Visual CAPTCHAs for Document Authentication

Igor Fischer and Thorsten Herfet
Telecommunications Lab, Saarland University
Campus Building C6 3, P.O. Box 151150
66041 Saarbrücken, Germany
Email: {fischer, herfet}@nt.uni-saarland.de

***Abstract*— Visual CAPCTCHAs, like gimpy images, are commonly used to screen human users from automated computer scripts. We propose using them for direct, visual authentication of digital documents. The basic assumption is that if it is hard or impossible for the computer to recognize a document, then it cannot manipulate it. If such a document is still recognizable by a human, he or she can be confident that it is authentic, or at least that no automated process has manipulated it. Such authentication is highly desirable in the context of digital signatures, which, due to the computational complexity, have to be produced by a computer. It can be a specialized, trusted cryptographic computer, e.g. a smartcard, but the path to it is generally not secure. Visual CAPTCHAs can be used for securing the path without the need for additional hardware.**

I. Introduction

Digital signatures [1], made public almost three decades ago, have a number of advantages compared to their classical, hand-written counterparts: they are much harder to forge, can be detached from the document, guarantee the integrity of every bit even in a 100-page document, and — maybe the most practical advantage — leave the document in the digital form, allowing for its transmission by electronic means and for further processing. It can be safely assumed that in today's developed world the majority of documents — letters, contracts, school transcripts etc. — is produced entirely or mainly on computers. Still, most documents today are authenticated in the classical way, by hand-signing the hardcopy. The reason is, we believe, that in most cases the whole setup for *producing and signing* a digital document cannot be trusted.

Being a cryptographic method, digital signatures require complex and cumbersome computations on large numbers. The computations are almost impossible to be performed by humans and need to be done by computers. On the other hand, common, general purpose personal computers are susceptible to various attacks — viruses, Trojan horses, worms, phishing and other hacker attacks — and generally cannot be trusted. Attaching a trusted hardware to the PC, like a smartcard, to perform the cryptography, does not solve the problem, either. As long as the PC is vulnerable, the document can be tampered with inside it, on its way to the smartcard, and the user has no way of noticing it. It is perfectly possible to produce a document on the PC and it is also possible to produce a valid and secure digital signature of a document, once the smartcard (or other secure module) receives it. The problem is to ensure that it is the same document in both cases.

Suppose, for example, that you are using your PC for writing a contract, in which you agree to pay, say, 1000 USD to Bob. You let your personal smartcard, attached to the PC, digitally sign the contract and you send it to Bob. Unfortunately, you have a malicious program in your PC (maybe even planted by Bob, who knows?) which changes the number 1000 into 2000 in the document as it travels towards the smartcard. The smartcard will produce a perfectly valid digital signature. Because a digital signature is simply a large, random-looking binary number, as a human you cannot see that it corresponds to a different document than the one you wrote. As you send the document along with the signature to Bob, the malicious program again changes 1000 into 2000. Now Bob is happy, he receives a contract, digitally signed by you, where you agree to pay double of what he has hoped for.

Three solutions have been proposed: equipping the external hardware with display capabilities, making the whole PC secure, or using visual cryptography for checking that the document arrived unchanged to the module performing the cryptography.

The first solution is typically a smartcard reader, produced by some trusted entity, with a display and possibly a keypad. Such devices are commonly used in electronic payment: the price and possibly other information is shown on the display and the user authorizes the payment by inserting the payment card and typing in his or her PIN. The method could, in principle, be used for digitally signing documents: before signing, the document would be shown on the trusted reader's display and the user would have to authorize the signature by typing in the PIN. In practice this can work only for very simple and short documents, due to limited displaying capabilities of common smartcard readers. Equipping the readers with a big display, just for the purpose of secure digital signing, is technically possible, but not economical.

An economically more viable solution is envisioned by the Trusted Computing Group (TCG). The idea is to make the whole PC secure, including its peripherals. On the other hand, it should remain a general-purpose computer, where the user can still install new software and hardware. To remain affordable and compatible with the existing computers, the computer must not differ much from today's PCs, otherwise it would not be accepted by the market. The solution includes a minor hardware modification, adding a specialized cryptographic chip called "trusted platform module" (TPM) to the mainboard, and adjusting the system software (BIOS, OS, ...) to use the TPM to check the system integrity. The system

works through so-called chain of trust: the first program executed after power-on (typically from the BIOS) has to be unconditionally trusted. It computes hash values of the next programs to be executed (e.g. the operating system) and of measurable properties of the hardware to be used, like hard disk ID etc. It produces their digital signatures and checks them using the TPM. If the checks pass, the control is passed to the next program (e.g. OS loader), which repeats the process for its successors. The chain unfolds up to application programs.

Trusted computing does not guarantee that the installed hard- and software are not malicious, but it notifies the user of any changes to the system. If the computer has been delivered in a trusted state, it will remain in it as long as the user does not install anything new or trusts everything he or she installs. This trust can be established through signatures by trusted entities, which are checked by the TPM. TPMs are already available, but whole systems based on them are not. It is not yet known how the actual implementation will look like, since potential customers have expressed some doubts. They fear, for example, that through TC manufacturers might coerce users into using or not using some software or hardware, simply by declaring it "not trusted". TCG best practice manual [2] denounces such abuses of the technology, but cannot prevent it. Also, the flow of personal information in such systems is not transparent and it is feared that they might be unwillingly disclosed [3]. It is also not clear how backups would work and what happens in the case of a hardware failure.

The third solution attempts to secure the path between the secure cryptographic module and the human visual perception. It is remarkably low-tech, requiring almost no additional hardware. Visual authentication [4] is a method that was initially introduced for secure electronic payments over non-trusted terminals. The method is based on visual cryptography [5], which is generally a visual implementation of secret sharing [6], but for the authentication purposes can be looked at as a symmetric cryptographic method. The "plaintext" is a black-and-white image. This image is first oversampled, typically by a factor of 2 along both axes, and then transformed, so that each 2×2 square (corresponding to a pixel in the original image) is made completely black if the corresponding original pixel was black, or, if the original pixel was white, contains two random black pixels and the other two white. Such squares visually appear grey. The actual encryption is performed by splitting the transformed image, square-by-square, into two half-images ("shares"). Each grey square is split into two identical squares, and each black square into two complementary. Each share then looks like a uniform distribution of black and white pixels. However, if the shares were printed on transparencies, their superposition would produce the transformed image, from which the original one is still visually recognizable. One of the shares can be fixed in advance among the communication parties and considered "key", and the other, "cyphertext" computed from the key and the document.

Visual cryptography cannot be used directly for message authentication if the attacker knows the message, as can be assumed in the case of a vulnerable PC. From the message and the cyphertext he or she can deduce the key and completely forge the document. Instead, it was proposed to enrich the cyphertext with information not contained in the message, but known to the user, for example by enlarging the key and the cyphertext and requiring that the decrypted image appears in a predefined area. A more recent approach [7] proposes keeping the physical size unchanged, but incorporating some secret information, a "watermark" into the cyphertext. Such watermark is unknown and thus not noticeable by the attacker, but, in the decrypted document, visually recognizable by the user. The drawbacks of visual authentication are that it is applicable only to black-and-white documents, requires a new "key" (transparency) for every document page and needs the cyphertext and the key be precisely physically aligned for the "plaintext" to reappear. In practice, this usually requires printing the cyphertext on the paper.

The method proposed here is also a type of visual document authentication and secures the path from the cryptographic module to the user's eye. But, instead of encrypting the document and requiring the user to use an auxiliary device (like the key transparency) for decrypting it, the document is transformed in a way that makes it practically impossible to recognize for a computer, but not for a human observer. Unlike the above one, this method is not perfectly (i.e. theoretically proven) secure, but sufficiently secure for all practical purposes, considering the state-of-the-art of available pattern recognition technologies. The method is much easier to use, works directly on the screen and does not need external tools, like transparencies.

II. Proposed method

In this section we

1) Review CAPTCHAs as a method for obtaining security,
2) Describe the assumed setup for applying the proposed method, and
3) Describe a typical usage scenario of the method.

The proposed method is envisioned to secure visual documents on their way from a trusted module (a smartcard, a remote trusted computer, a tamper-resistant software [8] etc.) to the human observer against tampering. It is based on a visual CAPTCHA [9], which exploits the fact that some problems, which are easily solved by humans, cannot be solved by current algorithms. A CAPTCHA is basically a program that can generate and grade tests based on such problems and thus distinguish between human and computer provers. The idea is widely used on the Internet to block automated access to services, like e-mail accounts, search engines, and similar. A very common such test is EZ-gimpy, a word displayed in a distorted way, which a human can easily recognize, but computers presumably cannot. A more complicated version, Gimpy, contains several (typically eight) words, from which the prover has to recognize a subset (typically three). Both problems can in the meantime be automatically solved: Gimpy in about 1/3 of the cases and EZ-gimpy even in 92% [10], but

it still remains an accepted fact that humans by far outperform computers in recognition of visual patterns.

In the proposed method, we apply more complex visual CAPTCHAs for obscuring visual documents, which, to our knowledge, have not yet been broken. The trusted module transforms the document's appearance, so that the human can still recognize it, but not the computer. The transformation can include inclining the document in 3D, dropping shadows, projecting it onto uneven and possibly morphing surfaces, letting moving spotlights and magnifying lenses cover it, displaying patterns in the background or semi-transparent in front of the document, and similar. Automated attackers (computer programs) cannot tamper with the document because of the high pattern recognition complexity involved in reconstructing the document transformation and separating it from the document. Human attackers might be able to roughly recognize the transformation (e.g. estimate the inclination), but not to forge it in the limited time. The primary application of the method is in digital signatures, but it can be used generally for a kind of steganography, hiding information from computers. In digital signatures, the purpose is to ensure that the trusted module, which performs the signing, has received the document unmodified.

The assumed setup is the following: The user (a human) produces a document on an ordinary general-purpose computer. The computer itself is not trusted to safeguard the integrity of the document, but there is a trusted module attached to it or included in it. The module has no input/output capabilities except for electronic communication with the computer. Beside its primary function, like producing digital signatures, the module is capable of transforming the visual appearance of the documents it receives. Actually, the module produces a digitized appearance, e.g. a digital video stream, which is sent back to the computer for displaying on its screen. In case of multi-page documents, each page is processed separately.

For security reasons, each specific transformation is used only for one document, like a one-time pad. The user and the module share an enumerated list, agreed in advance, of transformations which the module will be applying for the documents. Each transformation has a short alphanumerical code associated with it, by which the user authorizes the signing. The code, which is also used only once, has roughly the meaning of the Transaction Authorization Number (TAN), commonly used for online banking in Europe. In practice, such a list would be distributed by a trusted authority, e.g. electonically and encrypted for the module and by mail for humans.

To digitally sign a document, the user sends it to the trusted module. The module transforms the visual appearance of the document as it received it (remember that the document could have been tampered with on the way). It sends the transformed appearance back to the computer, which displays it on the screen. The user looks at the transformed document, checks if its content has not been modified and if the transformation is as expected, i.e. the next one from the transformations list. If both conditions are satisfied, the user is confident that the document has arrived unchanged to the module. He or she authorizes the signing of the document by typing in the authorization code associated with the transformation. The module checks if the authorization code corresponds to the applied transformation and, if yes, signs the document and sends the signature back to the user's computer. By deferring the signing until the user authorizes the document it is prevented that an attacker obtains a valid signature even *with* user's knowledge. This is necessary because if the compromised computer is networked the user might not be able to stop the forged signature from being distributed even if he or she notices the forgery.

III. Implementation and discussion

In our prototype, the "trusted module" is implemented as a program running on the user's computer. For real-world application the program can be made tamper-resistant or even moved to external hardware. It receives text documents, renders them as images and visually transforms them before displaying them. The transformations are computationally complex and include coloring, 3D rendering, and animation. Animations are especially practical, because they introduce not only one more degree of freedom, but also the constraint of smoothness. To forge a smooth animation, the attacker must be capable of solving the CAPTCHA on-the-fly, between the frames, or to delay the whole animation — something a cautious user would certainly notice.

Due to required computational power (our implementation consumes 3/4 of a 3.4 GHz Pentium processor) and transmission bandwidth for the animations, smartcards are hardly an option as external trusted devices. We therefore assume that such devices will be more sophisticated and will communicate over a high-speed link, like USB, Ethernet, or wireless.

We have implemented several types of transformations for conveying the document. As the most promising we consider the "Deforming Surface". The document, seen as black-on-transparent image, is first passed through several image-processing filters (shadow-dropping, water ripple, fish-eye lens. . .) and projected onto a 3D surface. The surface itself is morphing between different shapes, which contain some simple alphanumeric codes. The whole surface is inclined in 3D (see Figure 1). Another implemented transformation includes making the document semi-transparent, with an oscillating transparency, and superimposing it with an animated background, where occasionally some code appears. Yet another implementation cuts the document itself into pieces and lets them fly and rotate in 3D, so that they only occasionally come together and produce the original document (Figure 2).

The only competing method we know about, which works for larger documents, is Visual Authentication (Trusted Computing is not yet available and certified smartcard readers have too small displays). The advantage of Visual Authentication is that it is perfectly secure. On the other hand, it is clumsy to use, requires a staple of pre-printed transparencies and perfectly aligning each transparency with the encrypted document. We argue that perfect security — that is, one that cannot be broken, ever — is not needed for our application.

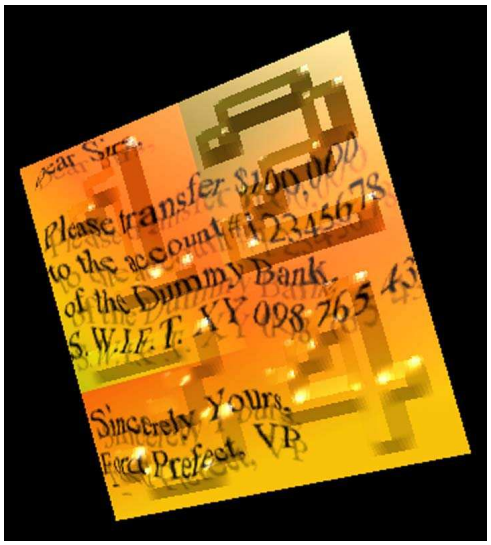

Fig. 1. A snapshot of the "Deforming Surface" animated captcha. The distorted text is projected onto a morphing surface. At the instant, the code "1234" can be recognized on the surface.

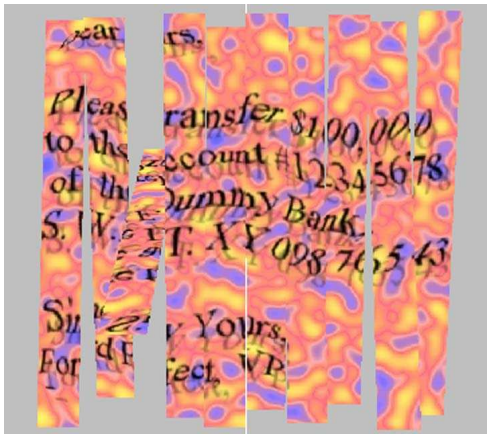

Fig. 2. A snapshot of the "Flying Pieces" animated captcha. The distorted text is projected onto a color patched surface which is cut into stripes. The stripes rotate in 3D and at the instant are close to coming together, so that the document can be recognized

It suffices if the attacker cannot forge the document in the couple of seconds or, in the worst case, minutes the human user needs to check the document integrity and initiate the signing. We are not aware of a pattern recognition algorithm which is capable of breaking an animated CAPTCHA as ours, especially not given the time constraint. Even as pattern recognition algorithms advance, we expect new CAPTCHAs to be invented.

IV. Conclusions and future work

We have presented a method, which:

1) uses animated visual CAPTCHAs
2) for obfuscating human-recognizable text, so that it is not recognizable by computers,
3) with the purpose of securing the transmission path from a trusted module to the human eye,
4) so that humans can check the integrity of documents received by the module. The method also includes
5) a simple challenge-response protocol to prevent the attacker from posing as the human user to obtain services from the module.

The security of the method is currently not quantifiable. Future work is intended in two directions. One is to explore how complex the transformations can get before they prevent humans from recognizing the document. This question can only be solved empirically, by testing the method using different CAPTCHAs on large number of human observers. The complexity itself is a subjective value, depending on the human observer. Ideally, a kind of perceptual model would evolve, which would allow for quantifying the CAPTCHA complexity in terms of resolution, distortion, animation speed, and similar, and relating it to the ratio of humans capable of solving it. The complementary research is in the field of pattern recognition, trying to develop algorithms for breaking the CAPTCHAs and quantifying their performance as a function of CAPTCHA complexity. Combined, they should lead to a relationship between human-perceived transformation complexity and the security of the method.

Acknowledgment

The authors would like to thank Jerry Huxtable for his image filters and Sarmad Hussain for help with implementation.

References

[1] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, February 1978.

[2] TCG Best Practices Committee, "Design, implementation, and usage principles for TPM-based platforms," 2005. [Online]. Available: https://www.trustedcomputinggroup.org/downloads/bestpractices-/Best_Practices_Principles_Document_v1.0.pdf

[3] F. Chiachiarella, U. Fasting, T. Fey, S. Leppler, G. Lux, P. Lubb, A. Moser, G. Otten, J. Schlattmann, S. Schumann, L. Schweizer, and F.-J. Souren, "Das Risiko Trusted Computing für die deutsche Versicherungswirtschaft," *Schriftenreihe des Betriebswirtschaftlichen Institutes des GDV*, vol. 13, 2004. [Online]. Available: http://www.gdv-online.de/tcg/pos_tcg.pdf

[4] M. Naor and B. Pinkas, "Visual authentication and identification," in *CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 1997, pp. 322–336.

[5] M. Naor and A. Shamir, "Visual cryptography," *Lecture Notes in Computer Science*, vol. 950, pp. 1–12, 1995. [Online]. Available: citeseer.ist.psu.edu/naor95visual.html

[6] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, November 1979.

[7] I. Fischer and T. Herfet, "Visual document authentication using human-recognizable watermarks," in *Proceedings of ETRICS 2006, LNCS 3995*. Springer-Verlag, June 2006, pp. 509–521.

[8] D. Aucsmith, "Tamper resistant software: An implementation," in *Proceedings of the First International Workshop on Information Hiding*. London, UK: Springer-Verlag, 1996, pp. 317–333.

[9] L. von Ahn, M. Blum, N. Hopper, and J. Langford, "CAPTCHA: Using hard AI problems for security," in *Proceedings of Eurocrypt*, 2003, pp. 294–311. [Online]. Available: citeseer.ist.psu.edu/vonahn03captcha.html

[10] G. Mori and J. Malik, "Recognizing objects in adversarial clutter – breaking a visual captcha," in *Proc. Conf. Computer Vision and Pattern Recognition*, vol. 1. Madison, USA: IEEE Computer Society, June 2003, pp. 134–141. [Online]. Available: citeseer.ist.psu.edu/mori03recognizing.html