High-Entropy Visual Identification for Touch Screen Devices

Nathaniel Wesley Filardo and Giuseppe Ateniese

Johns Hopkins University
Computer Science Department
3400 N. Charles Ave.
Baltimore, MD 21218
http://www.cs.jhu.edu/~{nwf,ateniese}/
{nwf,ateniese}@cs.jhu.edu

Abstract. We exhibit a system for improving the quality of user-derived keying material on touch-screen devices. We allow a device to recover previously generated, highly entropic data suitable for use as (part of) a strong secret key from a user's act of identifying to the device. Our system uses visual cryptography [21], using no additional electronics and no memorization on the part of the user. Instead, we require the use of a transparency overlaid on the touch-screen. Our scheme is similar to the identification scheme of [22] but tailored for constrained, touch-screen displays.

1 Introduction

Mobile devices have become pervasive features of modern life. While handy, these devices typically do not have input mechanisms that make entering secure passwords easy. (In fact, many of them use predictive text models to make entering even low entropy prose easier. This does not bode well for asking the user to enter even short, highly entropic strings such as t5Ax9zK%.) Therefore, we expect mobile devices either to not be used for storing sensitive data or to present a likely vulnerability.

Our system enhances password or pass-phrase security by pairing traditional password entry with the requirement that the user answer a randomly chosen visual challenge. The system does not require that a user memorize any static secret material beyond their extant password; instead, our challenges use visual cryptography [21] and require that the user carry a transparent slide to respond. Informally, this puts our system in the category of systems which "authenticate with something you have" (or as one factor of a multi-factor system) rather than "with something you know." Our scheme is similar to the one in [22] (a detailed comparison may be found in Appendix A).

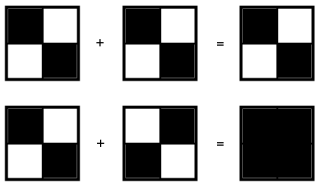
We believe our system to be useful as a generic tool for augmenting password strength, without requiring that users memorize yet more secrets. The challenges encode many bits of entropy in their solution, and are well-suited as a drop-in augmentation to systems, both for authentication and for deriving encryption

keys, which have traditionally used passwords or phrases. Our system is designed so that any attacker not in possession of the user's slide gains no insight into how to answer by collecting any number of challenges (but without seeing responses).

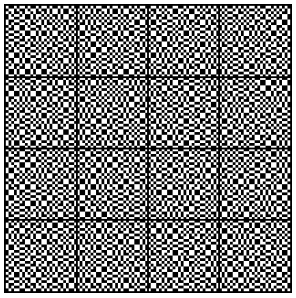
In our demonstration prototype using OpenIntent's OI Safe [23],[1] a password store and cryptography provider for the Android environment, the entropy encoded in the challenge is **concatenated** with the user's password and fed into a traditional Password-Based Encryption (PBE) [17, 1] scheme to decrypt the (strong, random) key used to encrypt individual password entries. That is, authentication takes place by successfully decrypting a second key, rather than the more typical hash-and-compare done with password authentication schemes. The safe stores, in addition to the user's data, enough information to create challenges. A new challenge is generated and the safe is re-keyed every time it is successfully opened.

We will first give a brief review of the basic visual cryptography we use (section 2), followed by an overview of our scheme and prototype implementation (section 3) and a contrast to prior work (section 4). We then discuss our threat model more fully, and the resulting theoretical design of a parametric family of systems (section 5). Having laid out our parameter space, we exhibit our particular realization within this family and apply standard human-computer interaction tools to estimate performance of an ideal instantiation (section 6).

2 Visual Cryptography



(a) The basic 2 × 2-subpixel, 2-of-2 visual secret splitting scheme. Shown here, four display pixels of each share are being combined to produce one pixel of the hidden image.



(b) An example slide, which should be scaled to match the device's display and printed on a transparent sheet of plastic. Slides are composed of random noise rendered as display pixels as per section 2. The grid lines separate independent instances of visual secret splitting (each grid cell) and are imposed only to aid in subsequent use; see Figure 1.

[1] All of the code used for this paper is available on the Web at `http://github.com/nwf/android-vcpass` and `http://github.com/nwf/android-vcpass-oisafe`

Visual Cryptography [21] is a method for encrypting or hiding visual information in a way where decryption may be done by a human without the use of code-books, tables, or computers. The prototypical example, and the one we use in our prototype scheme, is a two-of-two secret splitting scheme, in which a black and white secret image is split into two "black and transparent" shares, neither of which alone conveys any information about the encoded image.[2]

To hide a single pixel of the image, we follow the most basic 2-of-2 secret splitting scheme of [21]. A $b \times b$ block of pixels in the shares will have either identical (for a white pixel) or complementary (for black) diagonals set black (the other pixels will be *transparent*), as shown in 1a. The resulting shares will have b^2 as many pixels as the original image; to avoid confusion we distinguish between "display pixels" of the shares and "image pixels" of the original and reconstructed images. Information security is attained by setting one share's blocks independently, identically distributed (iid) uniformly at random, making it clearly uncorrelated with the hidden image. The other share's corresponding block is then set to the appropriate diagonal. While the image was an input to the values of this share, no information survives due to the iid uniform bit flip channel defined by the first share. Thus the secret is only recoverable from the *pair* of shares, as intended.

3 System Overview

As is typical of secure document stores, OI Safe has a "master key" which is used to encrypt individual entries. The master key is chosen at random and is itself encrypted using a salted PBE scheme fed with the user's password; this makes changing the password independent of the amount of the data being stored. OI Safe may be "opened" by entering the password, which allows it to decrypt the master key, and thereby allows the user and external applications to access stored passwords and its encryption functionality. It may be "closed" (either explicitly or after a configurable timeout) by erasing the in-memory copy of the master key plain-text.

Our prototype enhances the security of the system by combining, prior to PBE strengthening, the user's password with (roughly 36 bits of) random data, hidden using visual cryptography. To open the safe under this new scheme, the user first provides a plain-text password, as with the non-augmented OI Safe, and then decrypts the previously generated random data. The latter step involves placing a transparency (carrying a gridded image such as 1b), previously generated and printed, over the display and indicating the direction (or absence) of an arrow in each grid cell by touching and dragging in the appropriate direction. (The set of arrow and blank images we call the "vocabulary" in subsequent dis-

[2] Formally, the requirement is that there is zero mutual information between either of the shares in isolation and the secret.

cussion.[3]) We emphasize that it is possible to use the touch screen, and therefore to answer the challenge, without removing the overlay slide first.

A camera shot of the challenge prompt and slide overlay running on a Motorola Droid™ phone may be found in Figure 1. This picture gives an idea of the "arrow" vocabulary used and gives an example of what a user of the system would see when answering a challenge.[4] By virtue of visual secret splitting, absent the slide, the phone appears to be displaying random noise. To answer the challenge, the user would touch each cell and drag in the direction indicated in the table. Additional details of implementation may be found in section 6.

The system is initialized by using a desktop computer and printer to create the user's slide. Sufficient information about the slide (i.e., the seed to a CSPRNG and other material; see section 6) are then imported into the safe at construction of the secure store, i.e., at the same time as the user first sets their traditional, plain-text password, just as they would in the conventional (i.e., purely plain-text password based) OI Safe scheme. The safe then generates an initial challenge and stores these parameters under the same encryption used for its own master key. Subsequently, each time the safe is opened, the parameters are decrypted and a *new challenge* is generated, using the device's strong random number generator. This challenge is written to non-volatile store for the next authentication attempt, and the safe master key is re-crypted with the user's password and the answer to this new challenge.[5] That is, whenever the safe is opened, it updates itself to be ready for the next authentication.[6] Cycling challenges in this way helps thwart incomplete surveillance attempts: repeated observation of, say, the user solving the challenge without being able to see the challenge or slide in detail, will not lead to an in-aggregate solution to a challenge, whereas repeated incomplete obsevations of password entry might.

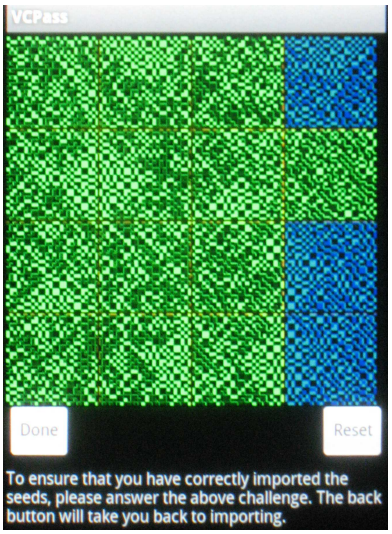
---

[3] Our prototype chooses to use a vocabulary of four arrows, one for each cardinal direction, and two blanks. There are sixteen independent cells in our challenges. These choice balances the entropy of the challenge (see subsection 5.2) against the (estimated) time to answer the challenge (see subsection 6.1), but are not the only possible choice; section 5 will discuss the system, and its parameters, more fully.

[4] In this case, the system is prompting for an answer to check that it knows what the user's slide looks like; in general, the user would see this after having typed in their traditional password at the prompt in OI Safe.

[5] This is akin to the user changing their password every time they use the safe. In fact, the same code-path is invoked when the user does change their password or chooses to change their slide; both of these actions require that the safe be already open.

[6] This act is done entirely in the background since it takes noticeable amounts of CPU time—roughly 10 seconds—on current Android phones. The safe will not close itself until it is ready to be opened again. Possibly we should require visual challenges on a different schedule than closing the safe; perhaps once per day or reboot, so that the user does not typically need their slide with them. That is, we can hold the visual challenge answer in memory on a different schedule than the user's password.



(a) Camera shot of the application's challenge prompt.

none	down	none	down
left	right	up	none
up	none	none	right
none	left	left	right

(b) Solution to challenge

Fig. 1: Challenge and solution. To improve visibility, challenges are displayed using only one color subpixel – in this case green. As the user provides answers (correct or not), the cells are shaded blue; answers are provided by touching each triangle and dragging away from the broad side. The black lines on the slide align to yellow lines between cells on the display. Due to the difficulty of aligning the display, slide, and camera, it may be hard to make out all the triangles in the challenge; the challenge is readable only within a very narrow field of view, even when properly aligned.

4 Prior Work

There are deployed systems (usually under the heading of "biometrics") which attempt to make the statement that "with high probability, the operator of the device is in fact a legitimate user" based on fingerprints [14, 19, 18], facial recognition [14, 2], typing patterns [26, 25], retinal scans [9], etc. While much of this work was based on probabilistic matching, making it untenable as a source for keying material, recent work [8] has shown how to derive good keying material from biometric data. These systems typically require cameras or specialized scanners and may involve a lengthy initial data acquisition phase; our system requires only a touch-screen display (and a printer during initialization).

There are "visual identification" schemes, such as Déjà Vu [7], which use visual recognition for (remote) authentication. The secret here is entered by the discrimination of a series of pre-selected, randomly-generated visuals from a larger set. This scheme trades entropy (it derives at most one bit per displayed image) for a more pleasant user experience (this system does not require that the

user carry a slide). These systems were generally conceived of for desktop, not mobile, environments and therefore use relatively large images and can present many at a time. Further, their low entropy per challenge makes them usable for one-time passwords but less ideal for encryption keys.

The system of [11] uses visual secret splitting to authenticate bank transactions. Here, the user confirms that the bank's share decrypts to correctly identify the requested transaction and then reveals the location of two markers within the image to indicate acceptance. This paper appears not to consider an adversary which accumulates information across multiple uses of the system in order to learn about the user's transparency.

There has been prior work on visual cryptography for authentication of humans to devices. [16] gives a system which requires the user to memorize a secret and (mentally) perform some unspecified "simple operation" on that secret and the message received via visual cryptography. The system of [22] proposes challenges which illuminate regions of a *multi-colored* slide, the responses to which are enumerations of the indicated colors; a detailed comparison may be found in appendix A. The system of [24] uses visual secret splitting to encode passwords in a different context: authenticating users for remote voting; that paper does not consider the amount of entropy in the secret to be split (they offer, for purposes of illustration, only a very low security example; however, as they work on larger displays than we do, and need use a slide only once, this is not really a limitation so much as an omission from the paper).

We also briefly contrast our system to a hypothetical scheme where we used a camera to scan a secret image (like a QR code). Other than the obvious need for the target device to have a camera, this system would suffer from the likely constraint that these secret images should have relatively low pixel density, for ease and reliability of picture-taking. Unfortunately, this would also ease surveillance and adversarial capture of the secret.

Our system focuses on providing a moderately sized, secure channel for entropy with a simple, touch-screen user interface. As with all visual cryptography schemes, our system comes with the added cost of needing to carry a transparency containing a visual cryptography share.

5 Design

5.1 Threat Model

Our design, as with most secret-based systems, aims to defend against semi-active attackers with incomplete surveillance capabilities. We are primarily concerned with an adversary who steals the user's mobile device or finds it after it has inadvertently been left behind and thus has **absolute control** of the hardware for the duration of their attack. Since secure erasure of long-term data (which includes challenges but *not* the user's responses thereto) may be impossible, such acts may compromise all past challenges.[7] In the case of a remote

[7] In particular, modern flash devices engage in "wear-leveling" whereby writes to a logical sector are actually spread among several physical sectors. This greatly

authentication system, the adversary may be able to prompt the challenger to provide a challenge at any time and then abort the protocol. Our system ensures that challenges do not leak data about their interpretation, **even in aggregate**.

At no time will the adversary be given both a challenge and its solution (e.g., through device compromise or surveillance). This restriction may sound severe, but recall that our system is still fundamentally password-like, and that any secret-based system fails if the secret can be observed.[8]

In the same vein, we do not consider software attacks (e.g., viruses, trojan horses, "malware") on the system, as once an adversary is able to observe our process's memory, it becomes a simple matter to read out the secret keys directly. Even in absence of such abilities, software which can capture touch screen events and screen contents can read out the user's password and answers to visual cryptography problems. In the specific case of a password safe, it may be possible to impersonate the legitimate client and simply ask for the secrets contained within the safe directly! We therefore assume that the underlying trusted computing base is indeed sufficiently worthy of the trust placed in it.[9]

The formal game we play is to give our adversary the parameters of the system and the complete set of challenges that the system may ever produce.[10] The adversary wins the game if they can gain a non-negligible advantage over merely guessing.

5.2 The Challenge Schema

Our challenge to the user is relatively simple: given N cells, each of which may each take on one of $|K|$ values (which we call the *vocabulary*), discriminate between $|D|$ ($D \subseteq K$) individual values and the remaining $|K \setminus D|$. Upon prompting, the user is required to answer with which of the cells contain a value from D and to indicate which value in particular. The remainder of the cells require no explicit user action. To generate a challenge, the cells are set unformly at random (iid) from the vocabulary (of size $|K|$). We therefore expect $N\,|D|\,|K|^{-1}$ cells to require user interaction, and each cell will contribute

$$-\sum_i p_i \log_2 p_i = -\frac{|D|}{|K|}\log_2\frac{1}{|K|} - \frac{|K \setminus D|}{|K|}\log_2\frac{|K \setminus D|}{|K|} \tag{1}$$

improves the useful life of the flash, but means that many old copies of rewritten material may be extractable by an adversary.

[8] In fact, our system does marginally better than traditional password-based systems in terms of the effects of perfect observation (see the discussion in Appendix B).

[9] Perhaps if the system were being used for remote authentication, rather than decryption of local data, there would be some room for correctness even in the face of local compromise. Our focus here is, as stated, to guard against the loss of secret data if the device is stolen.

[10] Our system generates each challenge iid uniformly from this space; in a system where that is not the case, the probability of a challenge share might leak information about its contents. If such a system were to be designed, security under our game would require that the theoretical adversary be told these probabilities as well, since it may be possible for a real-world adversary to estimate them with high precision.

bits of entropy. We allow systems with $|D| + 1 < |K|$, (with less than maximum potential entropic return) to let us trade between expected user actions and the resulting entropy.

To generate a challenge, the generator (e.g., the safe after successful authentication) must have access to all the vocabulary entries and slide data for each cell. Naturally, this information must be kept secret, as it allows anyone in possession of it to reply correctly to challenges without being able to see the images themselves. We assume that the generator has access to sufficiently safe encrypted storage (in the case of our prototype, these secrets are stored inside the presumably secure safe). The generator also needs a (cryptographically) secure random number generator to provide the entropy that will later be read back from the challenge.

5.3 System Game

The system may be described as a game between the three parties of user (the operator of the device), generator (the device when it has the system secrets in memory), and verifier (the device when it does not have the system secrets in memory).

1. For each cell $n \in \{1, \ldots, N\}$, the user uses an initialization program to create a vocabulary of equal-length, independent bit strings, K_n, with distinguished subset D_n, subject to the criteria from subsection 5.4. The user then provides all D_n privately to the generator. We use $\boldsymbol{K}$ and $\boldsymbol{D}$ to denote the in-n-order concatenation of all K_n and D_n, respectively.
2. For each of the N cells, the user's initialization program further creates a (iid) random string of bits s_n, the slide, of the same length as strings in K_n. This string is also privately provided to the generator, and the user announces all $\{k_n \oplus s_n | k_n \in K_n\}$ (without revealing any information about D_n). We use $\boldsymbol{s}$ to denote the in-n-order concatenation of s_n.
3. For each of the N cells, the generator selects an element $k_n \in K_n$ iid uniformly at random and (privately) informs the verifier. The generator publishes the challenge, $c_n = k_n \oplus s_n$.
4. The generator stores $E_{\boldsymbol{k}}(\boldsymbol{D}, \boldsymbol{s})$ (an encrypted copy of the private parameters of the system) for later use.
5. The user now computes $\boldsymbol{k'} = \boldsymbol{s} \oplus \boldsymbol{c} = \boldsymbol{s} \oplus (\boldsymbol{k} \oplus \boldsymbol{s})$ and (privately) reveals the answer to the verifier.
6. The verifier accepts if $\boldsymbol{k'} = \boldsymbol{k}$.

An adversary wins the game if they may replace the user in the last two steps and cause the verifier to accept with odds better than $\geq 1/k + \epsilon$ for some $\epsilon > 0$. However, the adversary cannot win without successfully attacking the private exchanges (e.g., via surveillance, timing, or software attacks): an $\epsilon > 0$ implies either nonuniform selection of k_n or correlation between $s_n \oplus k_n$ and k_n, which in turn would imply nonuniform selection of s_n. (In actual usage, steps 3 through 6 are repeated many times; the publication of all the encrypted vocabulary in step 2 is intended to capture this repeated use of the system parameters.)

In our system, comparison of $\boldsymbol{k}$ against $\boldsymbol{k'}$ is checked implicitly: $\boldsymbol{k'}$ is fed through a PBE scheme and used to decrypt a block (containing the safe's random master key) encrypted with $\boldsymbol{k}$. $\boldsymbol{k'}$ and $\boldsymbol{k}$ are never compared directly: successful decryption is taken to imply that $\boldsymbol{k} = \boldsymbol{k'}$. Further, $\boldsymbol{k}$ and $\boldsymbol{s}$ are derived from cryptographically secure pseudo-random number generators; the seeds for these generators stand in for their outputs in step 4 and the re-keying procedure above. After every successful verification, the verifier knows $\boldsymbol{k}$ and may use it to decrypt $E_{\boldsymbol{k}}(\boldsymbol{D}, \boldsymbol{s})$, revealing the secret parameters of the system. Since, in our system, the verifier and generator are the same (i.e., the device, just at different points in time), at this point, steps 3 and 4 may be repeated to produce a new secret key and a new challenge for later authentications, which allows the system to (limitedly) frustrate even perfect surveillance (unlike a pure password scheme, where no such mitigation is possible; see appendix B).

5.4 Visual Secret Shares That Don't Leak

The constructions in [21] are all intended to produce n shares for a single message; no share is ever used for multiple messages. [22] does present a multiple-use scheme for visual identification (authentication of a human), but that scheme considers the equivalent of a cell to be entirely revealed to an adversary after a use; therefore, it requires the use of many more cells to combat an adversary. To keep the number of cells low, thereby allowing for larger cells on smaller displays, we use a standard visual secret splitting scheme to obscure the challenge. To ensure that no number of challenge visual shares will reveal any information about the user's share, we must impose some constraints on the system.

As before, we have N cells, a set of K vocabulary entries for each, $D \subseteq K$ of which are to be distinguished in some way (when combined with the user's share). The cells are each made up of some number of image pixels, P. For each image pixel, we permit only one of the D values to "claim" it. We then set, iid uniformly at random and independent of the user's share, the values of all image pixels in the $K \setminus D$ values and all unclaimed image pixels of each $d \in D$. This "vocabulary generation" happens independently for each of the N cells. Were we to permit more than one $d, d' \in D$ to claim a given pixel, then there would be correlation between challenges containing d and d', thereby leaking information to an adversary.

We assume that the parameters $|D|$, $|K|$, N, P, the subset of the pixels claimed by each D, and the intended decoded value (i.e., the intended image) of these claimed pixels are public. Because each pixel of the slide is only meaningfully correlated with one pixel out of all of the D, and not correlated with any element in $K \setminus D$, the information security argument continues to hold. Instantiation of this scheme requires $NP\,|K|$ independent uniform random bits: $NP\,(|K| - 1)$ of which are consumed by the $K \setminus D$ and unclaimed D image pixels, and NP of which determine the user's share.[11]

[11] We do run the risk of generating a confusing vocabulary: that is, one in which two elements of K may not be sufficiently distinguishable. To mitigate this risk,

5.5 Incomplete Erasure Attacks

As mentioned earlier, whenever one rewrites sensitive material, (e.g., by changing the password in a staged keying system like OI Safe's) there is always a danger that the old copy is not completely erased. In our case, however, the rewritten material is encrypted with the result of a PBE scheme, and each of those copies' keys was derived, in part, from the answer to a visual challenge. Therefore, the key is reasonably entropic. Under standard assumptions of the system used to encrypt the master key, namely IND-CPA, the multitude of messages offers no gain to the adversary.[12]

6 Implementation

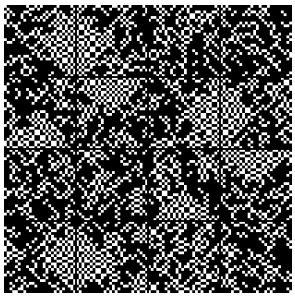
Fig. 2: An example vocabulary for one cell, as seen when overlaid with the slide, with $|K| = 6$ and $|D| = 4$. The four distinguished values (resp. up, down, left, right) are shown at the left and would be distinguished by the user touching the cell and dragging away from the broad side of the triangle. The rightmost two cells do not call for a user's response. Each cell has an independently generated vocabulary encoding the same arrows but with different "filler" pixels.

We now turn our attention to the particular parameters used by our prototype implementation. Rather than being a rigid encoding of this particular choice of parameters, our prototype has been designed to encapsulate the application (e.g., OI Safe) using it from the details of visual cryptography whenever possible: the application is almost entirely oblivious to the contents of the values it passes to our visual cryptography front-end. In testing, we switched a number of these parameters and the design of the distinguished elements without having to (additionally) modify OI Safe at all.

Mobile devices by necessity do not have large displays, both in the sense that there are few pixels present and that the pixels themselves are small. The former restricts the number of cells we can reasonably fit in a challenge. Despite the small screen size limiting our cell count, we chose to present only a single

instantiations of the system should present the full vocabulary to the user when the system is being initialized (i.e., when the slide is being generated). We assume that this clear-text is not intercepted. We assume that any correlation between the vocabulary entries K by the user's rejection of confusing vocabularies are negligible.

[12] Our prototype uses OI Safe's default "strong" choice storing both the encrypted master key and our the secret seeds: BouncyCastle's Java cryptography provider in mode `PBEWithSHA1And256BitAES-CBC-BC`.



(a) An example challenge, ideally rendered, with slide overlaid.

left	right	none	none
up	down	left	right
none	none	up	down
left	right	none	none

(b) Answer to the challenge.

Fig. 3: An example of a system with $|K| = 6$, $|D| = 4$, and $N = 16$.

challenge at a time.[13] The latter had unexpected consequences: we found that using display-native pixel size for the sub-pixels of the visual cryptography made alignment of the slide and challenge almost impossibly difficult; we therefore set the ratio of display to image pixels to 6 : 1. With the additional impact of the inter-vocabulary-item constraints on pixels from subsection 5.4, our vocabulary tends to have images which are not immediately obvious; even an ideal rendering (see Figure 3) leaves something to be desired. It is possible that there are better vocabulary designs to be had or that the issue will be less severe on future display technologies (higher density LCDs or e-ink displays); for the moment, our design suffices.

The size of our cells are chosen so that a 4×4 grid of cells fits on a display of 312×312 (display) pixels. Given the 6 : 1 pixel ratio, this makes our cells 13×13 image pixels and sets $P = 169$. This allows for 42 image pixels to be set by each distinguished value, which is likely large enough that confusable values are unlikely to be generated. Our prototype instantiates our scheme with $N = 16$, $|D| = 4$, and $|K| = 6$. In particular, this gives a 16 cell grid, with an expected 10.6 cells requiring user action (though some challenges will have all or no cells requiring user interaction). Each of the $|D| = 4$ cells is a direction indicator, as shown in Figure 2. The image pixels of these cells which are not fixed by the direction indicator are set randomly as described above.

This system provides roughly 2.3 bits of entropy per cell, or roughly 36 bits per challenge. If all 95 printable ASCII characters are available and used to greatest effect, then each of our challenges may be seen as having added the equivalent of 5.5 characters to the password ($36/\log_2(95) = 5.47\ldots$). Restricting ourselves to the 26 letters of the alphabet, the requisite string length becomes 7.7 ($36/\log_2(26) = 7.65\ldots$). See subsection 6.1 for a way to estimate how long a user would take to respond in each case.

[13] Presenting multiple challenges for the same slide sequentially does not yield linear increase in entropy; the marginal utility of the next challenge behaves as in Table 1.

Initial vocabulary and user share generation currently happens on a desktop computer, to make printing easier. To further the ease of use and development, we instantiate two Cryptographically Secure Pseudo-Random Number Generators (CSPRNGs) using AES in CTR mode with seeds of a few hundred random bits each (rather than store and manipulate the full string of $NP|K| = 16224$ bits). One CSPRNG is used to generate the user's slide, the other is used to generate the "noise" pixels in the vocabulary.[14] While the resulting bit stream is necessarily not an iid uniform string of bits, a CSPRNG's output should be indistinguishable from one by any probabilistic polynomial-time adversary.

Our prototype produces a new challenge after every successful opening of the safe. The secrets of the visual cryptography subsystem are themselves guarded by the same PBE-derived key as the safe's master key. To reduce space consumption, we store the CSPRNG seeds and recreate both the slide and vocabulary in memory on demand.

A few words must be said about the odd coloring of Figure 1. We use a green, rather than white, cell color as it uses only one color sub-pixel in each LCD pixel; this in turn helps the user see the reconstructed image. Yellow lines are shown between cells and corresponding thin black lines are printed on the user's slide to aid in alignment of the two. Due to the mechanics of LCD displays and perhaps also imprecise printing, there is some difficulty in aligning the slide to the screen in a way that makes the entire image clear from any single vantage point; the challenge is readable only within a very narrow field of view, even when properly aligned.

6.1 Estimating Timing

We currently have difficulties reliably both producing a slide with pixels sized correctly for the phone's display and aligning the slide and the display. More precise printing than that of a laser printer, shaping of transparencies, or some form of software-assist[15] may be sufficient to ease usage of our system. However, we do not believe that users would currently put up with the difficulty of use.[16]

In lieu of actual users, we can use Fitts's law [10, 20],

$$M_{ij} = .204 \log_2 \left[1 + \frac{D_{ij}}{W_j} \right], \qquad \overline{MT} = \sum_{i,j} P_{ij} \left[M_{ij} + RT \right]$$

to *estimate* the time it will take for a user to answer a challenge with an ideal slide and display. The left equation relates D_{ij}, the distance between objects

[14] In our prototype, those seeds are passed to the device via a QR barcode rather than as a file, enabling us to work on devices where externally manipulating storage is annoying or impossible. This is not, however, central to the scheme.

[15] For example, we could have the user touch a series of distinguished points on the slide, giving the device a better idea of how to display the challenge. Alternatively, for devices supporting multi-touch displays, we could perhaps manufacture slides that triggered touch events merely by being placed upon the display.

[16] We acknowledge that this represents a practical weakness of our design. We do, however, believe that it can be overcome.

i and j, and W_j,the width of object j, to M_{ij}, the estimated time of motion from i to j. Informally, it can be read as "short distances and large objects allow fast positioning." The right equation computes the average positioning time by weighting the positioning time of each pair M_{ij} with the probability of needing to make that move, P_{ij}; RT is the "reaction time," the time it takes the positioning system (i.e., the user) to find the next move. We can compare the derived estimate for our prototype with similar estimates for using touch-screen keyboards[17].

To estimate the effects of reaction time in the visual cryptography setting, where order of entry is irrelevant and there are potentially many acceptable responses at any moment (i.e., cells not yet answered), we use a weighted version of Hick's Law [12, 20]: $RT(n) = .200 \log_2 [n+1]$. Our variant is a recurrence form, which should capture that users do not re-search already searched cells:

$$RT(n) = \sum_{i=0}^{n} p_{n,i} \left[-.200 \log_2 p_{n,i} + RT(n-i) \right], \quad p_{n,i} = \begin{cases} \frac{|D|}{|K|} \left[\frac{|K \setminus D|}{|K|} \right]^i & i < n \\ \left[\frac{|K \setminus D|}{|K|} \right]^n & i = n \end{cases}$$

where $p_{n,i}$ is the probability that searching n cells for one that requires activity takes i steps. Note that in this variant there is no ambiguity about whether to respond and so no $+1$ inside the $\log_2$—the nonresponse case is handled as $i = n$. For our instantiation, we estimate a total of $RT(N) = 2.9$ seconds spent searching per challenge. To get expected motion time, we compute the expectation of $n\overline{MT}$:

$$\mathbb{E}\left[n\overline{MT}\right] = \sum_{n=0}^{N} p(n) n\overline{MT} = \sum_{n=0}^{N} \binom{16}{n} \left[\frac{|D|}{|K|} \right]^n \left[\frac{|K \setminus D|}{|K|} \right]^{N-n} n\overline{MT}.$$

For our instantiation, this yields an estimate of 2.2 seconds of motion. Combining these yields a grand total of 5.1 seconds to respond to a challenge, neglecting slide positioning time.

Using the Android on-screen keyboard as a prototypical example, we estimate that it would take an expert user (for whom $RT = 0$) 3.6 seconds to enter a (memorized) random 8-character mono-case string or 4.7 seconds to enter a (memorized) random 6-character mixed-case alphanumeric string.[18]

7 Future Work

Our system avails itself only of the most basic form of visual cryptography. Visual Cryptography has been actively studied by many researchers over the

[17] In all cases, we ignore errant entries, so these times are lower bounds. Numbers reported in this section are derived from measurements of a Motorola Droid™ phone running Google Android version "2.1-update1" build "ESE81."

[18] The increase in time is due to the need to transition between shifted and un-shifted modes of the on-screen keyboard.

years. The original Naor and Shamir paper [21] discusses k-out-of-n threshold schemes more general than the 2-out-of-2 we used here. Visual cryptography has been extended to work with full-color images [13, 15, 6], with "meaningful" (i.e., non-random) cover images [5, 27, 28], and general access structures both without [3] and with [4] meaningful cover images. This prior work has tended (the identification schemes of [22] aside) to focus on the act of secret splitting itself, rather than its potential application to authentication.

8 Conclusion

We have developed and exhibited a system which allows users to answer high-entropy challenges without having to memorize said entropy or provide biometric information. The trade-off is carrying a visual cryptography share on a transparency and an (estimated) increase of a few seconds of entry time relative to a memorized, keyboard-based equivalent.

9 Acknowledgements

We are indebted to Matthew Wright for his combinatorial help in deriving the correct form of $a_M(k, i)$. The phone pictured was generously donated to JHU ISI for student use by Google, Inc. We would further like to thank our shepherd Moritz Becker and the several anonymous reviewers for their very helpful comments.

A The Visual Identification System of Naor and Pinkas

The identification scheme with one verifier of [22] uses a transparency, known by the verifier and possessed by the human, composed of N squares, each of which is iid *colored* with one of 10 colors. The challenges in this system serve to illuminate d of these squares and keep the rest dark. The answer to the challenge is the list of colors lit, in some pre-defined order. As with our work, this system assumes incomplete surveillance; indeed, their colored slide is likely easier to observe accurately from a distance or at low resolution than our visual secret splitting share, which uses relatively small pixels.

Our goal is slightly different than the above system, as we seek to generate a large amount of entropy in addition to identifying the user. For the above system to output more than 30 bits of entropy (as ours does), it must be that $d > 10$. After M observed responses, the adversary has attack probability of $\left(\frac{1}{10} + \frac{9dM}{10N}\right)^d$. As their security threshold is 10^{-7} and goes as $1-5^{-d}$, to be robust to even a single answer being observed, this system requires $N > 9dM > 90$.[19]

[19] Our system, if instantiated with $N = 16$, $|K| = 5$ and $|D| = 4$, achieves this threshold; raising $|K|$ to 6 as in our instantiation lowers our entropy moderately. Both systems quickly fall below any reasonable threshold. Our system could be

While only d of those need to be interacted with, 10 choices is enough to warrant a menu or keypad UI element, slowing response time.

If we change the system to have 5 "colors" (the four cardinal directions and blank, as in our system), then $d > 13$ (with an expected $4/5 * 13 = 10.4$ interactions required from the user) and the security threshold is $(1/5 + 4dM/5N)^{-d}$ by analogy. Making this 10^{-7} requires $N > 116$. To make layout easy, we should use an 11×11 grid of squares: the slide would contain arrows and blanks, and the display just needs to light up the requested cells. On our demo device, each cell would be 5 millimeters on a side. On average, the user will have to scan $11^2/(4/5 * d) \approx 11.5$ cells at a time, so Hick's search time should be on the order of $.200(121/11.5)\log_2[11.5] \approx 7.4$ seconds. Fitts's response time should be on the order of 5.6 seconds. We expect that answering one of these challenges will therefore take between two and three times as long as one of ours (13.0 vs 5.1 seconds). It may be simpler to align and read off the challenges in this scheme, but that comes with increased risk of successful surveillance. Were we to do a user study, it would be interesting to compare the two.

B Adversary Information Gain From Leaked Answers

The constraint that our system never yield (challenge,answer) pairs to the adversary may seem odd. However, we can demonstrate that expanding our system to work in the face of leaked answers to challenges is nontrivial by demonstrating a substantial loss of entropy for each leaked answer. Thus our system requires that the user be able to respond without the adversary's having perfect surveillance whereby they are able to see the pixel values of a challenge. For remote authentication, it should be easy to generate a novel challenge for every interaction, making surveillance-based attempts to recover (challenge,answer) pairs harder than they might otherwise be. Our prototype's situation is harder, as the challenges are necessarily created while the system secrets are available, that is, when the safe is open. An adversary may therefore collect the challenge and wait for the user to answer it; however, taking the same rudimentary precautions one takes with passwords should be sufficient.

Before we can compute the probability of an adversary's successful guess, we need to define a combinatorial relation. We need to count the number of strings of length $M > 0$ drawn from a vocabulary of $k > 0$ symbols which use exactly $0 < i \leq k$ of them. This is $a_M(k,i) \stackrel{\text{def}}{=} \binom{k}{i}\left(i^M - \sum_{j<i}\binom{i}{j}a_M(j,j)\right)$. We will not rigorously prove this, but can sketch the inductive argument. First, note that $\forall M.a_M(1,1) = 1$, which is correct as there is only one unary string of a given length. i^M counts all possible strings for a vocabulary size i, and $\binom{k}{i}$ provides the number of such vocabularies that can be built out of our total vocabulary of size k. i^M multiply counts strings which use fewer than i symbols, and so we must

augmented to generate challenges which contain voids in certain cells; this would slow the adversary's rate of information gain but would also lower the entropy were N held constant. We have not thoroughly analyzed the impact of such a change.

subtract them off; if the undesired string uses $j < i$ symbols, it will be generated in $\binom{i}{j}$ aliases. There are, by induction, $a_M(j, j)$ such miscreant strings.

Having observed $M > 0$ answers to a given cell in a system in which $|D| = |K| - 1$, the probability of an adversary getting the right answer is

$$p(M) = |K|^{-M} \left[1 \cdot a_M(|K|, |K|) + |K|^{-1} \sum_{k=1}^{|K|-1} (k+1)\, a_M(|K|, k) \right].$$

The first term is the probability that our adversary has seen all vocabulary entries for that cell, at which point there is no guessing left. The second term is a sum over the number of entries seen, k. Having observed $k < |K|$ distinct entries, the adversary need not guess if the challenge uses one of these, which will happen with odds $k/|K|$; the adversary otherwise guesses uniformly, getting the right answer with odds $1/(|K| - k)$. All told, $1 \cdot \frac{k}{|K|} + \frac{1}{|K|-k} \frac{|K|-k}{|K|} = \frac{k+1}{|K|}$. Similarly, we can measure the expected entropy of the system:

$$\begin{aligned} H(M) = &- |K|^{-M} \left[0 \cdot a_M(|K|, |K|) \right] \\ &- |K|^{-M} \sum_{k=1}^{|K|-1} a_M(|K|, k) \left[0 + \frac{|K| - k}{|K|} \log_2 \frac{1}{|K| - k} \right] \\ = &\, |K|^{-M-1} \sum_{k=1}^{|K|-1} a_M(|K|, k)\, (|K| - k) \log_2 (|K| - k). \end{aligned}$$

The adversary's guesses are independent for each cell in the challenge, so after M challenges the odds of a successful guess is $p(M)^N$ and the entropy is $NH(M)$; see Table 1.[20] For systems instantiated with $|K| > |D| + 1$, the odds are necessarily higher and the entropies lower.

M	Expected entropy	Expected probability correct guess
0	37.2	6.6×10^{-12}
1	25.6	4.3×10^{-7}
2	17.3	9.4×10^{-5}
3	11.4	2.5×10^{-3}

Table 1: The expectations, after M challenges' answers have been revealed, of remaining entropy (in bits) and probability of the adversary's first guess being correct. The values here are for a system instantiated with $N = 16$, $|K| = 5$, $|D| = 4$.

[20] Note that even $M = 1$ is sufficient for the adversary to win the game, as the probability of success is dramatically increased. However, in a traditional password scheme, the probability of success after $M = 1$ revealed answers is 1. While the expected attack odds under our system will never hit 1, it rapidly falls below any reasonable security threshold, and so we would still recommend re-keying after discovered surveillance.

Bibliography

[1] Abadi, M., Warinschi, B.: Password-based encryption analyzed. In: Caires, L., Italiano, G., Monteiro, L., Palamidessi, C., Yung, M. (eds.) Automata, Languages and Programming, Lecture Notes in Computer Science, vol. 3580, pp. 664–676. Springer Berlin / Heidelberg (2005)

[2] Abeni, P., Baltatu, M., D'Alessandro, R.: User authentication based on face recognition with support vector machines. In: CRV '06: Proceedings of the The 3rd Canadian Conference on Computer and Robot Vision. p. 42. IEEE Computer Society (2006)

[3] Ateniese, G., Blundo, C., De Santis, A., Stinson, D.R.: Visual cryptography for general access structures. Inf. Comput. 129(2), 86–106 (1996)

[4] Ateniese, G., Blundo, C., Santis, A.D., Stinson, D.R.: Extended capabilities for visual cryptography. Theor. Comput. Sci. 250(1-2), 143–161 (2001)

[5] Chang, C.C., Yu, T.X.: Sharing a secret gray image in multiple images. In: Cyber Worlds, 2002. Proceedings. First International Symposium on. pp. 230 – 237 (2002)

[6] Cimato, S., De Prisco, R., De Santis, A.: Colored visual cryptography without color darkening. Theor. Comput. Sci. 374(1-3), 261–276 (2007)

[7] Dhamija, R., Perrig, A.: Déjà vu: a user study using images for authentication. In: SSYM'00: Proceedings of the 9th conference on USENIX Security Symposium. pp. 4–4. USENIX Association (2000)

[8] Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. 38, 97–139 (2008)

[9] Farzin, H., Abrishami-Moghaddam, H., Moin, M.S.: A novel retinal identification system. In: EURASIP Journal on Advances in Signal Processing. vol. 2008, p. 10 (2008)

[10] Fitts, P.M.: The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology 47(6), 381391 (1954)

[11] Greveler, U.: VTANs - eine anwendung visueller kryptographie in der onlinesicherheit. In: GI Jahrestagung (2)'07. pp. 210–214 (2007)

[12] Hick, W.E.: On the rate of gain of information. Quarterly Journal of Experimental Psychology (4), 11–26 (1952)

[13] Hou, Y.C.: Visual cryptography for color images. Pattern Recognition 36(7), 1619 – 1629 (2003)

[14] Jain, L.e.a. (ed.): Intelligent Biometric Techniques in Fingerprint and Face Recognition. CRC Press (1999)

[15] Jin, D., Yan, W.Q., Kankanhalli, M.S.: Progressive color visual cryptography. Journal of Electronic Imaging 14(3), 033019 (2005)

[16] Kim, M.R., Park, J.H., Zheng, Y.: Human-machine identification using visual cryptography. in Proc. The 6th IEEE Int. Workshop on Intelligent Signal Processing and Communication Systems pp. 178–182 (1998)

[17] Laboratories, R.: Pkcs #5: Password-based cryptography standard, v2.0 (1999), ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2-0.pdf
[18] Lenovo: Thinkvantage® client security solution, http://www.pc.ibm.com/us/think/thinkvantagetech/security.html
[19] Ltda., A.S.: Fingerauth password manager, http://www.fingerauth.com/
[20] Mackenzie, S.I., Soukoreff, W.R.: Text entry for mobile computing: Models and methods, theory and practice. Human-Computer Interaction 17(2 & 3), 147–198 (2002)
[21] Naor, M., Shamir, A.: Visual cryptography. Tech. rep. (1994)
[22] Naor, M., Pinkas, B.: Visual authentication and identification. In: CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology. pp. 322–336. Springer-Verlag (1997)
[23] OpenIntents: OI Safe, http://www.openintents.org/en/node/205
[24] Paul, N., Evans, D., Rubin, A.D., Wallach, D.S.: Authentication for remote voting. In: Workshop on Human-Computer Interaction and Security Systems (2003)
[25] Pavaday, N., Soyjaudah, K.: A comparative study of secret code variants in terms of keystroke dynamics. pp. 133 –140 (2008)
[26] Admit One Security: Keystroke dynamics, http://www.biopassword.com/keystroke_dynamics_advantages.asp
[27] Yang, C.N., Laih, C.S.: New colored visual secret sharing schemes. Des. Codes Cryptography 20(3), 325–336 (2000)
[28] Youmaran, R., Adler, A., Miri, A.: An improved visual cryptography scheme for secret hiding. In: Communications, 2006 23rd Biennial Symposium on. pp. 340 –343 (2006)