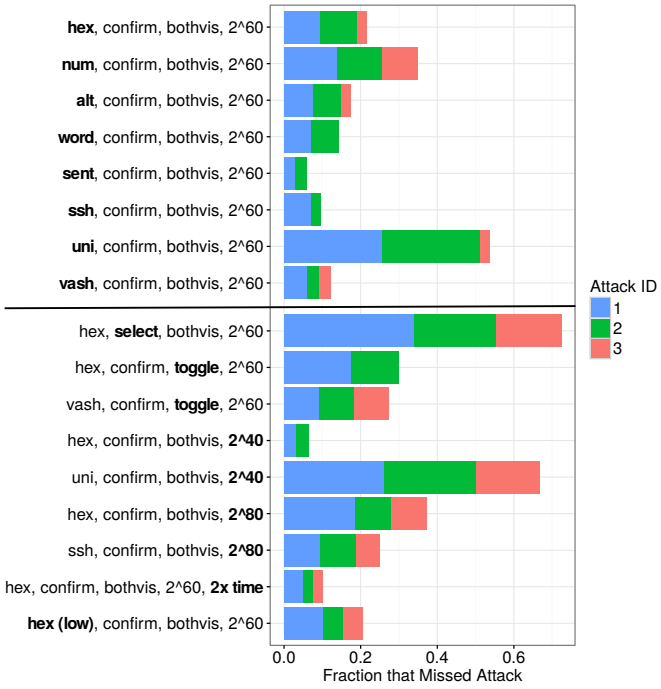


Figure 4: Fraction of participants in each condition that missed the attack, grouped by condition and attack instance. The attack IDs correspond to three different attacks we generated for each configuration.

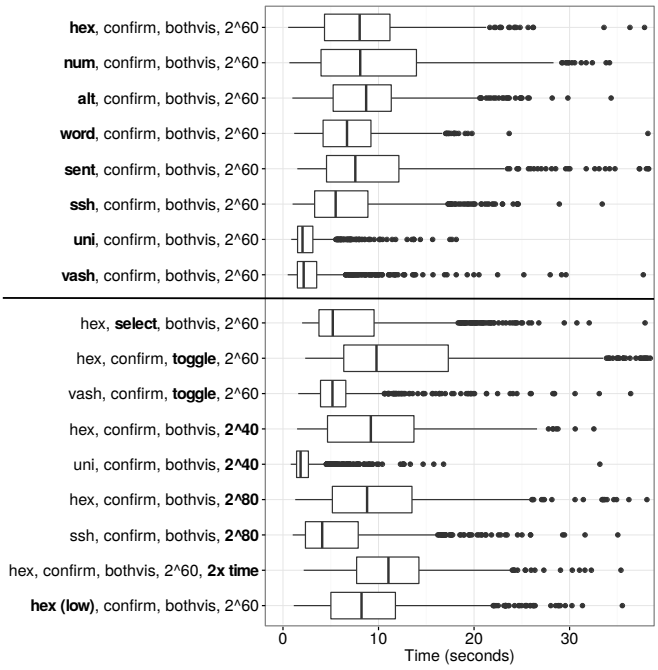


Figure 5: Median comparison time by condition.

Host Key, uppercase hexademical, unicorns), we did not observe any statsticically significant differences in the fraction of participants that missed an attack between attacks of different strengths.

Comparison Time

The median time spent comparing fingerprints[4] ranged from 2.0 seconds for the unicorns representation to 8.7 seconds for the alternating vowels/consonants representation. Graphical representations were generally faster to compare than textual ones. The median comparison times for each representation are shown in Figure 5.

Differences in comparison time between conditions were significant ($\chi^2 = 289.39$, $df = 16$, $p < .001$). Looking at individual conditions, the median comparison time was significantly lower for unicorns (2.0 s) compared to both hexadecimal (8.0 s; $p < .001$) and Visual Host Key (5.51 s; $p < .001$). In contrast to related work by Dechand et al. [6], we did not observe any statistically significant differences in comparison time between textual representations.

Subjective Ratings

We asked participants for their subjective ratings before we revealed whether they had missed the attack. For all representations, most participants (70% for alternating to 91% for Vash) believed that the time it took to compare fingerprints was reasonable for a security check. The majority also thought that it was easy to compare fingerprints and were confident that they could do so correctly. We did not observe statistically significant differences in ratings by condition for confidence ($p = .386$), ease of use ($p = .102$), or reasonableness of comparison time ($p = .117$).

[4]For participants in experimental conditions where the time to beat was set at 540 seconds.

Compare-and-select

Compare-and-select participants did not spend much time comparing fingerprints. Of the configurations involving textual representations, the compare-and-select configuration had the lowest median comparison time (5.2 seconds), though this difference was not statistically significant.

Across all experimental conditions, the compare-and-select condition had the lowest attack detection rate; 72% of participants missed the simulated attack. The difference in attack detection rate between our baseline compare-and-confirm hexadecimal condition and the compare-and-select hexadecimal condition was statistically significant ($p < .001$).

Toggle Use Case

Our toggle use case explored the effect of requiring users to toggle back and forth between fingerprints in order to make comparisons, as opposed to being able to view both simultaneously. For hexadecimal, we did not observe statistically significant differences in attack detection rate or comparison time between toggle and simultaneously visible configurations. For Vash, only the difference in comparison time between toggle and simultaneously visible configurations was statistically

significant (median time of 5.2 seconds when toggling, compared to 2.2 seconds when not; $p < .001$).

While it is not surprising that participants would take longer to compare fingerprints when they have to toggle between two views to perform that comparison, it is interesting to compare the interaction between visibility mode and representation. While participants did not take significantly longer comparing hexadecimal fingerprints when they had to toggle (median toggle: 9.8 s; both visible: 8.0 s), they did take significantly longer comparing Vash fingerprints when they had to toggle (median toggle: 5.2 s; both visible: 2.2 s). One explanation for this is that fingerprints based on abstract art are difficult to commit to memory, and so require more toggles (and thus more time) to compare than textual representations like hexadecimal.

Target Time-to-beat

We set the time-to-beat to 1080 seconds in one condition, which allowed approximately 36 seconds per task. All participants finished before the time-to-beat, with a median elapsed activity time (reflected in the on-screen stopwatch) of 716 seconds. The median comparison time for participants in the 2x time configuration was significantly different from those in the comparable configuration where the time-to-beat was 540 seconds (11.03 compared to 8.02 s; $p = .005$). The difference in the fraction of attacks missed was not significantly different between the 2x time configuration (10% missed) and the corresponding regular configuration (21% missed).

Comparison Strategies

Participants had a variety of strategies for comparing fingerprints. For textual fingerprints, participants often compared some subset of the beginning, middle, and end of fingerprints. Some participants also chose to compare random parts of the fingerprint, including one participant who was shown the hexadecimal representation, who said: "I first checked the last set of numbers, then randomly glanced at other sections until I felt I had verified enough values."

Other participants compared fingerprints in reading order, and instead focused on methods for efficiently doing so. For example, some participants had a strategy similar to the one described by a participant shown uppercase hexadecimal: "I would quickly read a segment and shift my eyes over as I repeated it, then compare and immediately/cross over into reading the next set from the other card to myself as I shifted back and compared to the next set of numbers, etc. - constant crossovers, but not having to cross over without going towards the next step to reduce time." Interestingly, one participant chose to compare hexadecimal fingerprints in reverse reading order: "I started with the rightmost set of numbers on the first line of the card, found one of the offered fingerprints that matched, then compared numbers back and forth in reverse reading order. I thought it would be easier to be accurate if using a technique that wouldn't make me fall into an easy 'reading' mode."

For all textual fingerprint representations besides sentences, fingerprints were presented in chunks of a fixed size, which provide a natural unit of comparison.[5] Indeed, many participants reported comparing fingerprints chunk-by-chunk. However, some participants chose to devise their own chunking strategy, a strategy prior work in the area of system-generated PINs has observed [12]. Many participants chose to compare multiple hexadecimal chunks at a time. Other reported units of comparison for textual representations included sentences, rows, and columns. Participants also described chunking strategies for graphical representations, such as comparing fingerprints by quadrant. Interestingly, more than one participant treated the visual host key representation more akin to textual representations, and compared in units of lines of text.

Although some participants compared graphical representations according to a chunking strategy, more commonly participants mentioned comparing particular features or characteristics of the specific fingerprints shown. One participant shown Vash said they tried "to pick out a couple things that might be different between pictures and then alternate between them to see if they are different." For Visual Host Key, one participant's strategy was to "look at general cues like the placement of the dots or big letters like B or S or E."

Participants strategies sometimes distinguished between the size of differences they looked for when performing comparisons, particularly for graphical representations. Some participants only checked for large differences. Others adopted a layered approach in which they looked for large differences first, followed by a search for more subtle differences. For example, one participant described this strategy for comparing Visual Host Key fingerprints: "The first thing I did was to try and glance at the whole fingerprint and see if anything jumped out at me. If I saw no difference by looking at the basic overview of it then I looked with a little more detail. If I was still unsure of its validity then I examined each line as quickly and accurately as possible."

DISCUSSION

Impact of Methodology

A main difference between our study and closely related work [6] was our focus on examining practical effects such as habituation, stress, and difficulties in comparison caused by variations in color, font, and position of the two fingerprints relative to each other. Based on both quantitative and qualitative data, we believe we successfully simulated some of these practical constraints, which we show do affect user behavior.

For example, in practice, attacks are likely to be few and far between. We simulated this by asking users to perform many comparisons of matching fingerprints before exposing them to an attack. Failing to find differences after several comparisions, most participants refined their strategy to compare only selected parts of fingerprints, which in turn increased the chance that they would miss attacks. Our experiment appeared to successfully simulate situations where users feel pressed

[5] For the sentence representation, individual sentences served a similar purpose, though the way we presented it made it difficult to immediately discern sentences as individual units.

for time. When asked what the hardest thing about the activity was, many participants responded that they felt stressed, pressured, distracted, or in a rush.

To verify that results weren't unduly driven by participants' need to finish tasks quickly, we included a condition in which participants had twice the time to complete the activity. In this condition, participants took only about 37% more time for each comparison, suggesting that time pressure was no longer a significant factor. These participants did make fewer mistakes (though the difference was not significant) but continued to miss attacks, suggesting both that time pressure in the study was at least partly successful at simulating real-life stress and that this stress was not the only factor which lead to mistakes.

Compare-and-select vs. Compare-and-confirm

We were surprised by how susceptible the compare-and-select method of verifying fingerprints was to attacks. Compare-and-select seeks to ease comparisons by offering users multiple options from which to select a fingerprint that matches another one that they are looking at. Over time, however, the compare-and-select approach appears to train users that one of the options is always correct (since some options always aren't). More specifically, at the end of the study we asked participants to report their concern about false negatives (saying that the fingerprints did not match when in fact they did) and false positives (saying that two fingerprints matched when they did not). While there was no significant difference in the rate at which compare-and-confirm and compare-and-select participants reported concern about false negatives ($p = .657$), compare-and-select participants were significantly less worried about false positives, i.e., failing to detect an attack ($p < .001$).

At the same time, while current implementations of compare-and-select appear to be a poor fit at least for fingerprint comparisions, there has been discussion of a compare-and-select approach where the options are chosen to be visually similar. A potential benefit of this approach is that users would be forced to focus on small details (since through a cursory comparison all options would look alike), leading to more effective security. A potential downside is the usability cost of performing detailed comparisions between all the options that need to be compared.

Desirable Properties and Tradeoffs

For both textual and graphical representations, participants struggled to decide how detailed a comparison to perform. For graphical representations, participants noted slight differences in color (as could potentially be caused by comparing a fingerprint printed on a business card to one on a computer screen) that caused uncertainties.

Participants shown graphical fingerprints tended to look at the big picture more often. While this is fine if small differences do not exist, it may be feasible for a determined attacker to find a key whose fingerprint is overall similar to the target fingerprint but different in small details, as was the case for our unicorn condition.

One advantage of textual formats like hexadecimal over image-based formats like Vash is that the former allow a user to know for certain whether two fingerprints are the same. For textual formats, a motivated user can check each digit and compare. For Vash, manual human comparison can only go so far; the user cannot confirm each pixel value through just visual inspection, and the representation does not convey what the smallest difference the user should look for is.

Another advantage of textual representations relates to the fact that they easily lend themself to being segmented into a particular structure, e.g., chunks of four characters, lines of text, etc. This structure seems to offer participants a useful reference point at semantically arbitrary locations within a fingerprint. Participants reported (unknowingly) taking advantage of this structure by making multiple detailed comparisons between various parts of corresponding fingerprints. This kind of behavior seems likely to make successful attacks less likely.

Recommendations

Overall, all the representations and configurations we experimented with exhibited higher rates of successful attack than seems desirable for high-risk situations. This strongly suggest that additional effort should be put towards removing the human in the loop, e.g., by using a smartphone camera to capture a printed fingerprint and having smartphone software make the comparison. When manual fingerprint comparison is necessary, the right choice likely depends on the context, since the different fingerprint representations we experimented with showed substantially different security and usability properties.

For all representations we tested, we observed participants making rational (if not always well informed) assumptions about how to go about comparing fingerprints. Graphical representations in general seemed to be more susceptible to comparison strategies that ignored fine details; at the same time, they allowed seemingly easy and quick comparisons. Consequently, unless the representation is accompanied by measures to help the user compare small details between two fingerprints, graphical representations appear not to be well suited for high-risk situations, but could be of benefit in low-risk environments, when attackers are not likely to be strong and usability is paramount.

When security is paramount, the best option is likely one we did not test: manually copying a printed fingerprint into a device and having software on that device make the comparison. This virtually eliminates the possibility of missing an attack, but at a high usability cost. For situations in which risk is not high and there is a need to balance security and usability, textual representations like hex (but also others like ASCII art and sentences) may be appropriate.

REFERENCES

1. akwizgran. 2014. Basic English: Encode random bitstrings as pseudo-random poems. (2014). https://github.com/akwizgran/basic-english

2. J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. 2007. OpenPGP message format. (2007). https://tools.ietf.org/html/rfc4880

3. Sandy Clark, Travis Goodspeed, Perry Metzger, Zachary Wasserman, Kevin Xu, and Matt Blaze. 2011. Why (special agent) Johnny (still) can't encrypt: A security analysis of the APCO Project 25 two-way radio system. In *Proceedings of the 20th USENIX Conference on Security (SEC'11)*. http://dl.acm.org/citation.cfm?id=2028067.2028071

4. Terrence Cole. 2011. Vash: Visually pleasing and distinct abstract art, generated uniquely for any input data. (2011). https://github.com/thevash/vash

5. Colin Davis. 2016. Robohash. (2016). https://robohash.org/

6. Sergej Dechand, Dominik Schürmann, Karoline Busse, Yasemin Acar, Sascha Fahl, and Matthew Smith. 2016. An empirical study of textual key-fingerprint representations. In *25th USENIX Security Symposium (USENIX Security 16)*. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/dechand

7. Michael Farb, Yue-Hsun Lin, Tiffany Hyun-Jin Kim, Jonathan McCune, and Adrian Perrig. 2013. SafeSlinger: Easy-to-use and secure public-key exchange. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking (MobiCom '13)*. DOI:http://dx.doi.org/10.1145/2500423.2500428

8. J. Galbraith and R. Thayer. 2006. The Secure Shell (SSH) public key file format. (2006). https://www.ietf.org/rfc/rfc4716.txt

9. Simson L. Garfinkel and Robert C. Miller. 2005. Johnny 2: A user test of key continuity management with S/MIME and Outlook Express. In *Proceedings of the 2005 Symposium on Usable Privacy and Security (SOUPS '05)*. DOI: http://dx.doi.org/10.1145/1073001.1073003

10. Shirley Gaw, Edward W. Felten, and Patricia Fernandez-Kelly. 2006. Secrecy, flagging, and paranoia: Adoption criteria in encrypted email. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. DOI: http://dx.doi.org/10.1145/1124772.1124862

11. Hsu-Chun Hsiao, Yue-Hsun Lin, Ahren Studer, Cassandra Studer, King-Hang Wang, Hiroaki Kikuchi, Adrian Perrig, Hung-Min Sun, and Bo-Yin Yang. 2009. A study of user-friendly hash comparison schemes. In *2009 Annual Computer Security Applications Conference*. DOI: http://dx.doi.org/10.1109/ACSAC.2009.20

12. Jun Ho Huh, Hyoungshick Kim, Rakesh B. Bobba, Masooda N. Bashir, and Konstantin Beznosov. 2015. On the memorability of system-generated PINs: Can chunking help?. In *Eleventh Symposium On Usable Privacy and Security (SOUPS '15)*. https://www.usenix.org/conference/soups2015/proceedings/presentation/huh

13. Antti Huima. 2000. The Bubble Babble binary data encoding. (2000). http://web.mit.edu/kenta/www/one/bubblebabble/spec/jrtrjwzi/draft-huima-01.txt

14. Ronald Kainda, Ivan Flechais, and A. W. Roscoe. 2009. Usability and security of out-of-band channels in secure device pairing protocols. In *Proceedings of the 5th Symposium on Usable Privacy and Security (SOUPS '09)*. DOI:http://dx.doi.org/10.1145/1572532.1572547

15. Ruogu Kang, Stephanie Brown, Laura Dabbish, and Sara Kiesler. 2014. Privacy attitudes of Mechanical Turk workers and the U.S. public. In *Proceedings of the Tenth Symposium on Usable Privacy and Security (SOUPS '14)*. https://www.usenix.org/conference/soups2014/proceedings/presentation/kang

16. Alfred Kobsa, Rahim Sonawalla, Gene Tsudik, Ersin Uzun, and Yang Wang. 2009. Serial hook-ups: A comparative usability study of secure device pairing methods. In *Proceedings of the 5th Symposium on Usable Privacy and Security (SOUPS '09)*. DOI: http://dx.doi.org/10.1145/1572532.1572546

17. Arun Kumar, Nitesh Saxena, Gene Tsudik, and Ersin Uzun. 2009. Caveat eptor: A comparative study of secure device pairing methods. In *2009 IEEE International Conference on Pervasive Computing and Communications*. DOI: http://dx.doi.org/10.1109/PERCOM.2009.4912753

18. Raph Levien and Donald Johnson. 1998. Snowflake. (1998). http://dlakwi.net/snowflake/snowflake.html

19. Dirk Loss, Tobias Limmer, and Alexander von Gernler. 2009. The drunken bishop: An analysis of the OpenSSH fingerprint visualization algorithm. (2009). http://dirk-loss.de/sshvis/drunken_bishop.pdf

20. Skylar Nagao. 2016. Avatars. (Oct 2016). https://peerio.zendesk.com/hc/en-us/articles/202729949-Avatars

21. Off-the-Record Messaging. 2016. Fingerprints. (2016). https://otr.cypherpunks.ca/help/fingerprint.php

22. OpenSSH. 2008. OpenSSH 5.1 release announcement. (2008). https://www.openssh.com/txt/release-5.1

23. Adrian Perrig and Dawn Song. 1999. Hash visualization: A new technique to improve real-world security. (1999). https://users.ece.cmu.edu/~adrian/projects/validation/

24. Plasmoid. 2003. Fuzzy fingerprints: Attacking vulnerabilities in the human brain. (2003). https://www.thc.org/papers/ffp.pdf

25. David Roundy. 2014. Visual hash. (2014). http://visual-hash.readthedocs.io/en/latest/

26. Scott Ruoti, Nathan Kim, Ben Burgon, Timothy van der Horst, and Kent Seamons. 2013. Confused Johnny: When automatic encryption leads to confusion and mistakes. In *Proceedings of the Ninth Symposium on Usable Privacy and Security (SOUPS '13)*. DOI: http://dx.doi.org/10.1145/2501604.2501609

27. Maliheh Shirvanian and Nitesh Saxena. 2015. On the security and usability of crypto phones. In *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC 2015)*. DOI: http://dx.doi.org/10.1145/2818000.2818007

28. Ryan Stedman, Kayo Yoshida, and Ian Goldberg. 2008. A user study of Off-the-Record Messaging. In *Proceedings of the 4th Symposium on Usable Privacy and Security (SOUPS '08)*. DOI: http://dx.doi.org/10.1145/1408664.1408678

29. Marc Stevens, Pierre Karpman, and Thomas Peyrin. 2016. Freestart collision for full SHA-1. In *Proceedings of the 35th Annual International Conference on Advances in Cryptology (EUROCRYPT 2016)*. DOI: http://dx.doi.org/10.1007/978-3-662-49890-3_18

30. Zettabyte Storage. 2011. Vash: The visual hash. (2011). https://web.archive.org/web/20130127121849/http://www.thevash.com

31. Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. 2015. SoK: Secure messaging. In *Proceedings of the 2015 IEEE Symposium on Security and Privacy (SP '15)*. DOI:http://dx.doi.org/10.1109/SP.2015.22

32. Ben Dumke v. d. Ehe. 2012. Unicornify! How does it work? (2012). https://unicornify.appspot.com/making-of

33. Serge Vaudenay. 2005. Secure communications over insecure channels based on Short Authenticated Strings. In *Proceedings of the 25th Annual International Conference on Advances in Cryptology (CRYPTO'05)*. DOI:http://dx.doi.org/10.1007/11535218_19

34. WhatsApp. 2016. WhatsApp encryption overview: Technical white paper. (April 2016). https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf

35. Alma Whitten and J. D. Tygar. 1999. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proceedings of the 8th Conference on USENIX Security Symposium (SSYM'99)*. http://dl.acm.org/citation.cfm?id=1251421.1251435

36. Wickr. 2016. What is the key verification feature? (2016). https://wickr.desk.com/customer/en/portal/articles/2342342-what-is-the-key-verification-feature-

37. Min Wu, Robert C. Miller, and Greg Little. 2006. Web Wallet: Preventing phishing attacks by revealing user intentions. In *Proceedings of the Second Symposium on Usable Privacy and Security (SOUPS '06)*. DOI: http://dx.doi.org/10.1145/1143120.1143133