Human Public-Key Encryption

Houda Ferradi, Rémi Géraud, and David Naccache

École normale supérieure,
Information Security Group,
45 rue d'Ulm, F-75230 Paris CEDEX 05, France
given_name.family_name@ens.fr

Abstract. This paper proposes a public-key cryptosystem and a short password encryption mode, where traditional hardness assumptions are replaced by specific refinements of the CAPTCHA concept called Decisional and Existential CAPTCHAs.

The public-key encryption method, achieving 128-bit security, typically requires from the sender to solve one CAPTCHA. The receiver does not need to resort to any human aid.

A second symmetric encryption method allows to encrypt messages using very short passwords shared between the sender and the receiver. Here, a simple 5-character alphanumeric password provides sufficient security for all practical purposes.

We conjecture that the automatic construction of Decisional and Existential CAPTCHAs is possible and provide candidate ideas for their implementation.

Introduction

CAPTCHAs[1] [vABHL03] are problems that are hard to solve by computers, while being at the reach of most untrained humans. There might be many reasons why, at a particular time, a given type of CAPTCHA is considered hard for computers. The automated solving of CAPTCHAs may either require more computational power than is available, or algorithms have yet to be invented. It might well be that computers are inherently less efficient, or even incapable, at some tasks than human beings. Whichever the cause, several candidate CAPTCHAs are widely used throughout the Internet to keep robots at bay, or at least slow them down (e.g. [EDHS07, CGJ+08, vAMM+07, CB03, NASK14, SHL+10]).

Most CAPTCHAs are used as human-interaction proofs [BL05] but their full potential as cryptographic primitives has not been leveraged so far despite a few exploratory papers. Early attempts [Dzi10, CHS06, vABHL03, CHS05] faced the inherent difficulty of *malleability*: given a CAPTCHA Q, an adversary could generate Q', whose solution gives a solution to Q. Thus the security of such constructions could only be evaluated against unrealistic "conservative adversaries" [KOPW13]. All in all, we propose to fill the gap by providing a finer

[1] "Completely Automated Public Turing test to Tell Computers and Humans Apart".

taxonomy of CAPTCHAs as well as cryptosystems based on them, which can reach real-life security standards.

The organisation of this paper is as follows: Section 1 defines the classes of problems we are interested in, and estimates how many of those problems can be solved per time unit. We then refine the classical CAPTCHA concept into Decisional and Existential CAPTCHAs. Section 2 describes how to implement public-key encryption using Decisional CAPTCHAs; Section 3 describes a short password-based encryption mode that uses Existential CAPTCHAs to wrap high-entropy keys. Section 4 presents Decisional and Existential CAPTCHA candidates.

1 Preliminaries and Definitions

1.1 CAPTCHA Problems

Let $\mathcal{Q}$ be a class of problem instances, $\mathcal{A}$ a class of answers, and S a relation such that $S(Q, A)$ expresses the fact that "$A \in \mathcal{A}$ is a solution of $Q \in \mathcal{Q}$". Solving an instance Q of problem $\mathcal{Q}$ means exhibiting an $A \in \mathcal{A}$ such that $S(Q, A)$. We assume that for each problem there is one and only one solution, i.e. that S is bijective. This formal setting (similar to [KOPW13, CHS06]) allows us to provide more precise definitions.

Because CAPTCHAs involve humans and considerations about the state of technology, we do not pretend to provide formal mathematical definitions but rather clarifying definitional statements.

Definition 1 (Informal). *A given problem $\mathcal{Q} \in \mathsf{CP}$ (CAPTCHA Problem) if no known algorithm can solve a generic instance $Q \in \mathcal{Q}$ with non-negligible advantage over $1/|\mathcal{A}|$, which is the probability to answer Q correctly at random; yet most humans can provide the solution A to a random $Q \in_R \mathcal{Q}$ with very high probability in reasonable time.*

In Definition 1, it is worth pointing out that future algorithms might turn out to solve efficiently some problems that evade today's computers' reach. As such, CP is not so much a complexity class as it is a statement about technology at any given point in time.

There exist today several approaches to building CAPTCHAs, based for instance on deformed word recognition, verbal tests, logic tests or image-based tasks. We are chiefly interested in those tests that can be automatically generated.

We extend CP in two ways:

Definition 2 (Informal). *A given problem $\mathcal{Q} \in \mathsf{DCP}$ (Decisional CP) if $\mathcal{Q} \in \mathsf{CP}$ and, given a random instance $Q \in_R \mathcal{Q}$ and a purported solution A to Q, no known algorithm can decide whether A is a solution to Q, i.e. evaluate $S(Q, A)$, with non-negligible advantage over $1/|\mathcal{A}|$; while humans can determine with high probability $S(Q, A)$ in reasonable time.*

Finally, we introduce a further class of problems:

Definition 3 (Informal). *Let $\overline{\mathcal{Q}} \notin \mathsf{CP}$ be a set of "decoy data" which are not CAPTCHAs. A given problem $\mathcal{Q} \in \mathsf{ECP}$ (Existential CP) if $\mathcal{Q} \in \mathsf{CP}$ and, given a generic instance $Q \in \mathcal{Q}$ or a decoy $Q \in \overline{\mathcal{Q}}$, no known algorithm can decide whether $Q \in \mathcal{Q}$ with non-negligible advantage over $|\mathcal{Q}|/|\mathcal{Q} \cup \overline{\mathcal{Q}}|$; while humans can decide correctly if $Q \in \mathcal{Q}$ or $Q \in \overline{\mathcal{Q}}$ in reasonable time with high probability.*

Remark 1. Definition 3 depends on the set $\overline{\mathcal{Q}}$. We silently assume that, for a given problem $\mathcal{Q}$, an appropriate $\overline{\mathcal{Q}}$ is chosen. This choice makes no difference.

When $\mathcal{Q}$ is not exhaustively searchable, Definition 3 means that a computer cannot decide whether a given Q is a CAPTCHA or not, let alone solve Q if Q is indeed a CAPTCHA.

Remark 2. Definition 3 can be reformulated similarly to the IND-CPA [NY90] security game: we pick a random bit b and provide the adversary with Q_b, where $Q_0 \in \mathcal{Q}$ and $Q_1 \in \overline{\mathcal{Q}}$. The adversary is expected to guess b no better than at random unless it resorts to human aid.

Remark 3. $\mathsf{ECP}, \mathsf{DCP} \subseteq \mathsf{CP}$, but there is no inclusion of ECP in DCP or *vice versa*. Informally, CP is about finding an answer, DCP is about checking an answer, and ECP is about recognizing a question.

Remark 4. Solving a problem $Q \in \mathsf{CP}$ is either done using computers which by definition provide unreliable answers at best; or by asking a human to solve Q – effectively an oracle. However, there is a limit on the number of solutions humans can provide and on the rate at which humans can solve CAPTCHAs.

Consider a given $\mathcal{Q} \in \mathsf{CP}$ whose generic instances can be solved by a human in reasonable time. Let us estimate an upper bound b on the number of instances of $\mathcal{Q}$ that a human may solve during a lifetime. Assuming a solving rate of 10 instances per minute, and working age of 15–75 years, spent exclusively solving such problems, we get $b \sim 10^8$. Taking into account sleep and minimal life support activities, b can be brought down to $\sim 10^7$.

There should be no measurable difference between solving a problem in CP or in DCP, however it might be slightly simpler (and therefore quicker) for humans to *identify* whether a problem is a CAPTCHA without actually solving it. For simplicity we can assume that CAPTCHA recognition is ten times faster than CAPTCHA resolution.

There are various estimations on the cost of having humans solve CAPTCHAs. Some websites offer to solve 1000 CAPTCHAs for a dollar[2]. Of course, the oracle may employ more than one human, and be proportionally faster, but also proportionally more expensive.

[2] At a first glance, the previous figures imply that breaking a public-key (as defined in the next section) would only cost \$$10^4$. We make the economic nonlinearity conjecture there are no \$$10^4$ service suppliers allowing the scaling-up of this attack. In other words, if the solving demand d increases so will the price. We have no data allowing to quantify price(d).

2 Human Public-Key Encryption

We now describe a public-key cryptosystem using problems in DCP. Let $\mathcal{Q} \in \mathsf{DCP}$. We denote by $H(m)$ a cryptographic hash function (e.g. SHA-3) and by $E_k(m)$ a block cipher (e.g. AES-128). Here, m is the plaintext sent by Bob to Alice.

- *Key-pair generation*: The public key pk is a list of b instances of $\mathcal{Q}$

  $$\mathsf{pk} = \{Q_1, \dots, Q_b\}$$

  The private key is the set of solutions (in the CP sense) to the Q_i:

  $$\mathsf{sk} = \{A_1, \dots, A_b\}$$

  i.e. for $1 \leq i \leq b$, $S(Q_i, A_i)$ holds true.
- *Encryption*: Bob wants to send m to Alice. Bob picks k random problems $\{Q_{i_1}, \dots, Q_{i_k}\}$ from Alice's pk, and solves them[3]. Let $\sigma \leftarrow \{A_{i_1}, \dots, A_{i_k}\}$ and $\alpha \leftarrow \{i_1, \dots, i_k\}$. Bob computes $\kappa \leftarrow H(\alpha)$ and $c \leftarrow E_\kappa(m)$, and sends (σ, c) to Alice.
- *Decryption*: Given σ, Alice identifies the set of indices α and computes $\kappa \leftarrow H(\alpha)$. Alice then uses κ to decrypt c and retrieve m. Decryption does *not* require any human help.

The general idea of this cryptosystem is somewhat similar to Merkle's puzzles [Mer78], however unlike Merkle's puzzle here security *is not quadratic*, thanks to problems in CP not being automatically solvable. We may assume that the A_is are pairwise different to simplify analysis.

Remark 5. Indeed if $\mathcal{Q} \in \mathsf{CP}$ it might be the case that a machine could decide if given A, Q the relation $S(A, Q)$ holds *without* solving Q. Hence $\mathcal{Q}$ must belong to DCP.

Remark 6. A brute-force attacker will exhaust all $\binom{b}{k}$ possible values of α. Hence $\binom{b}{k}$ should be large enough. Given that $b \sim 10^7$ or $b \sim 10^8$, it appears that $k = 6$ provides at least 128-bit security.

Remark 7. The main drawback of the proposed protocol is the size of pk. Assuming that each Q_i can be stored in 20 bytes, a pk corresponding to $b \sim 10^8$ would require 2 GB. However, given that CAPTCHAs are usually visual problems, it is reasonable to assume that pk might turn out to be compressible.

Remark 8. Instead of sending back the solutions σ in clear, Bob could hash them individually. Hashing would only make sense as long as solutions have enough entropy to resist exhaustive search.

[3] Here Bob must resort to human aid to solve $\{Q_{i_1}, \dots, Q_{i_k}\}$.

Remark 9. It is possible to leverage the DCP nature of the Q_is in the following way: instead of sending a random permutation of solutions, Bob could interleave into the permutation d random values (decoy answers). Alice would spot the positions of these decoy answers and both Alice and Bob would generate $\alpha = \{i_1, \ldots, i_k, j_1, \ldots, j_d\}$ where j_d are the positions of decoys. Subsequently, security will grow to $\binom{b}{k+d}/d!$. This is particularly interesting since for $b = 10^7$, $k = 1$ and $d = 6$ we exceed 128-bit security. In other words, all the sender has to do is to *solve one CAPTCHA*.

Entropy can be further increased by allowing d to vary between two small bounds. In that case the precise (per session) value of d is unknown to the attacker.

3 Short Password-Based Encryption

In the following scenario Alice and Bob share a short password w. We will show how a message m can be securely sent from Alice to Bob using *only* w. This is particularly suited to mobile devices in which storing keys is risky.

Let $\mathcal{Q} \in \mathsf{ECP} \cap \mathsf{DCP}$.

- Alice generates a full size[4] key R and uses it to encrypt m, yielding $c_0 \leftarrow E_{0|R}(m)$. She generates an instance $Q \in \mathcal{Q}$, such that $S(P, R)$. Alice computes $c_1 \leftarrow E_{1|w}(P)$ and sends (c_0, c_1) to Bob.
- Bob uses w to decrypt c_1, and solves P. He thus gets the key R that decrypts c_0.

An adversary therefore faces the choice of either "attacking Shannon" or "attacking Turing", i.e. either automatically exhaust R, or humanly exhaust w. Each candidate w yields a corresponding P that cannot be computationally identified as a CAPTCHA. The adversary must hence resort to humans to deal with every possible candidate password.

Assuming that CAPTCHA identification by humans is ten times faster than CAPTCHA resolution, it appears that w can be a 5-character alphanumeric code[5].

Remark 10. R must have enough entropy bits to provide an acceptable security level. R can be generated automatically on the user's behalf. As we write these lines we do now know if there exists $\mathcal{Q} \in \mathsf{ECP} \cap \mathsf{DCP}$ admitting 128-bits answers. If such $\mathcal{Q}$s do not exist, R could be assembled from several problem instances.

Remark 11. In the above we assume that R is generated first, and then embedded into the solution of a problem instance P. All we require from R is to provide sufficient entropy for secure block cipher encryption. Hence, it might be easier to generate P first, and collect R afterwards.

[4] e.g. 128-bit.

[5] There are 64 alphanumeric characters, and $64^5 > 10 \times b$.

Remark 12. The main burden resting on Bob's shoulders might not be the solving on P but the keying of the answer R. 128 bits are encoded as 22 alphanumeric characters. Inputting R is hence approximately equivalent to the typing effort required to input a credit card information into e-commerce website interfaces[6]. Alternatively, Bob may as well read the solution R to a speech-to-text interface that would convert R into digital form.

Remark 13. $Q \in \mathsf{ECP} \cap \mathsf{DCP}$ is necessary because the adversary may partially solve Q and continue using exhaustive search. Under such circumstances, c_0 serves as a clue helping the attacker to solve Q. If $Q \in \mathsf{ECP} \cap \mathsf{DCP}$, such a scenario is avoided.

4 DCP and ECP Candidate Instances

The above constructions assume that ECP and DCP instances exist and are easy to generate. Because ECP and DCP depend both on humans and on the status of technology, it is difficult to "prove" the feasibility of the proposed protocols.

We hence propose a DCP candidate an ECP candidates and submit them to public scrutiny.

4.1 DCP candidate

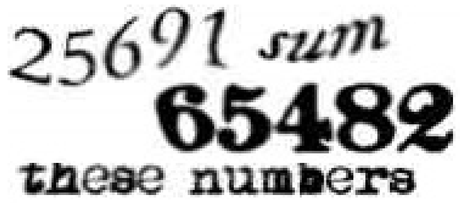

Fig. 1. A DCP candidate constructed from an existing CP.

As a simple way to generate DCPs, we propose to start from a standard CP (e.g. a number recognition problem) and ask a further question about the answer. The further question should be such that its answer may correspond to numerous potential contents. For instance, the further question could be whether two sequences of digits recognised in an image Q sum up to $A = 91173$ or not (see Figure 1).

4.2 ECP candidates

This section proposes a few candidate $\mathcal{Q}$ that we conjecture to belong to ECP.

[6] PAN (16 characters), expiry date (4 characters) and a CVV (4 characters).

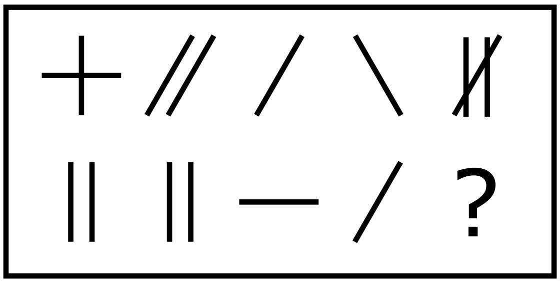

Fig. 2. An instance of a visual-logical task ECP problem. Recognizing objects in this image is insufficient to tell whether there is a solution, nor to compute the solution should there be one.

The first step is to design a task that we think is challenging for computers. Despite recent progress (see e.g. [GBI+13]), computer vision is still expensive and limited. Most computer vision algorithms have to be trained specifically to recognise objects or features of a given kind (dog breeds, handwritten characters, etc.), and fail whenever the task at hand requires more than mere object identification. Even in that case, occlusion, distortion and noise cause drastic performance loss for most techniques. Many CAPTCHAs ideas rely on this to generate problem instances [CLSC05].

Even if image contents can be detected, we can still pose a hard challenge. Indeed, while computers excel at solving logical reasoning questions when those questions are encoded manually as logical formulae, state of the art algorithms fail at even the most basic questions when challenges are presented in visual form. Therefore, solving for instance a visual-logical task is a problem that is at least in DCP (see Figure 2).

Good ECP candidates for cryptographic purposes should be easy to generate, they should have enough possible solutions to thwart exhaustive search attempts, and it should be hard to tell automatically whether there is a solution at all.

Temporal Sequence ECP. The intuition for this candidate is that although computer vision algorithms may reach human accuracy (and even beat it), humans can make use of external knowledge, which provides additional understanding of what is under scrutiny. Here the external knowledge is that real-life events abide by *causality*.

We provide k images (e.g. $k = 5$), each of which is a snapshot of some situation: buying goods, driving a car, dressing up, etc. The order of images is scrambled (some random images may be inserted as decoys) and the problem is to put images back in the correct order. This task, which we call *temporal sequence*, requires the contextual knowledge that some events can only happen after (or before) others. This is illustrated in Figure 3.

We conjecture that the temporal sequence task is both in DCP and in ECP.

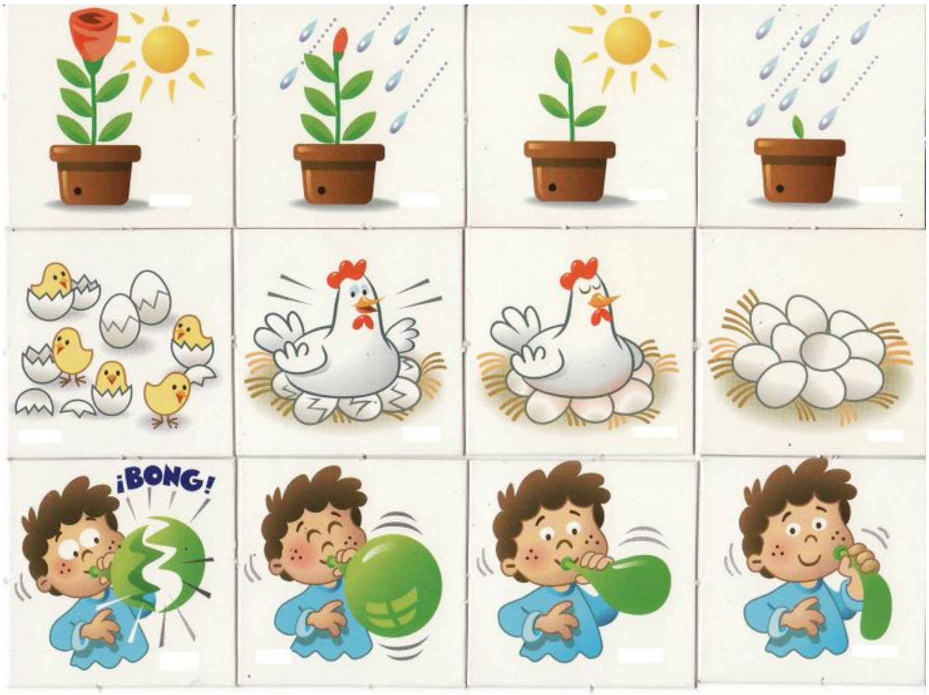

Fig. 3. Three instances of the temporal sequence ECP problem. The problem consists in temporally arranging the pictures.

One drawback of this approach is that to reach an 80-bit security level we need $k = 40$ images[7] which can be unwieldy. This may be solved by using ℓ collections of κ images, and tune ℓ, κ so that $(\kappa!)^\ell > 2^{80}$.

Temporal sequences may be automatically generated from videos, although it is not obvious how to ensure that sequences generated like this are always meaningful to humans.

Visual Letter Recognition ECP. Assume we have a CP problem $\mathcal{Q}$, whose instances can successfully conceal letters (a "one-letter" CAPTCHA). We provide k instances of $Q_1, \ldots, Q_k$ corresponding to answer letters $A_1, \ldots, A_k$, and ask for the alphabetically sorted list of these A_i.

As an example, we would generate instances of $\mathcal{Q}$ for the letters $\{A, M, T, O, B, R\}$, and ask for the solution ABMORT. Under the assumption that $\mathcal{Q} \in$ CP, determining whether a solution exists requires human aid. Therefore we conjecture that this problem belongs to ECP.

A further variant of this idea is illustrated in Figure 4. Note that the visual letter recognition problem is DCP if an only if $\mathcal{Q} \in$ DCP.

[7] There are $k!$ combinations, and $40! > 2^{80}$.

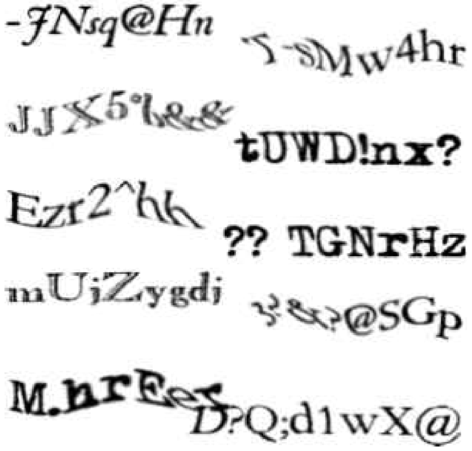

Fig. 4. Visual Letter Recognition ECP: letters are concealed using an existing CP, and one digit is inserted into each sequence of letters. The ECP problem is to reorder the CAPTCHAs in increasing digit order, discarding all non-digit symbols. Here the solution consists in selecting the 4th, 5th, 2nd, 3rd, and 1st images, in that order.

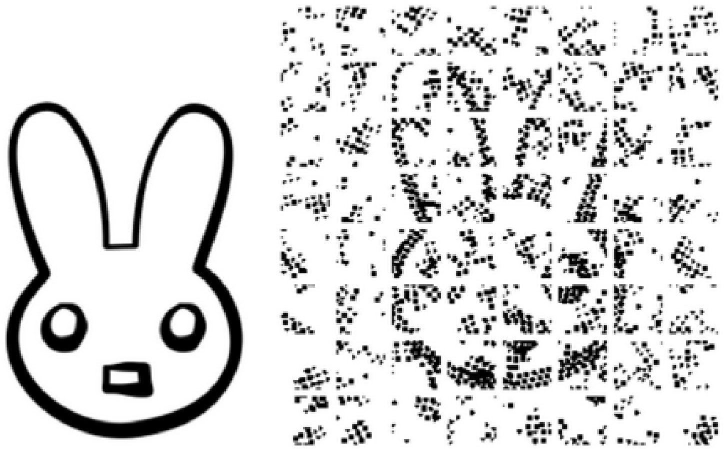

Fig. 5. A honey image ECP. Left: original image; right: $Q_{\ell_{\mathrm{OK}}}$, the transformed image for ℓ_{OK}.

Honey Images ECP. Another candidate problem is inspired by honey encryption [JR14, YKJ+15]. The idea is that any integer $1 \le \ell \le k$ would generate an

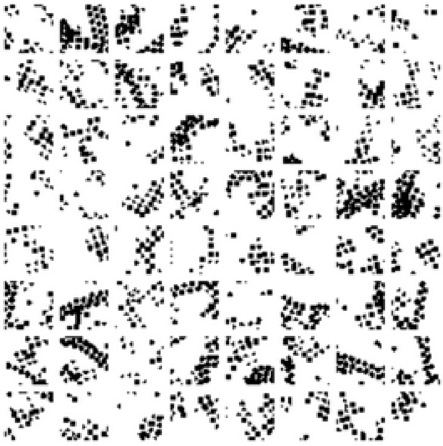

Fig. 6. All values of ℓ other than ℓ_{OK} produce decoys whose statistical properties are conjectured to be indistinguishable from the correct image, with salient features but no real meaning.

image, but that only one value ℓ_{OK} generates a *meaningful* image[8]. All values $\ell \neq \ell_{\mathrm{OK}}$ generate images in a way that makes them indistinguishable from meaningful images. The problem would then be to identify ℓ_{OK}, which we conjecture only humans can do reliably.

The main difficulty is that the notion of indistinguishability is tricky to define for images, and even harder to enforce: humans and computers alike use very specific visual cues to try and perform object recognition, which are hard to capture statistically. Following [YKJ+15], we may try and learn from a dataset how to properly encode images, but this is cumbersome in our context, especially when dealing with a large number of instances.

Our candidate is a simpler embodiment based on the following intuition: using biased noise (i.e. noise that is *not* random), we can elicit pareidolia in computer vision programs. Each candidate value of ℓ would then correspond to some object being recognised – but only one of those is really relevant. We conjecture that only humans are able to pick this relevant object apart.

The authors implemented this idea. We start from a black and white picture of a clearly identifiable object (Figure 5 left, here $A =$ "rabbit"), turn it into a collection of black dots[9] (1). The picture is then cut into blocks which are shuffled and rotated (2). Finally, noise is added, under the form of black dots whose size is distributed as the size of black dots in the original picture (3). The image is then rotated back in place (Figure 5 right) to provide the challenge $Q_{\ell_{\mathrm{OK}}}$.

[8] In the specific case of Figure 5, translation, rotation, mirroring as well as border cropping may also generate the meaningful image corresponding to ℓ_{OK}, but the overall proportion of such images remains negligible.

[9] For instance using an iteratively reweighted Voronoi diagram.

The motivation for this approach is as follows: (1) guarantees that individual pixels contain no information on luminance, and geometric features (lines, gradients and corners) – each dot being circular destroys information about orientation; the shuffling and rotation of blocks in (2) is encoded as an integer ℓ; and (3) inserts decoy features, so that any shuffling/rotation would make geometric features appear (to lure a computer vision algorithm into detecting something).

Now, many decoys $Q_\ell \in \overline{\mathcal{Q}}, \ell \neq \ell_{\mathrm{OK}}$ can be generated easily from this image by shuffling and rotating blocks (Figure 6). Each decoy shares the same statistical properties as the correct (unshuffled) image, but has no recognizable content.

Our conjecture is that the human brain can perceive structures very efficiently and assign meaning to them. Many such structures are irrelevant and inserted so as to fool computer vision algorithms, but the familiar ones are immediately and intuitively grasped by humans. Consequently, although the original picture is severely deteriorated, we conjecture that it should still be possible for humans to tell noise and signal apart and identify correctly the contents of this image.

5 Further Applications

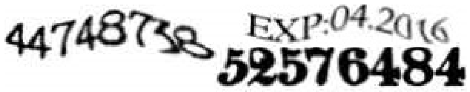

Fig. 7. Credit card PAN and expiry date, stored as a DCP instance.

Beyond their cryptographic interest, DCP and ECP tasks may have interesting applications in their own right.

One such application is the following: users may wish to store sensitive data as a DCP instance, for instance credit card information, instead of plaintext. Indeed, attackers often browse their victims' computers looking for credit card information, which is easy to recognize automatically. By storing credentials in an ECP the attacker's task can be made harder.

References

BL05. Henry S. Baird and Daniel P. Lopresti, editors. *Human Interactive Proofs, Second International Workshop, HIP 2005, Bethlehem, PA, USA, May 19-20, 2005, Proceedings*, volume 3517 of *Lecture Notes in Computer Science*. Springer, 2005.

CB03. Monica Chew and Henry S. Baird. Baffletext: a human interactive proof. In Tapas Kanungo, Elisa H. Barney Smith, Jianying Hu, and Paul B. Kantor, editors, *Document Recognition and Retrieval X, 22-23 January 2003, Santa Clara, California, USA, Proceedings*, volume 5010 of *SPIE Proceedings*, pages 305–316. SPIE, 2003.

CGJ$^+$08. Richard Chow, Philippe Golle, Markus Jakobsson, Lusha Wang, and XiaoFeng Wang. Making captchas clickable. In Mirjana Spasojevic and Mark D. Corner, editors, *Proceedings of the 9th Workshop on Mobile Computing Systems and Applications, HotMobile 2008, Napa Valley, California, USA, February 25-26, 2008*, pages 91–94. ACM, 2008.

CHS05. Ran Canetti, Shai Halevi, and Michael Steiner. Hardness amplification of weakly verifiable puzzles. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 17–33. Springer, 2005.

CHS06. Ran Canetti, Shai Halevi, and Michael Steiner. Mitigating dictionary attacks on password-protected local storage. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 160–179. Springer, 2006.

CLSC05. Kumar Chellapilla, Kevin Larson, Patrice Y. Simard, and Mary Czerwinski. Designing human friendly human interaction proofs (hips). In Gerrit C. van der Veer and Carolyn Gale, editors, *Proceedings of the 2005 Conference on Human Factors in Computing Systems, CHI 2005, Portland, Oregon, USA, April 2-7, 2005*, pages 711–720. ACM, 2005.

Dzi10. Stefan Dziembowski. How to pair with a human. In Juan A. Garay and Roberto De Prisco, editors, *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*, volume 6280 of *Lecture Notes in Computer Science*, pages 200–218. Springer, 2010.

EDHS07. Jeremy Elson, John R. Douceur, Jon Howell, and Jared Saul. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 366–374. ACM, 2007.

GBI$^+$13. Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *CoRR*, abs/1312.6082, 2013.

JR14. Ari Juels and Thomas Ristenpart. Honey encryption: Security beyond the brute-force bound. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 293–310. Springer, 2014.

KOPW13. Abishek Kumarasubramanian, Rafail Ostrovsky, Omkant Pandey, and Akshay Wadia. Cryptography using captcha puzzles. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, volume 7778 of *Lecture Notes in Computer Science*, pages 89–106. Springer, 2013.

Mer78. Ralph C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978.

NASK14. Mir Tafseer Nayeem, Md. Mamunur Rashid Akand, Nazmus Sakib, and Md. Wasi Ul Kabir. Design of a human interaction proof (HIP) using human cognition in contextual natural conversation. In *IEEE 13th International Conference on Cognitive Informatics and Cognitive Computing, ICCI*CC 2014, London, UK, August 18-20, 2014*, pages 146–154. IEEE, 2014.

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437. ACM, 1990.

SHL$^+$10. Graig Sauer, Jonathan Holman, Jonathan Lazar, Harry Hochheiser, and Jinjuan Feng. Accessible privacy and security: a universally usable human-interaction proof tool. *Universal Access in the Information Society*, 9(3):239–248, 2010.

vABHL03. Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: using hard AI problems for security. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2003.

vAMM$^+$07. Luis von Ahn, Ben Maurer, Colin McMillen, David Abraham, and Manuel Blum. Google reCAPTCHA: https://developers.google.com/recaptcha, 2007.

YKJ$^+$15. Ji Won Yoon, Hyoungshick Kim, Hyun-Ju Jo, Hyelim Lee, and Kwangsu Lee. Visual honey encryption: Application to steganography. In Adnan M. Alattar, Jessica J. Fridrich, Ned M. Smith, and Pedro Comesaña Alfaro, editors, *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec 2015, Portland, OR, USA, June 17 - 19, 2015*, pages 65–74. ACM, 2015.