# Password Visualization Beyond Password Masking

Nils Gruschka[1], Luigi Lo Iacono[2]

[1]NEC Laboratories Europe, Heidelberg, Germany
nils.gruschka@neclab.eu
[2]European University of Applied Sciences, Brühl, Germany
l.lo_iacono@eufh.de

**Abstract:** When entering a password (or other secrets) the typed input is most commonly masked, i.e. the characters are hidden behind bullets or asterisks. This, however, complicates the input and highly decreases the user's confident causing several issues such as login failure attempts. On the other hand, password masking is an important security requirement for a lot of applications and contexts to prevent a third person to read the password. Thus, simply dropping password masking is not feasible in general. A common solution provides the user with the choice of toggling password masking on and off, but due to distinct defaults (in dependency of the application and context) this is rather complex and confusing. Enhanced password visualization technologies beyond the simple masking of passwords can provide more sophisticated solutions from both a usability and security perspective.

In this paper, available password visualization technologies are presented and discussed. Furthermore a novel password visualization approach is introduced, the TransparentMask, which provides unique properties in comparison to the existing schemes. Amongst these are the ability to detect mistakes while typing and being able to localize and correct the typo within a certain range. Finally, a security analysis of the TransparentMask shows that the protection level given by a certain password length is slightly decreased in comparison to the fully masked approach.

## 1 Introduction

Passwords are still the most commonly used and widely deployed security mechanism today. Regardless in which domain or context, passwords are ubiquitous and dominant when it comes to user authentication in contemporary information and communication systems. The broad application of passwords and the various issues surrounding them made this authentication mechanism an important research topic.

Usability aspects of passwords have been studied intensively in the past, since most of the issues coming along with passwords are rooted in usability short-comings and have always been a focus topic of usability conferences and workshops considering security and privacy. However, the masking of passwords has not been investigated in research so far. Initial attempts appeared after Jakob Nielsen challenged that masking of passwords should be stopped [Nie09]. According to his view, usability suffers as soon as passwords are typed in and the only feedback available is a sequence of bullets or asterisks. Since users can not see what they are typing they feel less confident and will make more errors which can have serious consequences, such as refused service delivery or even wiping the content of a device when typing the wrong password too often. Thus, password masking leads to increased support calls or increased costs due to lost business (e.g. users refusing to log into a site). Another issue relying in users feeling less confident about typing passwords affects security. The more uncertain users feel about entering passwords, the more likely they are to employ overly simple passwords leading to a true loss of security.

The mentioned drawbacks get even more serious with passwords policies getting constantly stricter—especially in terms of password length which needs to be adapted to steadily improving attack methods [Oec03]—and pass-phrases getting more frequently used. Stopping the masking of passwords seems therefore as a sensible step to resolve these issues. This becomes even more evident when reconsidering the threat against which password masking aims to protect. In the so-called *shoulder surfing* attack an unauthorized entity is trying to capture a password by looking over the shoulder of a victim. The easiest way to capture the password is of course to read it from the screen and that is where password masking comes into play. However, under many circumstances this threat does not exist especially when users are sitting in a trustworthy environment with no unauthorized entities spying at their screens. This led Nielsen to his claim mentioned before.

Still, there are applications and contexts in which the masking of passwords in order to prevent shoulder surfing attacks is required [Sch09]. Examples include public terminals such as ATM machines, areas which are monitored by video surveillance or situations in which third persons may watch on the user's screen such as during presentations. Another argument in favor of password masking is that it alerts and remembers users that passwords are a secret.

The conclusion of this discussion is that for certain types of applications and contexts password masking has its right to exist. In these cases, the drawbacks concatenated with password masking needs to be accepted due to the lack of more appropriate solutions.

Going a step beyond the simple displaying of undifferentiated bullets while users enter complex codes towards password visualization approaches might have the potential to provide appropriate solutions between the binary masked or unmasked approaches discussed so far. The goal of such visualization technologies is enable a compromise between usability and security by providing a more secure but still usable middle ground between masked and unmasked passwords.

## 2 Related Work

Enhancements to simple password masking started to appear in the recent past. Some mobile phones and most smart phones include an approach which shows each character briefly while typing and before getting masked. The purpose of this mechanism is targeting the fact that the keys of these devices are commonly very small and their distribution as well as the multiple assignments of letters to keys differs a lot from usual keyboards. This is an additional source for typos and that is what the mentioned approach is primarily trying to enhance. Still, this gives just a character-by-character snapshot which is difficult to follow, since the typing flow is scattered by the continuous checking of each typed character. At the end, the password remains fully masked and there is no hint on whether a character has been missed or another fault occurred without being spotted on the ”by-character” check.

HalfMask [Dar09a] is an approach, that gets rid of the masking and obscures the password instead with semi-visible random characters (lower case alphabet letters) in the background (see Figure 1). The password is displayed with a very thin-lined font in black. The overall readability is blurred by putting a thick-lined font consisting of random characters in the back. Reading the password from the screen is still feasible , but only when focusing on the text field very intensively. Thus, to a casual observer the field will be unreadable at a glance. On the other hand, for the user entering the password it remains practically unreadable as well. The improvements provided by HalfMask in comparison to common password masking are henceforth limited.

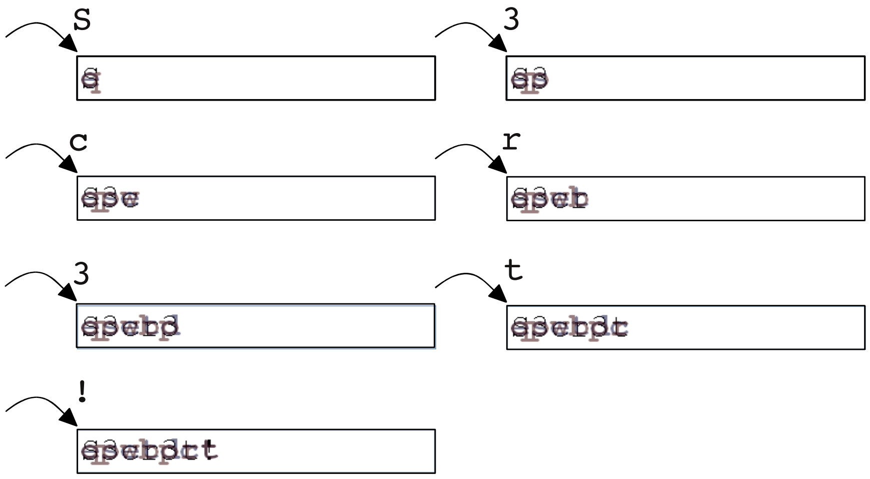


Figure 1: HalfMask

The HashMask [Dar09b] approach visualizes a hashed representation of a password as a spark-line with color which is placed and displayed at the right hand side of the password input field (see Figure 2). The password is hashed using the cryptographic one-way hash function SHA-1 [MOVR97] and a subset of the resulting hash is used as input to the spark-line drawing component.

The intent is that users would become familiar with these images (especially the last one providing a graphical representation of the whole password) and be able to confirm that they typed the right (or wrong) password. Since each entered character results in a distinct spark-line being drawn, the memorization of all of the curves is overwhelming especially in cases in which passwords reach or even exceed the recommended lengths. Also the final colored curve which represents the whole password is still difficult to remember.

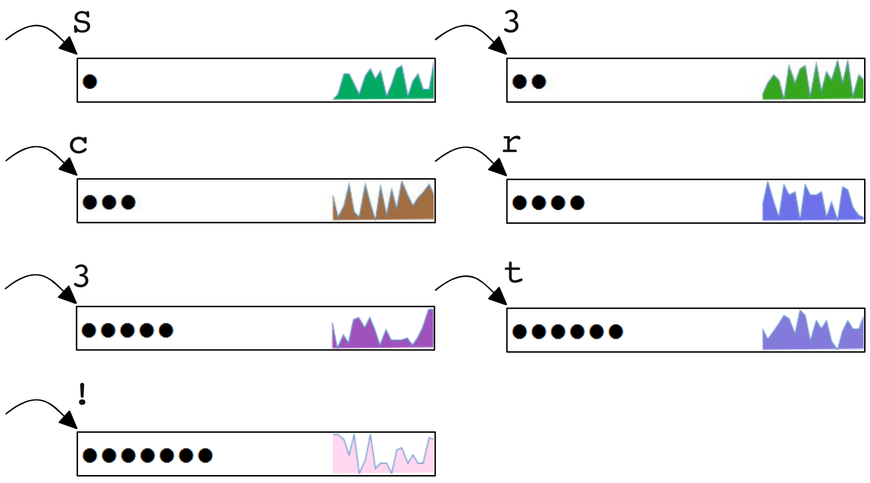


Figure 2: HashMask

Typos which at the end result in a slightly different color nuance and spark-line may be hard to spot by the user. In case the final pattern is different enough from the memorized one, the user needs to re-enter the whole password, since he has no clue about where the typo occurred. Finally, HashMask might not prevent an observer to reconstruct the password, if an attacker is able to monitor all spark-lines. The deterministic algorithm allows to construct a-priori and off-line a mapping table between entered characters and corresponding spark-lines. Depending on the depth of this mapping table an according number of password characters can be determined (if not all), making shoulder surfing attacks more difficult but still possible to conduct.

ChromaHash [Tho09] follows a similar approach as HashMask but uses colors only and not a combination of colors and spark-lines (see Figure 3). Again, the SHA-1 hash function is used to compute a digital fingerprint of the entered characters which are then mapped to a color code consisting of three colored rectangular bars (placed at the right hand side of the password input field).

Due to the less complex pattern which needs to be remembered, the ChromaHash approach addresses the human mnemonic abilities more appropriately. Besides this, ChromaHash has the same disadvantages as HashMask. In fact, in order to overcome the issue of monitoring the graphical password representations and deducting the password characters from these visualizations, ChromaHash includes a time-delay. This means, that the three bars will only appear after a certain time. In case a user is typing in the first few characters quickly, there will be no graphical representation shown. Thus, the most important pattern to remember is the last one, which gives a representation of the complete password. As for HashMask, the intermediate representations do not help much, since there are too many of them and therefore they are unlikely to be memorized by the user.

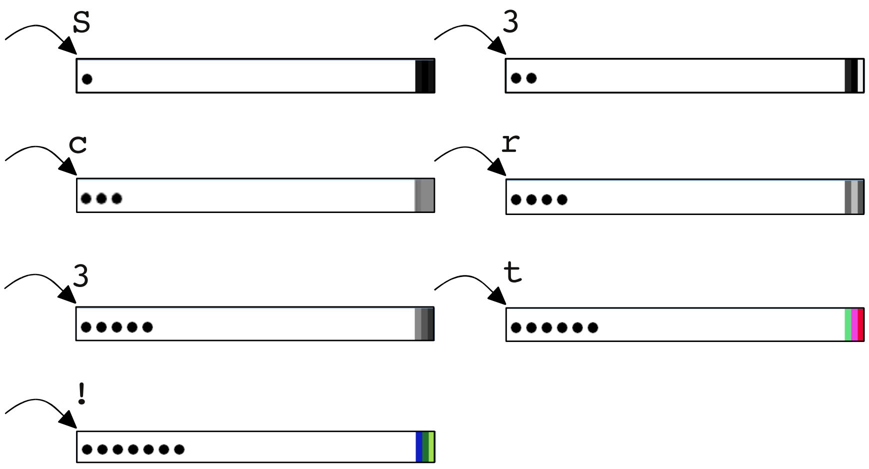
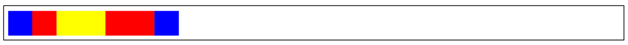


Figure 3: ChromaHash

## 3 TransparentMask

The approach introduced and presented in this paper is based on the idea to include visual clues directly in-line with the password masking characters. In other words, there is not exactly one password masking character in exactly one color anymore, but a multitude of colorful characters hiding the password and at the same time providing some information on the entered characters. By this, the mask gets a bit more transparent in the sense that it allows the user entering the password to glimpse partially through it and extract some limited information on the typed characters.

The naïve starting point is to encode each typed character into one out of the set of masking characters. This is illustrated in the following figure in which the set of masking characters is made of squares in five different colors (black, red, green, blue and yellow). The password used is—as for all the examples in this paper—*S3cr3t!*.



To compute the color for a square, the ASCII code of the typed character modulo $n$ (e.g. $n = \{5, 7, 11\}$) is calculated. Due to the small number of masking characters, such an approach still provides an appropriate protection against shoulder surfing, since a quite large number of distinct characters map to the same color maintaining a certain level of uncertainty for an attacker. However, obvious drawbacks of this intuitive scheme include that for large passwords (and especially pass-phrases) the resulting color pattern is difficult to remember. Then, using only colors will exclude color-blind people to use and benefit from such a technology. Also note, that since the color codes are generated for each character independently, the same character will always result in the same color code. To avoid this, the typed characters should be concatenated in some way.

These observations led to an improved version of the naïve approach, which we call *TransparentMask*. It chains the typed characters and groups them according to a certain block size $b$. Then, TransparentMask makes use of shapes in addition to colors. Figure 4 shows an example with $b = 3$ and $n = 5$.

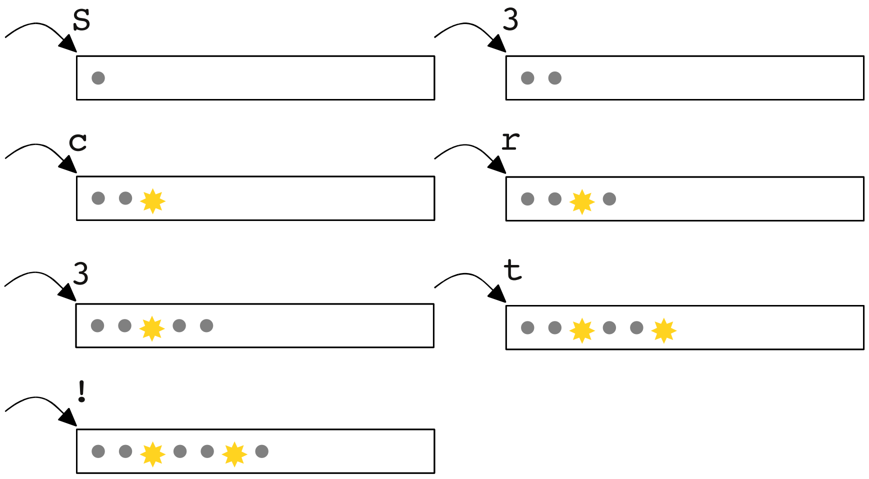
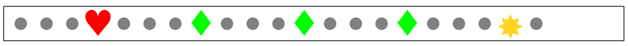


Figure 4: TransparentMask

An example, with a larger password—which is a three-times concatenation of *S3cr3t!*—and with the parameters $b = 4$ and $n = 5$ is given in the following figure.



By using shapes in addition to colors—with a unique and fixed color for each shape—color-blind people will be able to use this scheme as well. Furthermore, it helps in memorizing the visual representation of the passwords, since individuals have distinct capabilities of memorizing things. Following this human factor, TransparentMask provides the possibility to rely on the shapes, the colors, or both for memorizing the password visualization.

The grouping of characters to a certain block size reduces the amount of things which need to be memorized. A further advantage of the grouping is, that shoulder surfing attacks become more difficult, since a non-bullet masking character graphically represents the group of characters. Note that $b$ can be configured to any sensible size. For pass-phrase entering dialogs, e.g., larger group sizes are recommended. Another variant starts with a large block size and decrements it step-wise with an increasing number of entered password characters.

To compute the first colored shape the first set of $b$ characters are concatenated and then hashed using the hash function SHA-1. The hash value is reduced modulo $n$. The result specifies which of the colored shapes to select. In this concrete setting, five distinct shapes in an unique color are used to: a red heart, a yellow sun, a blue circle, a green diamond and a black square. For all following characters the hash computation is continued accordingly so that a generated shape represent all characters entered until this corresponding one.

The following more formal description gives the algorithm to compute the Transparent-Mask visualization pattern. Let $p = p_1|\dots|p_k$ the characters of the password entered by the user and let $v = v_1|\dots|v_k$ the respective visualized characters and let $S = \{s_1, \dots, s_n\}$ the set of visualization symbols. Then the following is true for $i \in 1, \dots, k$:

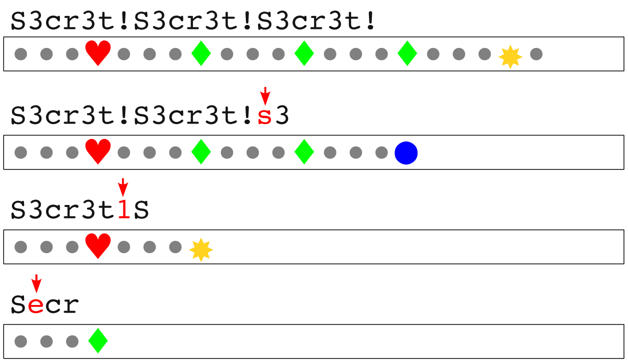
– if $i = 0 \mod b$:
  $v_i = s_h$ with $h = hash(p_1|\dots|p_i) \mod n$

– if $i \neq 0 \mod b$:
  $v_i =$ ●

For the standard implementation using $n = 5$ we chose the following set of symbols $S$:

{■, ♥, ♦, ●, ✸}

The TransparentMask scheme has another interesting property. It provides the user with a hint, where a typo occurs. The following figure gives some examples.



Whenever an intermediate password visualizing symbol is incorrect, i.e. does not match with the visualization of the real error-free password, then the user knows, that an typo occurred. The user needs then to go back to the last correct block and restart typing from there. The position to restart is $i \cdot b + 1$ (a multiple of the block size + 1) used to group the password characters. This is of course not a perfect solution, since in many cases it will be difficult to go back to the last correct block and to know what to continue typing. Further improvements are required, to provide more suitable solutions. In case of large pass-phrases, e.g., the space character might be shown in plain text to give the user additional feedback, in which word the typo occurred. However, these has implications in respect to shoulder surfing attacks, possibly influencing the adoption of such an approach in certain application domains or contexts.

It might also be the case, that a typo occurred in a previous block resulting to a distinct password visualization symbol not in the current but in a subsequent block. The probability of such collisions can be reduced by increasing $n$ (i.e. the amount of visualization symbols) which in turn will reduce the ability to memorize the visualization pattern. User studies still need to be conducted in order to find a well-balanced configuration.

## 4 Security Considerations of TransparentMask

As mentioned earlier, the main reason for password masking is the threat of shoulder surfing attack. Therefore, when reducing the masking effect by password visualization, one has to consider the impact on this attack. Thus the question is, how much more information does an attacker gain from looking at a TransparentMask-visualized password compared to a fully masked one.

An attacker can test if a guessed password results in the eavesdropped TransparentMask pattern. For a password of the length equal to the block size $b$ an attacker can eliminate $(n-1)/n$ of the possible passwords or in other word an attacker can reduce the set of passwords by the factor $n$ by using the information from the visualization. In general the reducing factor $\phi$ for a password of length $k$ then is:

$$\phi = n^{\lfloor k/b \rfloor}$$

If an attacker is only in possession of the visualized password it is obviously still very unlikely for the attacker to reconstruct the right password out of it. Additionally, testing the password using the according login system typically allows only a limited number of wrong tries. The question remains, how much the captured visualization simplifies an attack that allows testing a large number of passwords candidate per time. As an example, such an attack is possible if the attacker got knowledge of the encrypted or hashed password (e.g. from the the UNIX `/etc/shadow` file) [Kle90]. In the following, the attack scenario in which an attacker knows the TransparentMask visualization and the hash value of the password is discussed for two different password sizes. However, one should keep in mind that it seems rather unlikely that an attacker learns both the visualization and a password derived secret.

Consider a short password as an example: $k = 10$, $n = 5$ and $b = 4$. If one assumes that such a short password is well chosen (i.e. does not include a word contained in a dictionary), the only way of choosing a password candidate is *brute force* using the set of letters, numbers and special characters (e.g. the 95 printable ASCII characters). By using the information from the visualization the attacker can reduce the number of possible passwords by $\phi = 25$, i.e. from $95^{10} \approx 6 \cdot 10^{20}$ to $95^{10}/25 \approx 2 \cdot 10^{19}$. In other words, the usage of the TransparentMask visualization reduces the "effective" length of the password from 10 to 9,3. Thus, the negative effect of the visualization can be compensated by extending the password by one character.

Assume a longer pass-phrase as a second example with $k = 30$, $n = 5$ and $b = 4$. With these example settings, the reduction factor is $\phi = 78125$. As such a pass-phrase is typically created using dictionary words, a dictionary attack is feasible here. If one assumes a dictionary size of 250,000 entries and an average word length of five characters[1] plus one space character as word separator, the pass-phrase consists of 5 words and thus the shoulder surfer can reduce the number of possible passwords from $250000^5 \approx 10^{26}$ to $250000^5/78125 \approx 10^{22}$. In other words, the effective length of the pass-phrase is reduced from 30 to 25.

## 5 Conclusions

Password masking can be neglected for some applications and contexts, for others it is a necessity. Therefore, simply dropping password masking is therefore not feasible in general. Providing the user with the choice of toggling password masking on and off would be a solution, but due to distinct defaults (in dependency of the application and context) is it rather complex and confusing. Enhanced password visualization technologies beyond simple password masking can provide more sophisticated solutions from both a usability and security perspective.

The password visualization approaches proposed so far, fail to deliver the desired balance between usability and security in order to provide a usable middle ground between plain and masked passwords. A novel password visualization technique has been introduced in this paper, which provides enhancements and unique properties in comparison to the available schemes. Amongst these are the intuitive and easy to memorize visualization and the ability to detect an error while typing and being able to localize and correct the typo.

A usability study still needs to show, which of the discussed approaches satisfy the users' expectations and needs the most. Also, a more far-reaching security analysis needs to investigate, which of the approaches are suitable for what kind of applications and contexts. As future work, a catalog of best practices needs to be compiled from such studies in order to provide a foundation for developers and researchers to apply and respectively advance the current state of the art.

[1]This is the average word length for English, for other languages of course other values must be considered. See *Languages by average word length* at http://blogamundo.net/lab/wordlengths/ (last accessed on February 2010)

## References

[Dar09a] Chris Dary. *HalfMask*. Website, July, 2009. http://lab.arc90.com/2009/07/08/halfmask-an-experiment-in-password-masking/ (last accessed on April 2010).

[Dar09b] Chris Dary. *HashMask*, Website, July, 2009. http://lab.arc90.com/2009/07/09/hashmask-another-more-secure-experiment-in-password-masking/ (last accessed on April 2010).

[Kle90] Daniel V. Klein. *Foiling the Cracker; A Survey of, and Improvements to Password Security*. In Proceedings of the USENIX Security Workshop II, pages 5–14, Portland, 1990.

[MOVR97] Alfred J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone, and Ronald L. Rivest. *Handbook of Applied Cryptography*, CRC Press, 1997.

[Nie09] Jakob Nielsen. *Stop Password Masking*. Website, June, 2009. http://www.useit.com/alertbox/passwords.html (last accessed on April 2010).

[Oec03] Philippe Oechslin. *Making a Faster Cryptanalytic Time-Memory Trade-Off*. In Proceedings of Crypto 2003, LNCS 2729, pages 617–630, Santa Barbara, 2003.

[Sch09] Bruce Schneier. *The Pros and Cons of Password Masking*. Website, July, 2009. http://www.schneier.com/blog/archives/2009/07/the_pros_and_co.html (last accessed on April 2010).

[Tho09] Matt Thompson. *Chroma-Hash: A Belated Introduction*. Website, July, 2009. http://mattt.me/2009/07/chroma-hash-a-belated-introduction (last accessed on April 2010).