Consonant-Vowel-Consonants for Error-Free Code Entry

Nikola K. Blanchard[1(✉)], Leila Gabasova[2], and Ted Selker[3]

[1] Institut de Recherche en Informatique Fondamentale,
Université Paris Diderot, Paris, France
nikola.k.blanchard@gmail.com
[2] Institut de Planétologie et d'Astrophysique de Grenoble,
Saint-Martin-d'Héres, France
[3] University of Maryland, Baltimore County, USA
http://www.koliaza.com

Abstract. Codes and passwords are the bane of user experiences: even small mistakes can delay desired activities, causing undue frustration. Work on codes has focused on security instead of people's ability to enter them error-free. Difficulties observed in a security demonstration motivated this investigation of code transcription difficulty. A pilot study with 33 subjects and a follow-up study with 267 subjects from 24 countries measured performance and preference for codes of varying lengths, patterns, and character sets.

We found that, for users of all languages, long codes with alternating consonant - vowel patterns were more accurately transcribed and are preferred over shorter numeric or alphabetic codes. Mixed-case and alphanumeric character sets both increased transcription errors.

The proposed CVC^6 code design composed of six Consonant-Vowel-Consonant trigrams is faster to enter, more secure, preferred by users, and more impervious to user error when compared to codes typically used for security purposes. An extension integrates error detection and correction, essentially eliminating typos.

Keywords: Usable-security · Error correcting codes · Authentication · User study

1 Introduction

People all have different codes for their driver's license, social security, government ID, and bank accounts, and passwords to access email, social media, and each online transaction or community system they use. These codes are now central to protecting our identities. Passwords are codes used in conjunction with logins to authenticate users.

The most frequent answer to our increasing security needs [17,31] has been to add more passwords, increase length and character complexity with

A. Moallem (Ed.): HCII 2019, LNCS 11594, pp. 19–37, 2019.
https://doi.org/10.1007/978-3-030-22351-9_2

upper- and lower-case characters and special characters [8], and change them frequently, counter-productively making them even harder to remember [10].

Biometrics have been considered a candidate solution to those problems for a long time, but finger prints, iris detection, and face recognition systems have all been shown to be hackable [9,22,27]. Passwords have the advantage of being pure information, hence easier to create and share with a trusted party, but they have become an arms race with no predictable outcome [21].

We are required to come up with and/or enter codes with a variety of patterns, from copying credit card numbers to Wi-Fi codes to account passwords. Much progress has been presented on usable security, to study the perception of complexity and counter the failures of user-created passwords [13,24,29,30]. Yan's paper [32] was an important first exploration into password memorability. It showed that mnemonics can help memorability of passwords [3] while not compromising security strength, leading to additional work in that direction [14,18,20,26].

But how well do people succeed when using codes and how can their success be improved? We all enter codes many times a day, typing our name and well-practised passwords much faster than new character strings (especially automatically generated ones). Many of us forget or misenter codes while needing to get access to our resources [7].

Creating codes adapted to their use, whether it is memorability, ease of entry, or speed can greatly reduce stress and breakage in everyone's work. Trade-offs are inherent in privacy and security [1], and a single compromised password can be catastrophic [15]. Typical systems might require people to use at least 8 characters, upper- and lower-case, numbers, and special characters, although some have started questioning if people gain actual security with the added complexity [10,11]. In one illustrative example, frustration and confusion with character recognition in a code-based voting system caused at least 10% additional abstention [5].

User-created codes versus single-use or automatically generated codes present very different challenges for usability. Automatically-generated codes depend on multiple frequent assumptions that haven't been extensively questioned. Does increased character set complexity [25] make better codes? Does separation of a code into multiple fragments (in this paper called 'chunking' a code) with spaces in between reduce errors? Are nonsense patterns of alphanumerics for security better than syllabic codes or even words? This paper tests usability and security together for codes that are not user-created.

Are there trade-offs between length, character sets and structural patterns that can improve people's ability to use codes? How do these trade-offs change when considering the ability to reduce transcription errors for one-time codes? Can techniques for making codes easier to enter work across cultures or even languages?

The rest of the paper is organised as follows. After presenting the main results, we introduce the experimental protocol for the two crowd-sourced tests of usability and transcribability via a web-based approach. Experimental Results presents data from the pilot and main experiments, showing links between length and type of code trade-offs. The implications of these findings are developed. Inspired by the results, the paper then introduces CVC^6, a 6-trigram code design for higher-entropy higher-usability codes always composed of 6 consonant-vowel-consonant trigrams. An extension of CVC^6 is also presented that includes error detection/correction, CVC^{6++}. The paper concludes by showing how more work could be done to further explore the design of cross-cultural, easy-to-transcribe, high-entropy codes everyone finds themselves using several times a day.

1.1 Definitions

The experiments included randomly-generated codes composed of sequences of the following type:

- numeric: numbers from 0 to 9.
- alphabetic: lower-case Latin letters (excluding diacritics).
- alphanumeric: numbers, lower-case, and upper-case alphabetic characters, containing at least one of each.
- CVCs: consonant-vowel-consonant alphabetic trigrams in lower-case. Vowels are a, i, e, o, u and y. Consonants go from b to z, excluding y as well as q due to demonstrated discrimination problems between y, q and g.

2 Main Results

This paper has 4 main experimental observations, and one theoretical contribution.

- Transcribing codes takes concentration and is highly dependent on the code's structure. This work found that, for a given length, code structure can reduce transcription error rates from 16.9% to 1.9%.
- A majority of code transcription errors can be eliminated by using a set of unambiguous alphabetic characters (excluding visually ambiguous g/q/y and i/l), eliminating mixed case to prevent upper-case/lower-case confusions, and eliminating numbers.
- The relationship between code length and time needed to enter it strongly depends on the code's structure; spaces can help for long alphanumeric codes but can be confusing for others. Using a consonant-vowel-consonant (CVC) pattern in codes can reduce time to transcribe even with codes twice as long.
- People have a 75% chance of recognising a code they had seen 2 to 5 min earlier. However, they will correctly reject a novel code they haven't seen in 87% of cases.

Based on these findings, the protocol, CVC^6 is proposed that is easier and faster to transcribe, with fewer mistakes and increased security. We also introduce CVC^{6++}, an extension that includes error detection and correction.

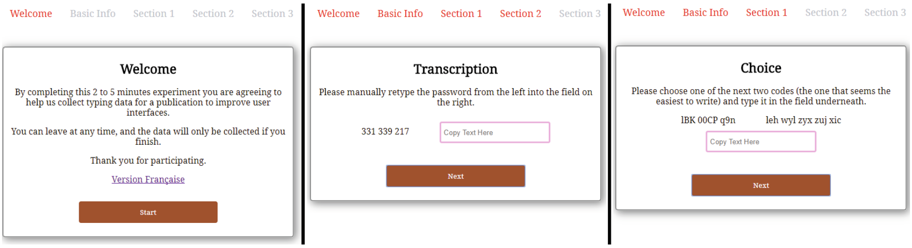
3 Experiment Design

The following experiments have the goal of demonstrating trade-offs between character sets, number of characters, and patterns of characters to create easy-to-enter secure codes. A web-based interface was developed in Javascript to sequentially present discrete code transcription problems. It was iteratively tested in a pilot experiment and then improved and extended for a main experiment. The goal was to have people type codes in the kinds of places they typically are when using online services. To understand how codes can be improved in the wild, experiments were conducted wherever a person was (real-life conditions, not a laboratory environment). Our analyses generally avoid raw averages and focus on trimmed averages and medians to eliminate anomalies (such as one participant taking close to 5 h to answer a single question).

3.1 Pilot Experiment

A protocol was developed that would take no more than a few minutes and test transcription of code length, character sets, and spaces. Engagement was initially solicited personally by a docent from 33 random attendants of the science fiction conference Worldcon 75 in August 2017 for a pilot experiment. The initial design did not adequately distinguish capitalisation problems and issues around the way input is entered on smart phones. Unfortunately it also didn't correctly disable auto-correct. Despite those setbacks, the data still showed that codes following syllabic patterns had many advantages and laid the groundwork for changes to put into the experiment (Fig. 1).

While several of the pilot experiment's results were statistically significant, the main experiment corroborates and extends these on a larger and more diverse sample. The pilot helped validate and improve the Javascript protocol and show where more data was needed. Results below detail only the main experiment (data will be available for both studies in a public online repository).



Welcome

By completing this 2 to 5 minutes experiment you are agreeing to help us collect typing data for a publication to improve user interfaces.

You can leave at any time, and the data will only be collected if you finish.

Thank you for participating.

Transcription

Please manually retype the password from the left into the field on the right.

331 339 217

Choice

Please choose one of the next two codes (the one that seems the easiest to write) and type it in the field underneath.

lBK 00CP q9n   leh wyl zyx zuj xic

Fig. 1. Screenshots from the experiment's interface

3.2 General Protocol

Participants were individuals that responded to an opportunity to volunteer online. They were told that they could quit the experiment at any time. Their data was only collected (through FormSpree) if they confirmed submission at the end. The total time taken generally varied from 3 to 10 min.

For security as well as privacy, all code executed was on the user's device and visible to the user, and only recorded their final answers and timestamps.

The study was presented as a sequence of web forms with an introduction and three main sections, designed to measure transcription performance, preferences between different kinds of codes, and ability to remember the codes shown in the first section.

Sections

- Welcome and basic respondent information
- Transcription: Nine codes of different length to transcribe into a prompt
- Choice: Nine pairs of codes varying from 9 to 22 characters in length and type (alphabetic, alphanumeric, and Consonant-Vowel-Consonant (CVC)) where the participant was asked to choose and transcribe the easiest.
- Memory: Participants were presented with seven codes and asked if they had seen this code earlier. They were also asked to give an estimate of the number of codes they had transcribed.
- Accept: Participants were asked to upload their answers and given the choice of adding their email to be kept informed.

Introduction and Basic Information. The welcome page presented the experiment as an opportunity to help research, and informed participants that they could leave at any time. It took care not to prime them with research goals. It asked their age, country, main language used, self-rating of their ability to remember passwords and strings of numbers/letters, as well as whether they were using a mobile device or a numeric keypad in the experiment. Optionally, participants could submit their email to receive the experiment results once published. Those emails were stored securely and separately from the data that was analysed. Participants were not asked their gender or other personal characteristics as they were not pertinent to the research.

Transcription. The codes were grouped by length (9, 12, and 15 characters), each group presenting three types of code trials in the following order: numeric, CVC, and alphanumeric. This gave a baseline error rate from which to replicate standard findings [12,16] such as the prevalence of the g/q/y error. It also gave rates for other types of errors, allowing the comparison of different code structures. Participants were not informed of errors they had committed and had a single try for each code.

Choice. Each trial included choosing to type in a 10 alphanumeric control code or a second code. The codes were grouped by character sets used – 3 using numeric, 3 CVCs, and finally 3 alphabetic. Trials were given in order of increasing length for each type.

Memory. Every question included a code participants had seen earlier, or a randomly generated code of the same type and length, with probability 0.5 for each. The types were numeric of length 9, and CVCs and alphanumeric of length 9, 12, and 15.

Randomisation. A between-subjects approach was used to observe priming and learning. Half the participants did the Choice section first, and the other half started with the Transcription. Half the participants also received the Transcription questions in reversed order.

For the Choice section, the order in which the two codes were presented was also randomised to avoid preference for the one on the left or right. The 9 codes in the Choice section were presented in order of increasing length. The pilot experiment seemed to indicate a tipping point close to 18, so we bracketed by testing codes of length going from 15 to 22. As this phase was already the most time-consuming for the participant, having one code for each length would have made the experiment too long. Hence, it was broken up in A/B testing giving participants two codes of length respectively 15 and 20, and one of random length between 16 and 22.

Times. Time taken was measured for each question, as well as the time spent reading the different sets of instructions. As the protocol was self-administered and self-paced, some people took breaks, ranging from a few minutes to five hours. Large delays on a single question were observed in around 15% of respondents. Taking breaks or getting distracted is part of life; long breaks alone did not disqualify all trials from analysis. Data for each question was independently evaluated and the abnormally short and abnormally long responses were removed (top and bottom 10%). Medians were consistently 5–10% under the trimmed averages and are not shown as they lead to the same conclusions.

Chunking. The Transcription section codes were split into "chunks" of 3, 4, or 5 characters followed by a space. In the Choice section, chunks of 3 were used. For lengths not divisible by 3, the last chunk had between 2 and 4 characters, and the 10-character alphanumeric codes avoided a 1-character last chunk by using a 4-character central chunk.

3.3 Demographics

The main experiment included 267 respondents, with some skipping a few questions[1]. Participants were solicited for the main experiment using three methods, creating three groups. The web links followed to get to the experiment identified which group a participant was in. The first group was international in scope, spread through Facebook and totalled 61 respondents.

The second was mostly French, using a translated form, and was composed in majority of software developers, as it was spread through a French computer engineering school's social network and Internet Relay Chat, with 91 respondents. Members of this group were highly tech-savvy compared to the other two groups (due to how they were recruited).

The third was overwhelmingly composed of people from the USA, with 115 respondents recruited through a website indexing psychological and social experiments often used by college students (http://psych.hanover.edu/research/exponnet.html).

All three groups included a wide range of ages, with the youngest being 19 years old for the pilot and 13 for the main experiment[2]. The eldest were respectively 70 and 73 years old, with most participants between 18 and 32.

People from 24 different countries and speaking 14 languages participated, including a few who were used to scripts written from right to left. English was the most frequent language indicated (129 people), with French second (114 people), and 34 participants indicating other main languages.

The goal in this recruitment method was to avoid having anomalies coming from a bias stemming from a single recruitment process. The results shown are only the ones that are consistent among all groups.

4 Main Experimental Data

4.1 Error Typology

The first section acted as a control to get a transcribing performance baseline, and allowed different patterns in transcribing behaviour to be observed. The following Fig. 2 shows the different error types observed in both sections – which differ as the text to transcribe varies. Underneath are the definitions for the error types.

- *Missing/added char*: a single character is either missing or was duplicated, which changes the length of the code.
- *Similarity*: confusion due to the similar shape of two characters, most commonly where one writes 0 instead of O, g instead of q or y (mostly present in the pilot), or confuses I with l and 1.

[1] This accounts for less than 3% of questions and is generally caused by a double-click on the "next" button, as timestamps show the participants spending a few hundred milliseconds on a page.

[2] The three participants who were younger than 16 all came through the psychological study website.

- *Transposition*: the order of two characters was reversed.
- *Adjacent key*: a key next to the target was hit, such as g instead of h. This mostly happens with horizontally adjacent keys.
- *Capitalisation*: an upper-case letter is written in lower-case, or vice-versa – this nearly only happens with alphanumeric codes.
- *Autocorrect*: despite our disabling of autocorrect via JavaScript, 2% of participants showed repeated revealing mistakes where whole words were changed.

4.2 Transcription Trial

Figure 3 shows the error rates for each code (structure/length) couple, for each group. Figure 4 shows the time taken (trimmed average) for those.

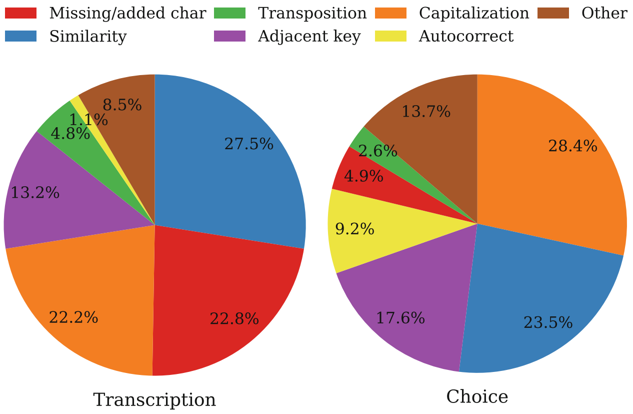


Fig. 2. Proportion of error types in each trial

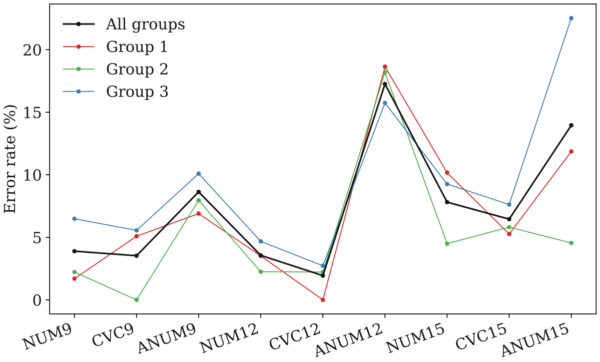


Fig. 3. Error rate, by code type and length

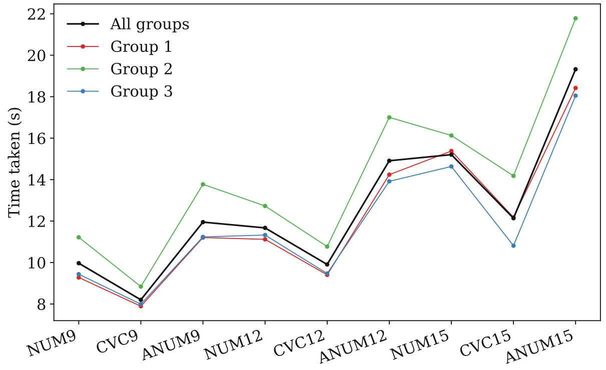


Fig. 4. Time taken to transcribe, by code type and length

4.3 Choice Trial

Figure 5 shows the proportion of people who chose to transcribe different code structures of varying lengths over a 10-character alphanumeric string.

Figure 6 shows the time taken for each structure by length, and the average time taken by the people who chose the 10-character alphanumeric.

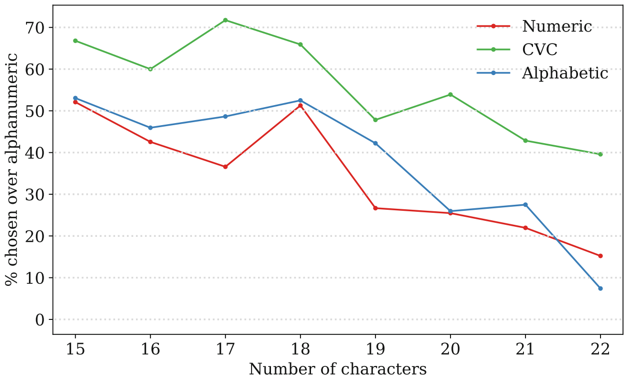


Fig. 5. Percentage of participants preferring alternative codes to 10-character alphanumeric ones, by code type and length

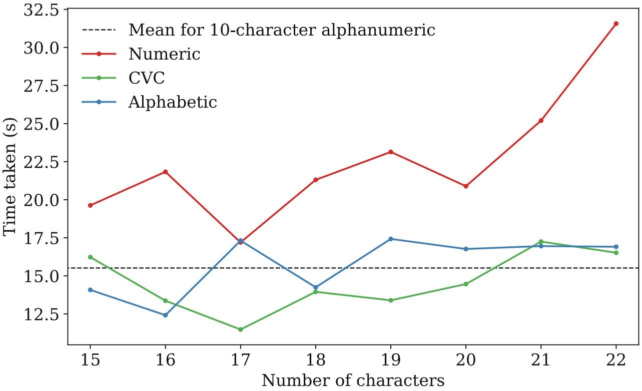


Fig. 6. Time taken by code length for each code type, Choice section

4.4 Strategies

Many people appeared to follow a strategy to choose which code to transcribe. Across 267 participants we identified 121 different patterns, of which here are the 5 most common ones (accounting for more than a third of participants):

- 31 people always chose the alphanumeric.
- 24 people chose the alphanumeric for all cases but one (either short or mid-length CVCs or numeric).
- 18 people only chose the alphanumeric against numeric codes.
- 12 people only chose the alphanumeric in one case.
- 11 people never chose the alphanumeric.

4.5 Memory

The ability to recognise codes that people had seen once in the past few minutes was 75%. People were also good at discarding codes they had not previously seen, with 87% success rate. This converts to a false negative rate of 25%, and false positive rate of 13%. The following table shows the error rate for each type of memory question:

NUM9	CVC9	CVC12	CVC15	ANUM9	ANUM12	ANUM15
22.5	28.8	8.6	14.4	29.2	12.0	16.7

The answer to the question asking them to estimate the number of codes they had written was relatively precise when we look at the trimmed average (18.7 for a true value of 18), but not with a simple average or a median (both at 20.0–20.1), because of a large variance, a strong tendency to write 20 (more than a quarter of participants) and the 8% of people who overestimated by a factor between 2 and 6.

5 Analysis

Here are the main effects observed:

- CVC codes were less error-prone than alphanumeric ones for all lengths ($p < 0.005$).
- Participants preferred CVCs of length at most 20 over 10-character alphanumeric codes, with rates varying between 72% and 48% in the worst case ($p < 10^{-4}$).
- 10-character alphanumeric codes were preferred to alphabetic and numeric codes of lengths greater than 19 ($p < 10^{-4}$). Only 7.4% chose the alphabetic code of length 22 over the alphanumeric alternative. They were preferred or equivalent for shorter lengths (with at most 53% choosing alphabetic codes over alphanumeric ones).
- For each length, CVCs were faster to type than numeric. Those were in turn faster to type than alphanumeric ($p < 0.05$ to $p < 10^{-4}$ depending on the couple). The speed increased by up to 59% for CVCs as opposed to alphanumeric.
- When presented with chunked codes (with spaces between groups of characters), 91% of participants wrote the spaces in the codes they typed.
- Chunked codes were faster to enter ($p < 0.015$) by an average of 8% (with a maximum of 14%).
- Chunking in three-character groups only statistically lowered the error rate for alphanumeric codes ($p = 0.033$).
- People were better at rejecting codes they hadn't seen than at confirming that they'd seen a specific code, ($p < 10^{-4}$, the false negative rate was more than twice the false positive rate).
- The typing speed and the error rate were not statistically correlated (both when considered by participant and by individual code).

Other significant effects presented themselves as well:

- There was no statistically significant difference on error rates between numeric and CVC codes.
- A great variability in typing speed was observed, with 20% of people typing above 1.34 characters per second, and 20% typing below 0.75 c/s (within the normal bounds for non-professional typists [19]). The top 5% entered codes more than three times faster than the bottom 5%.
- Recognition ability was correlated with self-rating in the memory section. Two cohesive clusters appeared, one around 28% error rate for people who rated their memory 1 or 2, and one between 16% and 18% for those who rated it higher ($p < 10^{-4}$).
- A learning effect was observed ($p < 10^{-4}$), with people reaching up to 18% higher speeds by the time they finished the transcription section. A/B testing compensated for this, making its effects negligible in other results.
- There was a recognition peak around length 12 for both CVCs and alphanumeric codes, strongly reducing both false positive and false negative rates ($p < 10^{-4}$).

6 Discussion

Pronounceable codes such as CVC are faster to type and more accurate than random alphanumeric ones. The crucial point is that the magnitude of the effect is such that it renders longer codes a viable alternative, even in contexts where security is the objective.

11 of the 31 errors present in the transcription phase of CVC were preventable by checking the length. An additional 4 could be automatically fixed by checking whether the letter typed was a vowel or a consonant. Among the 106 errors found in alphanumeric codes, only 7 were preventable in such ways. This motivated the development of CVC^6 below.

Chunking the codes in groups of 3 characters only reduced errors for alphanumeric codes (numeric codes seemed to benefit from chunking, although not enough for statistical significance). This might be explained from people's instinctive chunking of CVCs even without spaces. There was some confusion on whether to add the spaces between the chunks, but despite this and the added characters, the speed still improved overall for all codes.

Directly analysing error rates and speeds is difficult in the Choice section as they depended on the participants' strategies. Depending on the choice they faced, the average time taken by people who chose the alphanumeric code varied between 13.5 and 20.8 s. Presenting them with long numeric codes did slow them by up to 7 s, even for the people who ignored those long codes.

Memory was strongly influenced by length. The structure of the code did not visibly affect its memorability. Simple considerations of ability to discern two codes and memorability of long codes seem insufficient to explain a recognition peak at length 12 as they should differently affect false positive and false negative rates. When asked to estimate the number of questions, there was a tendency to answer with multiples of 5, in 76% of participants.

The three groups, with their different demographics and methods of recruitment, showed some variations in their performances. However, all the effects mentioned so far are observed not only in the general data, but also within each group, increasing their ecological validity as they do not depend on recruitment peculiarities. The most salient difference was that group 2 took more time but made fewer errors than the other groups. This could come from a variety of things such as their supposedly higher technical expertise (being mostly computer engineering students) or different keyboard layouts. The effect is also observed when we cluster by language (although the overlap is big between those two clustering methods).

7 CVC^6

The goal was to design a code that is easier and faster to enter, as well as more secure. CVC^6 codes are composed of 6 CVC trigrams, as in the following example:

cab dij kap pod myn ret

7.1 Advantages

From a security standpoint (for use as passwords), CVC^6 has high entropy, with 1.03×10^{20} total possibilities, or 66.5 bits of entropy. This is following Kerckhoff's principle with the adversary knowing the format of the code used (against a blind brute-force, it would instead correspond to 2.95×10^{25} possibilities or 86 bits). Current standards for passwords are between 8 and 10-character alphanumeric codes, which are not necessarily randomly generated. Those have at most 48 and 59.5 bits of entropy, meaning that CVC^6 takes at least 100 times more effort to brute-force, assuming the adversary already knows which system is used.

Nearly two thirds (66%) of the study's participants perceived CVC^6 as easier than both equivalent alphanumeric and alphabetic codes.

Despite its length, CVC^6 is faster to type than other codes of similar or lower entropy. In the Choice section, CVC codes demonstrated average and median speeds higher by 10% to 80% compared to equivalent code structures. This is despite entropies being as low as 59.5 bits for alphanumeric and 50 bits for numeric (the trade-off meaning that a lower entropy generally implies a faster typing speed).

The error rate is already more than a third lower in CVCs than in comparable codes, but this can be improved even further. CVC^6 can get under 5% error by eliminating the following sources of error:

- Capitalisation errors, as the code isn't case-sensitive
- Symbol confusions, which would almost entirely disappear, leaving only v/w (which is very rare)
- Thanks to the alternating consonant and vowel pattern, character deletion and transposition would be immediately detectable by the system and visible to the user. This would also apply to 10% of near misses.

This can be additionally improved by handling error correction, shown below with the improved CVC^{6++} approach.

8 CVC^{6++}

Getting an error when typing in a code frustrates most users, and not being able to find its location even drives some to abandon whatever task was at hand.

One improvement of considerable value would then be for the system to detect an error and point it out, possibly indicating what the error was. This would eliminate mistakes CVC^6 is vulnerable to (mostly near misses and phonetically similar characters). It would have eliminated all of the 495 errors in this paper's experiments. Only double or triple errors wouldn't be corrected, and the three double errors observed in the transcribed CVC codes would have been detected correctly. Error detection, localisation, and/or correction would reduce user confusion and input time. A natural extension to CVC^6 achieves all of those.

8.1 Protocol

The extended error detection/correction protocol is shown on Fig. 7 and works as follows:

- To add correction without compromising on entropy, one last chunk "YZ" of two consonants after the last trigram is added to the code.
- To detect, localise, and correct the error:
  - Values from 0 to 18 are assigned to each consonant: b = 0, c = 1, d = 2 etc. Since consonants and vowels are not used in the same position, numbers can be reused by vowels: a = 0, e = 1, i = 2, o = 3, u = 4.
  - Y is computed by summing all the values modulo 19.
  - Z is computed by summing all the values, multiplied by their position in the code, modulo 19.

Suppose that there is an error concerning a single character in position $i \leq 18$ (i.e. the error is not on Y or Z). If the value of the entered Y differs from the sum computed from the input, the error is detected. $d \times i \mod 19$ is the difference between the computed Z and the Z' entered with an error. The difference d between the character entered and the correct one is also calculated. The combination of those two directly shows the unique possibility for a single-character error. This is where having a base 19 system is crucial, as only prime bases allow this (as the multiplication modulo 19 is bijective). In the case where the single error concerns Y or Z, the other one is correct, which cannot happen in normal cases, so the system knows that the error concerns either Y or Z (and can ignore it).

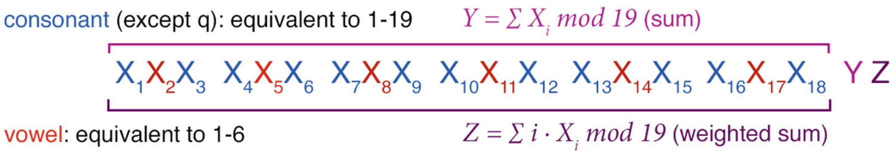

consonant (except q): equivalent to 1-19
$Y = \Sigma X_i \; mod \; 19$ (sum)
$X_1X_2X_3 \; X_4X_5X_6 \; X_7X_8X_9 \; X_{10}X_{11}X_{12} \; X_{13}X_{14}X_{15} \; X_{16}X_{17}X_{18}$ Y Z
vowel: equivalent to 1-6
$Z = \Sigma i \cdot X_i \; mod \; 19$ (weighted sum)

Fig. 7. Error correction in CVC^{6++}

8.2 Advantages and Limitations

The obvious advantage of CVC^{6++} is that it allows the system to automatically identify/correct the code and avoid wasting the time of a frustrated user. This automatic correction should not be used where correcting a double error into a different code would be strongly detrimental, such as voting. Instead, the system could indicate the location of the error to the user, to allow them to quickly check and correct it themselves.

The second advantage lies in its use in conjunction with cryptographic electronic voting, where one person can vote by proxy by giving a code to a trusted third party. This third party does not have access to the list of valid codes and would normally have no way to notice if something went wrong during the transmission of the code. As they have an obligation to make a selection, the only solution to an error would be a system that would allow them to quickly detect this error. A public website checking valid CVC^{6++} is available at www.koliaza.com/CVC.

The main limitation of CVC^{6++} is that it cannot work as naturally for CVCs longer than 6 trigrams, as multiple conflicting correction possibilities would appear. Probabilistic error correction could solve this, or an extension of the last two letters to a larger character-set.

Its length (for the same level of security) would also make it less popular than CVC^{6}, although a majority in our experiments would still prefer it to alphanumeric codes.

9 Conclusion

This paper explores how transcription of codes such as passwords can be affected by length, character sets, and structure. Results come from an experiment involving 267 subjects from 24 countries. The online experiments showed how large improvements to speed, transcription, and memorability can be made without compromising security for usable codes.

The value of generating passwords in general is discussed in the introduction but the original motivation behind this paper came from problems in electronic voting experiments which use automatically generated passwords. The results of this work are already helping in the ongoing voting technology experiments.

Discrete transcription trials showed that, as they are often found in words, codes based on CVC trigrams are preferred, faster, and less error-prone than alphanumeric, alphabetic, or numeric codes.

Most errors came from a few easily identifiable factors. Ambiguous shapes such as 0 and O, g, y and q, or l, i and 1 account for more than a quarter of errors. Along with wrong capitalisation, they explain why standard alphanumeric codes have much higher error rates than ones using simpler character sets. Moreover, when compared to language-like codes, they are much slower to enter, more than offsetting their increased security per character.

Although codes with simple syllabic patterns had better performance on all fronts, care has to be taken to prevent phonetic errors, and to avoid disadvantaging certain cultures in which some syllabic patterns are absent. This is especially important for codes used by diverse groups and in critical activities such as voting.

As a large majority of errors could be prevented by a simple pattern, a single length, and unambiguous characters, we propose a protocol, CVC^6, that is easier and faster to transcribe, with fewer mistakes and increased security. We also introduce CVC^{6++}, an extension that includes error detection/correction. Such codes could have wide-ranging applications, from voting technology to more accessible routers.

Finally, the memorability of codes was shown to depend strongly on pattern and length, albeit not in a trivial way. Subjects had a 75% chance of recognising a code they had seen 2 to 5 min earlier but correctly rejected a code they hadn't seen in 87% of cases.

We are hopeful that the increased reliability and usability of code-creation methods described here, together with new evaluation metrics for usable security [10], can help users create much more effective passwords and other codes, for improved security and usability.

9.1 Future Work

This study raises new questions on transcribing ability and code structure. Interesting follow-up experiments could be motivated by the following questions:

- Fonts have been shown to strongly impact reading ability [4]. What is the impact of font, spacing, and case on codes?
- Is there a cost associated with not typing spaces? Is the speed increase for chunking hampered by having to enter an extra space character? Why doesn't it increase transcribing ability? How would chunked input zones affect it?
- Other surface features also have important effects on memory and language learning [28]. How would the colour and texture coding of chunks affect transcribing ability?
- Different syllabic patterns, such as CCVC or CVCC, have higher entropy, but are less frequent and even absent in certain languages [2,6,23]. Could they constitute viable alternatives to CVC and would they be less language-dependent? Even further, could chunks made of real words be used, and would they be worth the entropy loss for English speakers?
- Some letters (like q or x) being less frequent in many languages, would transcribing ability increase with an even smaller alphabet? Could this compensate the entropy loss?
- The memory performance measured purposefully avoided tricky codes that were close to ones the subject had seen. What makes codes distinguishable? For goals of privacy, can easily transcribable but not memorable codes be formulated?

- The different error patterns shown are quite predictable, and could potentially be used for a CAPTCHA system where the error would be human. Could one game such a system?
- What is the impact of differences in typing ability among people who are used to a different alphabet (such as Ge'ez, Hiragana, or Cyrillic), non-alphabetic languages (Mandarin Chinese) or right-to-left writing systems?

This work also shows that new metrics might be needed to correctly analyse the benefits of code structure, depending on the application. Such metrics would need to include memorability, error probability and effect in case of error, typing speed, perceived ease, and cultural dependency.

The authors would like to thank Florentin Waligorski for his help with data analysis.

References

1. Acquisti, A., et al.: Nudges for privacy and security: understanding and assisting users choices online. ACM Comput. Surv. **50**(3), 1–41 (2017)
2. Adsett, C.R., Marchand, Y.: Syllabic complexity: a computational evaluation of nine European languages. J. Quant. Linguist. **17**(4), 269–290 (2010). https://doi.org/10.1080/09296174.2010.512161
3. Bellezza, F.S.: Mnemonic devices and memory schemas. In: McDaniel, M.A., Pressley, M. (eds.) Imagery and Related Mnemonic Processes, pp. 34–55. Springer, New York (1987). https://doi.org/10.1007/978-1-4612-4676-3_2
4. Bernard, M., Liao, C.H., Mills, M.: The effects of font type and size on the legibility and reading time of online text by older adults. In: CHI 2001 Extended Abstracts on Human Factors in Computing Systems, CHI EA 2001, pp. 175–176. ACM, New York (2001). http://doi.acm.org/10.1145/634067.634173
5. Blanchard, N.K.: Building trust for sample voting. International Journal of Decision Support System Technology (2018)
6. Borleffs, E., Maassen, B.A.M., Lyytinen, H., Zwarts, F.: Measuring orthographic transparency and morphological-syllabic complexity in alphabetic orthographies: a narrative review. Read. Writ. **30**(8), 1617–1638 (2017). https://doi.org/10.1007/s11145-017-9741-5
7. Brostoff, S., Sasse, M.A.: Are passfaces more usable than passwords? a field trial investigation. In: McDonald, S., Waern, Y., Cockton, G. (eds.) People and Computers XIV – Usability or Else!, pp. 405–424. Springer, London (2000). https://doi.org/10.1007/978-1-4471-0515-2_27
8. Burr, W.E., et al.: Electronic Authentication Guideline: Recommendations of the National Institute of Standards and Technology - Special Publication 800–63-1. CreateSpace Independent Publishing Platform, USA, U.S. Department of Commerce and National Institute of Standards and Technology (2012)
9. Cao, K., Jain, A.K.: Hacking mobile phones using 2D printed fingerprints. Technical report, Michigan State University (2016)
10. Cranor, L.F.: Time to rethink mandatory password changes (2016). https://www.ftc.gov/news-events/blogs/techftc/2016/03/time-rethink-mandatory-password-changes

11. Garfinkel, S., Lipford, H.R.: Usable Security: History, Themes, and Challenges. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan & Claypool Publishers, San Rafael (2014). https://books.google.fr/books?id=HPS9BAAAQBAJ
12. Grissinger, M.: Avoiding confusion with alphanumeric characters. Pharm. Ther. **37**(12), 663–665 (2012)
13. Hausawi, Y.M., Allen, W.H.: An assessment framework for usable-security based on decision science. In: Tryfonas, T., Askoxylakis, I. (eds.) HAS 2014. LNCS, vol. 8533, pp. 33–44. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07620-1_4
14. Huh, J.H., Kim, H., Bobba, R.B., Bashir, M.N., Beznosov, K.: On the memorability of system-generated pins: Can chunking help? In: Eleventh Symposium On Usable Privacy and Security (SOUPS 2015), pp. 197–209. USENIX Association, Ottawa (2015)
15. Ives, B., Walsh, K.R., Schneider, H.: The domino effect of password reuse. Commun. ACM **47**(4), 75–78 (2004). https://doi.org/10.1145/975817.975820
16. Keren, G., Baggen, S.: Recognition models of alphanumeric characters. Percept. Psychophys. **29**(3), 234–246 (1981)
17. de Leeuw, K.M.M., Bergstra, J.: The History of Information Security: A Comprehensive Handbook. Elsevier Science, Amsterdam (2007). https://books.google.fr/books?id=pQBrsonDp6cC
18. McCabe, J.A.: Learning and memory strategy demonstrations for the psychology classroom (2014). http://goblues.org/faculty/professionaldevelopment/files/2012/01/McCabe-2014-Learning-Memory-Demos1.pdf
19. Norman, D.A., Fisher, D.: Why alphabetic keyboards are not easy to use: keyboard layout doesn't much matter. Hum. Factors **24**(5), 509–519 (1982). https://doi.org/10.1177/001872088202400502
20. Pilar, D.R., Jaeger, A., Gomes, C.F.A., Stein, L.M.: Passwords usage and human memory limitations: a survey across age and educational background. PLoS One **7**(12), (2012). http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3515440/. pONE-D-12-21406[PII]
21. Reddy, P.V., Kumar, A., Rahman, S., Mundra, T.S.: A new antispoofing approach for biometric devices. IEEE Trans. Biomed. Circuits Syst. **2**(4), 328–37 (2008)
22. Ruiz-Albacete, V., Tome-Gonzalez, P., Alonso-Fernandez, F., Galbally, J., Fierrez, J., Ortega-Garcia, J.: Direct attacks using fake images in iris verification. In: Schouten, B., Juul, N.C., Drygajlo, A., Tistarelli, M. (eds.) BioID 2008. LNCS, vol. 5372, pp. 181–190. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89991-4_19
23. Schiller, N.O.: Masked priming of sublexical units segments vs syllables. In: Steiner, F. (ed.) Advances in Phonetics : Proceedings of the International Phonetic Sciences Conference (IPS) (1999)
24. Shay, R., et al.: Correct horse battery staple: exploring the usability of system-assigned passphrases. In: Proceedings of the Eighth Symposium on Usable Privacy and Security, p. 7. ACM (2012)
25. Shay, R., et al.: Designing password policies for strength and usability. ACM Trans. Inf. Syst. Secur. **18**(4), 1–34 (2016). https://doi.org/10.1145/2891411
26. Shay, R., et al.: Encountering stronger password requirements: user attitudes and behaviors. In: Proceedings of the Sixth Symposium on Usable Privacy and Security, SOUPS 2010, pp. 1–20. ACM, New York (2010). http://doi.acm.org/10.1145/1837110.1837113
27. Smith, D.F., Wiliem, A., Lovell, B.C.: Face recognition on consumer devices: reflections on replay attacks. IEEE Trans. Inf. Forensics Secur. **10**, 736–745 (2015)

28. Stenton, A.: The contribution of the computer to improving L2 oral production. an examination of the applied and theoretical research behind the swans authoring programme. Etudes en Didactique des Langues (19) (2012)
29. Ur, B., et al.: Design and evaluation of a data-driven password meter. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, CHI 2017, pp. 3775–3786. ACM, New York (2017)
30. Ur, B., Bees, J., Segreti, S.M., Bauer, L., Christin, N., Cranor, L.F.: Do users' perceptions of password security match reality? In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, CHI 2016, pp. 3748–3760. ACM, New York (2016)
31. Whitman, M.E., Mattord, H.J.: Principles of Information Security, 4th edn. Course Technology Press, Boston (2011)
32. Yan, J., Blackwell, A., Anderson, R., Grant, A.: Password memorability and security: empirical results. IEEE Secur. Priv. **2**(5), 25–31 (2004). https://doi.org/10.1109/MSP.2004.81