Visual Cryptography and Obfuscation: A Use-Case for Decrypting and Deobfuscating Information using Augmented Reality

Patrik Lantz[1][2], Bjorn Johansson[1], Martin Hell[2], and Ben Smeets[1][2]

[1] Ericsson Research, Sweden
{patrik.lantz, bjorn.a.johansson, ben.smeets}@ericsson.com
[2] Department of Electrical and Information Technology
Lund University, Sweden
{patrik.lantz, martin.hell, ben.smeets}@eit.lth.se

Abstract. *As new technologies emerge such as wearables, it opens up for new challenges, especially related to security and privacy. One such recent technology is smart glasses. The use of glasses introduces security and privacy concerns for the general public but also for the user itself. In this paper we present work which focus on privacy of the user during authentication. We propose and analyze two methods, visual cryptography and obfuscation for protecting the user against HUD and camera logging adversaries as well as shoulder-surfing.*

Keywords: Visual Cryptography, Visual Obfuscation, Augmented Reality, Wearables

1 Introduction

Recent research [1, 2] in privacy-preserving human-computer interaction allows users to authenticate and decipher data using smart glasses equipped with a camera. Decrypted data or one-time authorization codes (OTAC) are displayed as an image overlay in a heads-up display (HUD). The user can then interact with a terminal screen while preventing shoulder-surfing as an adversary cannot observe the HUD. However, this does not mitigate attacks where the adversary has access to the information presented in the HUD.

In our proposed methods, using visual obfuscation and a modified visual cryptography scheme, we split information shown in the HUD into two or three partitions. These partitions are displayed on a terminal screen and in the HUD. Decrypting and deobfuscating information is then a matter of aligning the image overlay in the HUD with the information displayed on the screen. For the attack model we assume that an adversary has access to a) one of two or b) two of three partitions. The adversary could be shoulder-surfing or is capable of observing the HUD. In case b) we assume that the adversary is capable of combining these partitions easily. In case of an adversary which has control over the camera, then we rely on b) only. However, case a) still holds if camera recording can be disabled or prevented to record [3].

2 Related work

Earlier work has focused on private interactions and authentications between a user with smart glasses and a terminal screen, protecting against a shoulder-surfing.

Forte et al. present EyeDecrypt [1] which allows an authorized user to decipher data shown on a display. The decrypted data is shown on a mobile device or smart glasses as an image overlay. If the overlay displays digits in a PIN pad, then using augmented reality, the user can securely enter input onto a screen if the user interface is randomized.

In Ubic [2], Simkin et al. describe an authentication protocol involving smart glasses. Users approach a terminal screen and decode a signed visual encoding (such as QR codes). Then the user initiates the protocol by sending his identification to the terminal host which checks the public key of the identifier. The terminal host creates a challenge, encrypted with the public key and displays it on the screen. The user decodes yet another visual encoding, decrypts it and the challenge is displayed in form of an OTAC in the HUD of the glasses which the user enters on a keypad. Simkin et al. shows that the protocol protects against shoulder-surfing but also against active attackers, such as a man-in-the-middle attack.

Previous work on visual cryptography which this paper is influenced by is the Naor-Shamir visual cryptography scheme [4]. In this secret sharing scheme 2 or n users can mechanically decrypt a visual image by overlaying the shares of the images, assuming transparency in the shares. A secret image is broken up into n shares so that the original image will only be decrypted by someone with possession of all the shares.

3 Visual Cryptography

3.1 Original version

The original idea of Naor-Shamir visual cryptography scheme uses two components created as a number of black and white sub-pixels. These two components are superimposed to reveal the original image. Using a one-time-pad (OTP) with the same size as the original image as the first component and creating an encrypted image by taking the XOR of the original image and the OTP is well known. In order to create a XOR visually each pixel in the original image is represented by a pair of, or 4 sub-pixels and the superimpose is performed by pixel-wise addition. This creates an image which has all white sub-pixels where the original image was 1 and half white/half black where the original image was 0.

3.2 Modified version

Now we consider the application where a number of secret digits (or characters) are to be shown to a user, e.g. an OTAC or a randomized PIN pad. The digit information is split into two (or more) parts shown on a terminal screen and a HUD. Instead of having two components of the same type consisting of black and white sub-pixels as in [4] , we have one component (screen) consisting of black and white sub-pixels, and one component (HUD) consisting of white and transparent sub-pixels. The HUD dominates the screen meaning that a white pixel in the HUD will make the corresponding pixel in the superimposed image white regardless of the value on the screen for this pixel. For a pixel position that is transparent in the HUD the superimposed image will get the value on the screen for this position. This can be formulated as follows; If we represent black/transparent as 0 and white as 1 we superimpose by pixel-wise OR, or the max-operation in the case of grayscale images as in our experiments.

The creation of the encrypted components is performed as follows, also shown in Fig. 1. First we create a temporary OTP with the same size as the original image consisting of ones and zeros. We then represent a one in the temporary OTP by four sub-pixels in a new OTP with white subpixels on the diagonal $A = \left(\begin{smallmatrix} W & T \\ T & W \end{smallmatrix}\right)$ and a zero by two transparent pixels on the diagonal $B = \left(\begin{smallmatrix} T & W \\ W & T \end{smallmatrix}\right)$. This larger image is now used as our OTP. We then create an encrypted original image with the following rule, assuming black digits on white background. If original image pixel is white and OTP is A or original image pixel is black and OTP is B then let the encrypted pixel be represented by $C = \left(\begin{smallmatrix} B & W \\ W & B \end{smallmatrix}\right)$, otherwise represented by $D = \left(\begin{smallmatrix} W & B \\ B & W \end{smallmatrix}\right)$. This way the black pixels on the screen are placed so that they are covered by white pixels in the glasses when we want to create a white pixel in the superimposed image, but placed to be seen through the transparent pixels in the glasses when we want to create a black pixel in the superimposed image. This corresponds to creating the encrypted image by taking the exclusive or (XOR) of the original pixel value and OTP.

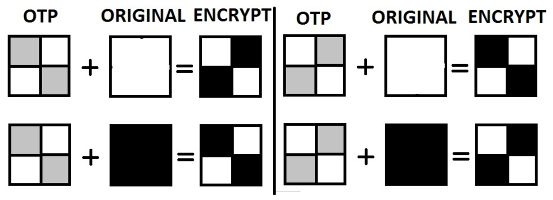
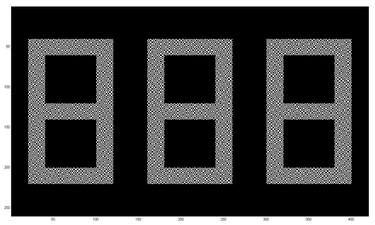
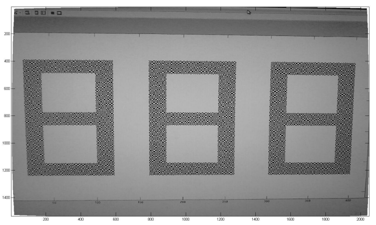


Fig. 1: Encryption and decryption by superimposing OTP

We use the described technique to encrypt digits which are visually revealed when the two components are superimposed. Fig. 2a) shows a randomized OTP where to each pixel in the original image we have created a 2x2 matrix of sub-pixels. Fig. 2b) shows a picture of the encrypted data visualized on a computer screen captured by a camera. For each pixel in the original image a 2x2 pattern for the encrypted image was created according to the above. Estimated camera parameters were used to compensate for radial and tangential distortions in the picture. The transformation used for warping the OTP to match the picture was estimated manually but could be estimated automatically using standard techniques in computer vision, alternatively the screen and head could be rotated and tilted so that the components match. The superimposed result is shown in Fig. 2c) where the original text can be seen. Due to the following sources for errors the visual decryption is not perfect

– image distortions due to imperfect camera (e.g. nonlinearities)
– 'bleeding' of white areas into black in picture smoothing OTP
– estimated transformation does not perfectly warp OTP to picture


(a) Randomized OTP


(b) Encrypted data

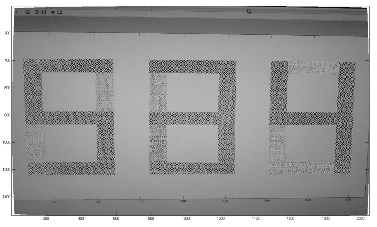
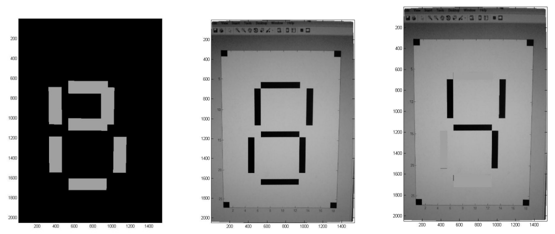

(c) Superimposed result

Fig. 2: Encrypting digits

3.3 Using a seven-segment display

Next we restrict the visual representation of the digits to the well known digital font consisting of seven bars. In this case we can use two sub-bars for each line-segment in the font, similar to what is described by Bochert [5]. In the OTP and in the encrypted image one of the two bars for each line-segment is set, white sets in Fig 3.a) (black background is transparency) and black sets in Fig. 3b) respectively. For the line-segment that are set in the original image to create the digit, the OTP and encrypted image will have different sub-bars set. For the line-segments that are not set in the original image, the line-segments in the OTP and the encrypted image will have the same sub-bars set and will cancel out each other. In the superimposed image the digit will appear in the clear, see Fig. 3c). Note that in the example below the OTP sub-bars are created larger than in the encrypted image, in order to make the system less sensitive to the errors mentioned before. Compared to the approach above, this approach is less sensitive to the alignment between the OTP and the encrypted image but may be visually less attractive.



(a) (b) (c)

Fig. 3: Using two sub-bars for each line-segment

4 Visual Obfuscation

In this next section we describe a method for obfuscating digits. As opposed to the visual cryptography scheme that might suffer from difficulties of alignment the proposed method is not as sensitive. However, in this scenario the attacker has a higher probability to guess correct digit compared to the visual encryption where no information about the digit is learned from by observing one of the partitions.

4.1 Digit representation

For the obfuscation case we represent the digits using seven-segment fonts. This font can easily be divided into several partitions that are shared among a number of digits. The more digits that share the same partitions, the harder it is to guess the correct digit. Fig. 4 shows the bars numbered 0 to 6 that can form all possible digits zero to nine. The digits can be encoded using the binary sequence $x_0x_1..x_6$ where

$$x_i = \begin{cases} 1, & \text{if bar } i \text{ is used} \\ 0, & \text{otherwise} \end{cases}$$

As an example, digit four in Fig. 4 is encoded as the sequence 1100101. The full table of encodings is shown in Table 1.

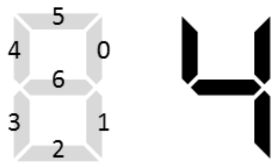


Fig. 4: Bar numberings for binary encoding

4.2 Analysis of 2-way partitioning

Each of the digits shown to the user are partitioned into two parts, one of the partitions are shown in the HUD of the glasses and the other partitions are located on the terminal screen. The user aligns the image overlay in the HUD with the partitions shown on the screen in order to deobfuscate. If an adversary gets access to one of the two partitions, the probability to guess correct digit should be low.

The number of combinations a digit can be partitioned into two parts in is shown in Table 1. Note that one partition can be displayed in the HUD and the other partition on the screen or vice versa which make the actual number of possible partitions twice the number shown in Table 1. In general, the number of partition combinations should be m^i where m is number of partitions and i is number of ones in the binary encoding needed to represent a digit. We only consider partitions which could potentially be used by more than one digit. Therefore, for the digit eight the value should be $2^7/2$ partitions but instead it is 48. This is due to the fact that in some partitions, one of the parts reveals the whole digit, leaving no other candidates to choose from except the digit eight.

Each partition can be part of n number of candidate digits giving a naïve attacker a chance of $1/n$ to guess the correct digit by observing only one partition.

Digit	**Binary encoding**	**# of partitions**
1	1100000	2
2	1011011	16
3	1110011	16
4	1100101	8
5	0110111	16
6	0111111	32
7	1100010	4
8	1111111	48
9	1100111	16
0	1111110	32

Table 1: List of binary encodings and number of partitions for $n >= 2$

However, the probability that a particular part is derived from a particular digit is not the same for all digits so an analysing attacker can do much better. Assume all digits have the same probability and we have an equal probability distribution for the partitions of each digit. The best strategy for an attacker in this case is to guess on the digit having the largest probability for the observed partition. As an illustrative example we choose the digit 4 and partition 5 from Table 2. In this case the adversary would guess number 4 with the probability to guess correct equal to $\Pr(4 \mid 0000100) = 0.3808$ using the values in Table 3. On the other hand, if the adversary has access to the second partition the probability becomes $\Pr(4 \mid 1100001) = 0.470$. We calculated the mean probability to guess one digit correct observing one partition to be 0.45 when observing only one part and assuming equally probable digits and equal distribution among the partitions for each digit. In Section 4.3 we elaborate on the details for how this is computed.

From simulations we also conclude that the mean number of possible PIN pad solutions that match an observed partition of a 10-digit PIN pad is approximately 1300. The number of solutions s for *one* PIN pad with one partition for each digit drawn randomly with equal probability is computed using $s = |S|$. S is a set of 10-tuples with distinct elements and is given by

$$S = \{t_j \mid a_k \neq a_l, \forall a_k, a_l \in t_j \wedge k \neq l\}$$

$$\text{where } t_j \in T, \text{ for } j = 1, 2, ..., |T| \text{ and}$$

$$T = D_{p_0} \times D_{p_1} \times ... \times D_{p_9}$$

p_i denotes the partition at index i in the PIN pad and D is a set of possible digits that a partition p_i can form. In the simulations we compute the average number of solutions for a large number of random PIN pads using s. This gives a naïve attacker about 1 in 1300 to guess the PIN pad correct. Note that the attacker only has to guess the pressed buttons correct in order to have the PIN though.

Partition	**Part 1**	**Part 2**	**Possible digits from part 1**	**Possible digits from part 2**
1	0000000	1100101	10	3
2	1000000	0100101	8	5
3	0100000	1000101	9	3
4	1100000	0000101	7	5
5	0000100	1100001	6	4
6	1000100	0100001	4	6
7	0100100	1000001	6	5
8	1100100	0000001	4	7

Table 2: All possible partitions of digit 4 as described in Table 1 and how many possible digits each part could originate from

Partition	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**0**
0000100	0	0	0	0.0625	0.0313	0.0156	0	0.0078	0.0313	0.0156
1100001	0	0	0.0313	0.0625	0	0	0	0.0078	0.0313	0

Table 3: Probabilities for the two parts of partition number 5 from Table 2 being part of a particular digit

4.3 Optimizing the partitioning

The equal probability distribution for the different parts a digit can be partitioned into as described above may not be the optimal choice if we want to minimize the probability for an adversary to guess correct digit when observing only one part of the partitioning. Let a_{ij} be the probability that digit D_j is chosen to be partitioned and that part P_i is used as one partition. We represent the distribution in a 128×10 matrix A where entry (i, j) is denoted a_{ij}. Only 380 out of 1280 entries in A have non-zero entries since most partitions cannot be used to create a particular digit. Note that Table 3 consist of two rows from A.

$$A = \begin{bmatrix} a_{00} & a_{01} \cdots & a_{09} \\ a_{10} & a_{11} \cdots & a_{19} \\ \vdots & \vdots \ \ddots & \vdots \\ a_{127,0} & \cdots & a_{127,9} \end{bmatrix}$$

First we assume that an attacker knows the distributions of the digits and partitions, that is the matrix A. The optimal strategy for the attacker is to guess the digit having the highest value among the observed partition. The probability of a correct guess when observing partition i is equal to

$$\frac{\max_j a_{ij}}{\sum_j a_{ij}}$$

and the mean probability to guess correct on any partition is equal to $\sum_i \max_j a_{ij}$. In order to minimize the probability for a successful attack we want to minimize this expression over all a_{ij}. There are constraints on A to be valid though. In some applications the probability for each digit should be equal and the probability for all digits should sum up to 1 which gives the constraint $\sum_i a_{ij} = 0.1$, for $j = 1, ..., 10$. Secondly, the two partitions that form the digit must have the same probability, $a_{kj} = a_{lj}$ if partition P_k and partition P_l form digit D_j. Our optimization problem can then be expressed as follows:

$$\text{minimize} \sum_i a_{ij} \max_j a_{ij}$$

$$\text{subject to: } \sum_i a_{ij} = 0.1,\ j = 1, .., 10$$

$$a_{kj} = a_{lj} \text{ if } P_k \text{ and } P_l \text{ form } D_j,\ a_{ij} = 0 \text{ if } P_i \text{ is not part of } D_j$$

This optimization problem is not linear, but can be made linear by the following trick. By introducing helper variables $x_0, ..., x_{127}$ one for each maximum, adding constraints $x_0 \geq a_0, ..., x_0 \geq a_9$, $x_1 \geq a_{10}, ..., x_{127} \geq a_{127,9}$ and change the expression to minimize to $\sum_i x_i$ we get a linear optimization problem that can be solved e.g. using linear programming (LP). We have solved the optimization problem above and the optimal solution gives the attacker a chance of 0.3743 to guess the correct digit observing only one part and knowing which distribution is used when splitting the digits.

A variant of the conditions above is that we do not require the digits to have equal probability. A possible scenario could be when we want to show an OTAC consisting of a number of digits to a user by partitioning it in two parts. If you want to minimize the chance for an attacker to guess the digits in the OTAC you might want to use the digits that are easier to guess (e.g. 1) less frequently than digits that are harder to guess. The first constraint from above is then substituted for

$$\sum_{ij} a_{ij} = 1$$

A simple distribution that fulfills the constraints is to let all a_{ij} be equal, in this case $a_{ij} = 1/380 \ \forall \text{i}, j$. This gives a probability of guessing correct 0.2947. If we use LP to determine the global optimum given this new constraint we get a probability of 0.2867 for an attacker to guess correct if he knows the distribution.

In the analysis above we assume that the attacker knows the distribution matrix and uses an optimal strategy for guessing. In reality we may use different distributions each time. If it is known that the attacker uses the equally distributed matrix B in the guessing we could create other distributions that make the attacker less successful. Also the problem to create such a distribution that minimizes the probability for a successful attack can be expressed as a linear optimization problem and solved e.g. using LP. Let b_{ij} be entries in the known distribution matrix B the attacker uses. The optimization problem can then be expressed as

$$\min \sum_{ij} c_{ij} \cdot a_{ij} \text{ over } a_{ij}$$

$$a_{kj} = a_{lj}, \text{if } P_k \text{ and } P_l \text{ form } D_j$$

$$a_{ij} = 0 \text{ if } P_i \text{ is not part of } D_j$$

where c_{ij} is constructed according to

$$c_{ij} = 0 \text{ if } b_{ij} = 0$$

$$c_{ij} = 1/d \text{ if } b_{ij} = \max_j b_{ij} \text{ and } d = \text{number of max on row } j$$

Using a distribution created this way an attacker guessing according to the equal probability distribution will only have a probability of 0.2833 to guess correct.

4.4 Analysis of 4-bar shape

The main reason the attacker has a rather high probability to guess a digit correct is that the font we use for the digits has 7 bars to represent only 10 digits. If we instead should use the minimal number of bars (four) it would be harder for an attacker to guess correct. Then we would not recognize the representation as our traditional digits but it is interesting to compare the probabilities for this case with our earlier results.

We use 4-bar shapes using the binary encodings $x_0x_1..x_3$ and choose 10 shapes among the 15 possible shapes. Similar calculations as for the 2-way partition analysis were performed for the 4-bar shapes in which the probability becomes 0.2667 instead to guess the correct shape. This result shows that using fewer bars improves this obfuscation method. If we compare this to a two digit OTAC where we have one digit in clear and require to guess the other one, the probability is 0.10 compared with using four 4-bar shapes in which the probability is 0.071.

An alternative way to minimize the number of bars is to let the 7 bars represent more figures. Besides the traditional digits we include a number of characters that can be created from the used digital font. These can be chosen from the set A,C,E,F,G,H,J,L,P,U. We found that the optimal distributions for the scenario with 10 digits and 8 characters, an attacker who knows the distribution has probability 0.2479 to guess correct for equal probable digits/characters and 0.1753 to guess correct for non-equal probable digits/characters. We did the same analysis for the case where we use all 127 figures that can be made up from the seven bars for comparison. In Section 5 we compare the results for the different scenarios from above and discuss the conclusions from them.

4.5 Analysis of 3-way partitioning

In our worst-case scenario, the attacker has access to the HUD display and has hijacked the camera in order to combine the HUD- and terminal-view and is able to record the user input. In this case, the adversary will always be able to guess the digits by either manual analysis or computational image analysis.

Now we assume that that there is an autostereoscopic [6] display available for which the left eye will see different information on the screen than the right eye (camera-view). The obfuscation can be split into three partitions designated for the left and right eye and the HUD.

In our simulations we assume that we do not know what partitions the adversary can observe, it could theoretically be any two. In a simulation similar to the 2-way partitioning we investigate the mean probability for an adversary to guess a digit while having two out of three partitions and combining these for calculating the best candidate digit. The result becomes now 0.60, assuming equal probability of the partitions. If observing only one of three partitions, the probability is 0.28 to guess correct.

5 Results

In this section we describe comparative results of the analysis from Section 3.

The probability 0.3743 for using all digits makes the chance to guess a two digit OTAC equal to 0.1401 which is larger than the probability to guess one digit out of 10 (0.10), which would be the case if we instead would have obfuscated using our benchmark case, that is showing half of the full digits on the screen and rest in the HUD. We focus on the OTAC scenario as we have came to the conclusion that for the PIN pad case, it is better to use the benchmark case if the attacker knows the distribution. The probability for guessing all digits in a PIN pad using the obfuscation is 0.3743^{10} which is larger than 0.10^{5} for the latter case. If we use certain digits that are easy to guess less frequently (non-equally probable digits) as we can in the OTAC case, the probability of guessing one digit correct becomes 0.2867, the probability to guess the two digit OTAC is 0.0822. Hence, the attacker is less successful if we use this approach for showing an OTAC compared to the benchmark case.

In the comparison we want to show how long OTACs are needed if we compare against an OTAC of length 100 as the benchmark case. The cases we compare with and the results are shown in Table 4.

More specifically, these values show how many digits we can use for the different cases before the probability to guess the digits becomes equal to guessing our 100 digit benchmark. According to our analysis our approach is much better than the benchmark if we use some digits more frequently. Even better results are achieved if we also use characters along with the digits. Note that these results assume that the attacker knows the distribution used for splitting the digits. If he does not know the distribution, or if it is known what distribution the attacker is using in the guessing, the probability to guess an OTAC correct is even lower.

6 Discussion

In the PIN pad case it is crucial that the same digits are placed in each partition. If we would place digits randomly between the partitions, an attacker that observes multiple sessions will be able to combine the sessions and learn the full PIN pad. The PIN pad can be randomized as the authors also suggest in [1], additionally, similar to their advantages, a keylogger running on the host will not learn anything about the user input if there is a touchscreen present. The main differences are that we prevent the information to be leaked when an adversary can observe the HUD or even perform camera logging while at the same time preventing shoulder-surfing. Instead of splitting information in three partitions we can split it into two partitions and disable the camera as our method does not rely on a camera. The user can align the glasses himself to the screen by moving

Type	Digits	Description
Benchmark	100	Benchmark case, half of the full digits on the screen and rest in the HUD
10Dig-I	118	10 digits, all with equal probability
10Dig-II	93	10 digits with different probability
10Dig10Let-I	106	10 digits and 10 letters, all with equal probability
10Dig10Let-II	85	10 digits and 10 letters with different probability
10Dig8Let-I	102	10 digits and 8 letters, all with equal probability
10Dig8Let-II	83	10 digits and 8 letters with different probability
127Char-I	93	127 characters, all with equal probability
127Char-II	79	127 characters with different distributions
10Dig4Bar-I	83	10 digits represented as 4-bars, all with equal probability
10Dig4Bar-II	79	10 digits represented as 4-bar with different probability
Uniform	91	It is known that the attacker guess according to equal distribution

Table 4: Comparison of OTAC lengths for different constructions against the benchmark case consisting of 100 digits.

and tilting the head. However, the camera might need to be activated during initial interaction with the terminal needed for any necessary setup between the glasses and the terminal, but after this it could be deactivated.

It is also worth mentioning that the visual cryptography scheme can also be split into three partitions. In this case, the encrypted data can be shown on an autostereoscopic screen while the OTP is displayed in the HUD.

Future work involves mainly to achieve better alignment for the visual cryptograph scheme and investigate the usability of both methods by implementing them in a real pair of smart glasses and conducting user-studies. Applying these methods on letters for sensitive text materials seems to be an interesting application as well.

7 Conclusion

In the first method we have adapted the traditional visual cryptography scheme for use with smart glasses. Using this instead of splitting the digits in two parts as described in Section 3 has the advantage that the number of bars in the model of the digit has no effect on the probability to guess correct digit if you view only one component. However, it requires a more precise aligning between the two components and is perhaps visually not as attractive.

Our second suggestion is to visually obfuscate digits by partitioning them into two or three distinct parts so that it is hard for the attacker to guess correct when observing only one part. The probability for an attacker to guess correct is dependent on how we chose to split up the digits and what strategy and information the attacker has. We have formalized this and formulated optimization problems to find the best distributions used to split up the digits into parts minimizing the probability for a successful attack. There are different scenarios depending on whether all digits must be equally probable or some digits can be more frequent, if the attacker knows the distribution used to split up the digits, if digits and characters are used, etc. By converting the nonlinear optimization problem to a linear optimization problem we can be sure that the found optimum is global. The conclusion from the results is that if the digital font with 7 bars is used to represent only the digits with the traditional appearance, all digits are equally probable and the attacker knows the distribution used to split up the digits, then the benchmarking approach to display half of the digits on the screen and the other half in the HUD makes it harder for an attacker to guess correct digit. However, if we are allowed to use some digits more frequently our approach is much better. Even better results can be achieved by also using characters along with the digits and if the attacker does not know the distribution he will be even less successful in guessing correct. These results makes the OTAC application more suitable than the PIN pad for our obfuscation as in the PIN pad all digits must occur once and only once.

References

1. A. G. Forte, J. A. Garay, T. Jim, Y. Vahlis. "EyeDecrypt - Private Interactions in Plain Sight". Conference on Security and Cryptography for Networks (SCN), 2014.
2. M. Simkin, D. Schröder, A. Bulling, M. Fritz. "Ubic: Bridging the Gap between Digital Cryptography and the Physical World". European Symposium on Research in Computer Security, 2014.
3. K. N. Troung, S.H. Patel, J.W. Summet, G.D. Abowd. "Preventing camera recording by designing a capture-resistant environment". UbiComp, 2005.
4. M. Naor, A. Shamir, "Visual Cryptography", Eurocrypt, 1994.
5. B. Borchert, "Segment-based visual cryptography" , Tech. Rep. WSI-2007-04. Wilhelm-Schickard-Institut für Informatik, Tübingen, 2007. Wilhelm-Schickard-Institut fur Informatik, Tubingen, 2007.
6. N. A. Dodgson, "Autostereoscopic 3D displays", IEEE Computer, vol. 38, no. 8, pp.31 -36, 2005.