Visual Document Authentication Using Human-Recognizable Watermarks

Igor Fischer and Thorsten Herfet

Saarland University, 66041 Saarbrcken, Germany,
{fischer, herfet}@nt.uni-saarland.de,
WWW: http://www.nt.uni-saarland.de/

Abstract. Digital signatures and message authentication codes are well known methods for ensuring message integrity. However, they rely on computations which are too hard to be performed by humans and are instead done on computers. Trusting a digital signature implies trusting the computer which produced/checked it. Often, this trust cannot be taken for granted. This paper presents a method for visual authentication of large messages which relies on embedding a human-recognizable watermark and needs practically no computational power on the receiver side. Also, using a simple challenge-response mechanism is proposed to prevent attackers from obtaining signatures without author's knowledge.

1 Introduction

Paperless office has been a vision for decades and still has not come to exist. Although much – probably the vast majority – of communication today is done electronically, the most important parts are still performed in hardcopy. Most banks today offer online banking, and acquisition of a real estate can be negotiated per e-mail. However, to open a bank account, as well as to buy a property, one will have to print the documents, sign them by hand and return them by mail. The main, if not the only reason, is a legal one: for a contract to be binding, it must be provable and non-repudiable that the contracting parties have given their consent to it. This condition is actually two: (1) The contracting parties, and no-one else, have to give consent and (2) the consent has to be given to the contract, and not to anything else.

With hand-written signatures, both conditions are satisfied: a signature can be traced back to the originator (or at least the legal practice assumes so), and it may be safely assumed that the signer has read the contract before signing it. Digital signatures [1], as envisioned for the application, also fulfill the first condition: only the righteous owner should be able to apply them, because they are in his possession, e.g. on a smart card in his wallet. Furthermore, they are protected by some secret (like a PIN), so that even in the case of theft or loss, no-one else can use them. Fulfilling the second condition, however, is not that simple. Because of the computational complexity, the signing is done by a computer (e.g. the chip on the smart card). Since such devices normally do not have sufficient

displaying capabilities, the user cannot actually know what exactly is being signed. To better understand the danger, consider a typical scenario:

Alice[1] is writing a document (e.g. a contract with Bob) on her desktop computer, using some text-processing program. The computer is an ordinary, general-purpose personal computer. In addition, Alice can count on help from Trent, a trusted entity, which, for example, can be implemented as a smart card and a reader attached to the computer. Trent is the keeper of Alice's signature and signs documents Alice sends him in her name. Having written the document, Alice authenticates herself towards Trent (e.g. types in her PIN) and clicks the "Sign!" button on the computer. The computer sends the document to Trent, which signs it and returns it to Alice's computer. Alice sends the document with the accompanying signature to Bob.

Unfortunately, Mallory has installed a malicious program on Alice's computer, which changes documents to Mallory's favor without Alice's knowledge. When Alice sends the document to Trent, Mallory's program intercepts it, changes it, and passes it further to Trent. Trent, not being aware of Mallory, signs the altered document and returns it back, again through Mallory's program. Alice notices nothing. If she tries to take a look again at the document, Mallory's program would show her her original document. But when she tries to send it to Bob, the program sends the altered, signed version. Bob checks the signature and is confident that the document comes from Alice.

This is, obviously, something that must never happen and the very reason why digital signatures have been introduced in the first place. The reason why the scheme fails is, interestingly, not some weakness of the digital signature algorithm, although such weaknesses usually get much publicity. The reason for the failure is that current general purpose personal computers are inherently unsafe and cannot be trusted. As far as secure devices, such as smart cards, are concerned, they can, by definition, be trusted to perform their function correctly. But, since they lack a display, the user cannot know on which data the function is performed. The weak link is from the computer display to the trusted device.

There are basically two ways of solving the problem: making the hardware trustworthy, or empowering the user to check what is actually happening inside the computer. Attempts to pursue both approaches already exist, but have their limitations. An overview is given in the next section. In section 3, an approach belonging to the latter family, based on visual cryptography, is described. Differing from previous approaches, it uses visually recognizable watermarks to ensure the authenticity of the message. Also, using a simple challenge-response protocol is proposed to prevent the attacker from obtaining valid signatures without authors consent.

[1] The actors' names try to follow the convention from [2]

2 Related technology

2.1 External trusted device with input and output capabilities

One possibility to make the hardware trustworthy is to equip trusted hardware with input and output interfaces directly usable by humans, e.g. a display and a keyboard. This approach is already in wide use for electronic payment, especially with debit cards. The readers for such cards include a small LCD and a numerical keypad and are certified by some authority. In Germany, for example, the authority is the Central Credit Committee (Zentraler Kreditausschuss, ZKA), which lays down the criteria which card readers have to fulfill for a certain application. The highest security level is provided by Class 3 readers, with a built-in display and a keypad.

With such readers, the user can verify the document on the display before using the keypad to initiate the signing of the document. As long as the card reader and the card can be trusted — and this is the basic assumption behind the technology — this procedure is perfectly safe. In practice, however, due to physical limitations on the display, it is useful only for very short documents: the reader's display is usually only a dozen or two characters wide with only a couple of lines. This is sufficient for displaying the price or merchant identification, but not for checking a 20-page legal document.

2.2 Trusted Computing

An opposite approach is pursued by the Trusted Computing Group (TCG), a "not-for-profit organization formed to develop, define, and promote open standards for hardware-enabled trusted computing and security technologies, including hardware building blocks and software interfaces, across multiple platforms, peripherals, and devices" [3], with AMD, Hewlett-Packard, IBM, Intel Corporation, Microsoft and Sun Microsystems, Inc. as promoters and over a hundred participants. Instead of attaching an external secure device with its own in- and output, the idea is to convert the whole user's computer, including the keyboard and display, into a "trusted platform" (TP). At the same time, the platform should remain a general purpose computer, where anyone (or at least the owner) can install software or add peripheral devices. An additional requirement is that computers must remain affordable in order to be accepted by the market. This seemingly impossible combination of goals is achieved by only a minor modification to the hardware, in which a per-definition trusted device, the so-called Trusted Platform Module (TPM), plays a central role. The TPM is a kind of cryptographic microcontroller – a processor with volatile and non-volatile memory and simple I/O bus, – much of the kind employed in smart cards.

The platform relies on the TPM to trace the state and changes to its hardware and software, so no change can pass unnoticed. The trust in the platform is achieved through the chain of trust. At power-on, the first program that runs (in common personal computers typically from the BIOS) would compute a checksum or some other hash value of the next program(s) to be performed, and

compare the value with the one stored and signed by the TPM. The initial values are stored by the some trusted entity, e.g. the computer manufacturer. If the values match, the next program can be trusted and is executed. This program might perform some "measurements" of the hardware (checking the graphics adapter, the hard disk etc.) and use the same mechanism to check if the values are correct, i.e. that nothing has been changed. Again, if this test passes, the hardware can be trusted. This chain unfolds further, over the operating system loader and the operating system, up to application programs. In each step, a program checks its successor before executing it.

The security in this approach relies on the trustworthiness of the BIOS (the trust in TPM is per definition granted). As long as Mallory cannot manipulate the first program executed after power-on, she cannot plant a virus (or any other program) unnoticed. In practice this usually means that Mallory would have to have physical access to the computer. Even then, the computer might be physically protected, e.g. sealed, so that user could detect unauthorized opening. The Trusted Computing (TC) specification requires at least that the BIOS is physically marked so that removing it cannot pass unnoticed [4], at least for someone who bothers to take a look inside the computer. For even higher protection it is envisioned for the future to place the first program to be executed inside the TPM.

Leading manufacturers have announced TC products for 2006. However, the full technology will not be a part of Microsoft's new operating system [5, 6], codenamed Longhorn, although it was expected to be, and the API specification has not yet been made public. The customers also seem to be reluctant about accepting it, as the technology has faced serious criticism. For example, the German Association of Insurance Industry (GDV) is decidedly against TC, for three basic reasons: lack of legal framework, lack of control possibility and high misuse potential [7]. It is feared, among other things, that through TC manufacturers might coerce users into using or not using some soft- or hardware, and that private information might be indirectly disclosed without user's knowledge. The TCG has attempted to dispel the fears [8], but their success remains unknown. That the fears are not baseless is indirectly confirmed by the TCG best-practice manual [9], which denounces such misuses of the TC technology. The manual is, however, only a recommendation, and compliance with it cannot be enforced. As experience teaches us, if something bad can be done, it usually will be done by someone. A recent example is the Sony BMG "rootkit" DRM tool, which has made computers vulnerable to virus attacks without notifying the users [10].

Notice that TC in its current form protects the cautious owner against malicious manipulations on the computer. It does not protect remote users (e.g. his communication partners) from willful abuse by the computer's owner. If the owner alters the BIOS, remote users have no way of knowing it. Therefore, to achieve the level of trust needed for legally binding documents, TC platforms will have to be complemented by technology which is unconditionally trusted, such as external smart cards.

2.3 Visual Cryptography and Authentication

Both above approaches have drawbacks, including the need to extend the trust from the personal smart card to other devices. An appealing, low-tech alternative for short messages is visual authentication [11]. The idea was originally developed for authentication of electronic payments and is based on visual cryptography [12].

Visual cryptography is a perfectly secure cryptographic method based on a visual secret key. The encryption is computationally intensive and is done by the computer, but the decryption is performed with little conscious effort by the human visual system. The method is most easily implemented for encrypting black-and-white images, and works as follows:

The image to be encoded is scaled up by an integer factor, typically a multiple of two. Thus, for each original pixel in the image there will be a square of $N \times N$ pixels in the scaled image. For a black pixel in the original image, all pixels in the corresponding $N \times N$ square (so-called "subpixels") will be made black. For white original pixels, half of the subpixels will be made black and the other half white. Visually, looking from appropriate distance, such squares appear gray. Which subpixels will be made black and which white is randomly decided for each square. This way, the whole black-and-white original image is transformed into a bigger one. Originally black areas are simply enlarged, but originally white are also transformed into a uniform distribution of black and white pixels, which appear gray to a human observer.

The next step is to split the transformed image into two "shares", so that neither of them alone reveals any information about it. A "black" square, consisting of only black subpixels, is randomly split into two complementary squares, each with half black and half white subpixels. A "white" square is split into two identical squares. It is useful to consider white subpixels to be transparent, as if printed on a transparency. Then, by superimposing the shares (laying the transparencies over each other), "black" and "white" squares, comprising the scaled image, reappear. If we denote black subpixels in the shares with 1 and white with 0, superimposing the shares corresponds to binary OR.

This is a visual implementation of the 2-out-of-2 secret sharing technique [13]. The basic idea of k-out-of-n secret sharing is to split a message into n "shares", so that none of them alone, nor any combination of less than k of them, reveal anything about the message, but k shares combined are sufficient to reconstruct the whole message. If $n = k = 2$, this is actually a one-time pad cryptography. One of the shares (transparencies in our case) is used as the cyphertext and the other as the key. But each for itself looks like a uniform distribution of black and white pixels.

Visual cryptography can be used for authentication, although not directly. Basically, the idea is for the user (Alice) and the trusted device (Trent) to share a secret key — Alice would have it in the form of a pre-printed transparency. Trent would send not only the signed document back to Alice, but also its visually encrypted version. She would visually check if it is identical to the document she sent him and only if yes, use the signed document. This simple approach,

however, would not work, because Mallory knows both the plaintext and the cyphertext and, consequently, can deduce the secret key. Knowing the key, she can produce any document and properly encrypt it, and Alice would believe it comes from Trent. Therefore, Trent must expand the document with information known to Alice, but not to Mallory, before encrypting it. Several related methods have been proposed in [11]:

1. Content/Black Areas: The transparency contains two areas, denoted "black" and "content", and Mallory does not know which is which. Trent constructs the cyphertext so that the document appears in the "content" area, and the "black" area is completely black. This requires the transparency to be a one-time pad, otherwise the "black" area of the cyphertext would not change from message to message and Mallory could simply identify it
2. Position on the Screen (or, generally, output device): The transparency has a marked area in which the document has to appear. Mallory does not know where this area is positioned.
3. Black and Gray: instead of "white", "gray" is used. It is encoded by having three quarters of the subpixels in a square black and one quarter white. This increases the security even when the plaintext is known, because for a fixed share of a gray square there are many ways (four in case of $N = 2$) of constructing the other share. So, for every "gray" pixel in the original image, Mallory has only a low probability of turning it into black. It does not hold for the opposite direction, however, so Trent is required to send the document (black on gray) and its inverted version (gray on black) to Alice for checking. The other drawback is that this approach reduces the contrast, making the result difficult to visually recognize.

3 Using watermarks for obfuscating the plaintext

The above visual authentication approaches require transparencies bigger than the document and possibly reduce its readability. These are not grave issues for the originally envisioned electronic payment application, where the documents would be short (like the price to pay) and displayed on a high-contrast screen. However, for documents consisting of several pages they might be impractical.

An alternative approach is presented here. Recall that the reason for not using visual cryptography directly was Mallory's knowledge of the plaintext, which allowed her to deduce the secret and, consequently, to arbitrarily modify the cyphertext, without Alice noticing it. To counter this danger, Trent can modify the document in a way known to Alice, but unknown to Mallory. On the other hand, the modified document should still allow Alice to check that the essential content arrived to Trent unaltered. Both can be achieved by incorporating a faint image, a kind of "watermark", into the document.

Watermarks have been used for centuries in paper production, presumably from the beginning for security-related purposes. They are images, visible under special circumstances, embedded into the paper. Today they are most often used on paper money as a protection against forgery. There is also an analogy for

digital data, so-called digital watermarks, which are hard-to-detect and hard-to-remove pieces of information hidden among the original data. For the purpose of this text, watermarks are understood as digital, but visible, human-recognizable images. The proposed method, which is suitable for longer documents, is basically as follows:

Alice (human) and Trent (trusted device) share a secret (a physical transparency in Alice's case and its digital representation in Trent's) and, in addition, a list of images, which function as visual challenges, and simple alphanumeric responses. Such a list is just a more sophisticated variant of a Transaction Authorization Number (TAN) list, often used in Europe for online banking. It can be produced and distributed in a similar way as the TAN lists, by the entity which manufactures or distributes Trent. Alice could get her list by mail and Trent online. The distributing entity can use a mechanism similar to Alice's to authenticate Trent before sending him the encrypted list.

Alice composes the document on her ordinary, not-to-be-trusted personal computer and sends it to Trent — an external hardware device (a smart card or a USB stick), a trusted remote computer, or a tamper resistant program [14] running on the user's PC (in the latter case no additional hardware is needed). The document can be in any form Trent understands (a PDF document, LaTeXsource code etc.), but it is practical to think of it as a black-and-white image, or a sequence of images in case of multiple pages.

Trent, having received Alice's i-th document (page), superimposes it (performs logical OR) with the i-th image (watermark) from the list, splits the result into two shares, one of the shares being the secret held by Alice, and sends her the other share. Alice lays her transparency over the share (she can do it directly on her computer display or print the share) and checks if

1. the document she wrote is embeded into Trent's image unaltered, and
2. the image superimposed to it is the i-th watermark from her list

If both conditions are satisfied, Alice sends Trent the corresponding alphanumeric response. Trent checks if the response corresponds to the i-th image in the list (the one he superimposed to the document) and if yes, signs the document and returns the signed file to Alice. The workflow is depicted in Figure 1. An example for the transparency, Trent's challenge, and the superposition result are shown in the Figure 2.

The purpose of superimposing the document with a watermark is to make any tampering with the document obvious to Alice. However, it does not prevent Mallory from misleading Trent into signing a modified document and using it for his own benefit. Mallory could impersonate Alice towards Trent and make him sign any document. Since she could be in control of Alice's computer, she could send the document in Alice's name, who would not be able to prevent her, even if she would notice the fraud. Only through the challenge–response mechanism can Trent be sure that it is not Mallory asking him to sign the document and can confidently send the signed document back to Alice.

3.1 Attacks

Knowing the document, Mallory can still deduce for each transparency how the squares corresponding to black pixels in the document (but not in the whole challenge!) are encoded (split into shares). But, this would not be enough to allow her to meaningfully modify it. She would need to be able to turn black document pixels into white, and white into black, without damaging the watermark.

Consider the task of turning a black document pixel $d = 1$ into white. Through the watermarking the pixel is ORed with the corresponding watermark pixel w, which is unknown to Mallory. She knows the superposition result, $d \vee w = 1$ and how it is coded in shares, but, not knowing w, cannot deduce $\overline{d} \vee w$. She can force the corresponding square to "white", by inverting its share, or throw a coin and decide whether to leave it as it is or invert it. In both cases she is guessing the value of w and his chances depend on the distribution of white and black in the watermark. For watermarks with an equal number of black and white pixels Mallory's probability of guessing one pixel are 1/2. In a typical document, where a character is composed of dozens of pixels, Mallory's chances of tampering with the document without distorting the watermark are negligible.

In turning a white document pixel into black, Mallory has similar problems. She knows $d = 0$, but not whether $d \vee w$ is 0 or 1. If she decides to invert the corresponding share, she again runs into the risk of distorting the watermark. What she can do is to force an illegal share, by making all subpixels in the square black, but such tampering with the cyphertext is easily spotted before overlaying it with the secret key.

Results of example attacks, where Mallory has tried to change the sum 10,000 into 100,000, are shown in Figures 3 and 4.

3.2 Security considerations for repeated use

The basic scheme above has a potential weakness. Since Trent uses logical OR to combine the watermark with the document, Mallory knows that any black square in the document will remain black in Trent's challenge image. This might not be enough to manipulate the document, because she still lacks the information corresponding to "white areas" in the document, but allows her to deduce a part of the secret key (the transparency). After a repeated use of the same transparency, the danger is that Mallory could collect enough data to reconstruct it in full.

There are several possibilities to counter this danger:

1. Instead of repeatedly using single transparency, let Alice and Trent share a whole codebook — a staple of transparencies in Alice's case. Each would be used only once, as a one-time pad.
2. Trent might use XOR instead of OR for combining the watermark and the document. This way Mallory, although knowing every document pixel d, does not know if $d \oplus w$ is 0 or 1 (white or black) for any pixel in the challenge

image and, consequently, cannot meaningfully modify the document. The drawback is that the document becomes visually harder to recognize.

3. Trent introduces some simple transformation to the document before watermarking it, like inversion, slanting, rotation, or translation. The transformation can be only slight, like rotation for several degrees and translation for a couple of pixels. The document remains visually recognizable to Alice, but becomes unknown to Mallory.

The last two approaches alone are still susceptible to statistical attacks. High-resolution black-and-white images have the property that for most pixels their neighbors have the same value. Pixels on edges are exceptions, but they appear much less frequently in images which are easy to visually recognize. If the same secret key is used over and over again, Mallory could use this property to gain knowledge about it. It is therefore advisable to combine them with the first approach. For practical purposes, however, it is not necessary to use each transparency only once. Depending on the complexity of the watermarks and Mallory's assumed pattern recognition capabilities, Alice could use a transparency several times before Mallory collects enough data for an attack.

4 Conclusion

The computational complexity of "classical", non-visual cryptographic techniques implies the need for cryptographic devices. Classical cryptography offers a high level of protection for digital documents and is essential in ensuring an efficient and secure electronic communication. However, a big challenge has been securing the path from the human to the cryptographic module. This path is currently the weakest link, which limits the security of the whole cryptographic chain.

In this paper, an approach for bidirectional document authentication based on visual cryptography and watermarking was presented. It requires no additional computer hardware and is very easy to implement using the existing infrastructure. Compared to previous such approaches, this method uses the available area of the visual shared secret area (the transparency) more efficiently, which makes it much more suitable for authentication of larger documents, even consisting of a number of pages. The proposed challenge-response mechanism prevents the man-in-the-middle attacker from obtaining a signed document without author's approval. Assuming that the author would not approve a forged document, the attacker is prevented from obtaining a valid signature on a forged document.

On the user's side, the method requires a list of watermarks and a staple of transparencies. For each document page, a watermark and a transparency are needed. The watermarks, which appear only faint over the document, can in the list be printed reduced in size, so that a dozen or two fit on a sheet of paper. The transparencies, however, have to be full-sized and would probably be distributed in a form of a booklet.

The method is not inteded for to be used among arbitrary number of users and trusted devices. It essentially relies on symmetric cryptography, so the number of key sets (staples of transparencies and watermarks) increases linearly with the number of user per trusted device. However, for the envisioned application — securing the channel between a user and her trusted device — there should be only one key set per user.

References

1. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21** (1978) 120–126
2. Schneier, B.: Applied Cryptography. John Wiley & Sons, Inc. (1996)
3. Trusted Computing Group: Home page (2005)
4. Pearson, S., ed.: Trusted Computing Platforms. Prentice Hall PTR, Upper Saddle River, New Jersey (2003)
5. Fried, I.: Microsoft: 'Trusted Windows' still coming, trust us (2005)
6. Slater, D.: Microsoft trusted computing updates (2005)
7. Chiachiarella, F., Fasting, U., Fey, T., Leppler, S., Lux, G., Lubb, P., Moser, A., Otten, G., Schlattmann, J., Schumann, S., Schweizer, L., Souren, F.J.: Das Risiko Trusted Computing für die deutsche Versicherungswirtschaft. Schriftenreihe des Betriebswirtschaftlichen Institutes des GDV **13** (2004)
8. Trusted Computing Group: Trusted Computing Group Clarifications for the German Insurance Industry Association paper "The Threat, Trusted Computing, to the German Insurance Industry" (2005)
9. TCG Best Practices Committee: Design, implementation, and usage principles for TPM-based platforms (2005)
10. Russinovich, M.: Sony, rootkits and digital rights management gone too far (2005)
11. Naor, M., Pinkas, B.: Visual authentication and identification. In: CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, London, UK, Springer-Verlag (1997) 322–336
12. Naor, M., Shamir, A.: Visual cryptography. Lecture Notes in Computer Science **950** (1995) 1–12
13. Shamir, A.: How to share a secret. Communications of the ACM **22** (1979) 612–613
14. Aucsmith, D.: Tamper resistant software: An implementation. In: Proceedings of the First International Workshop on Information Hiding, London, UK, Springer-Verlag (1996) 317–333

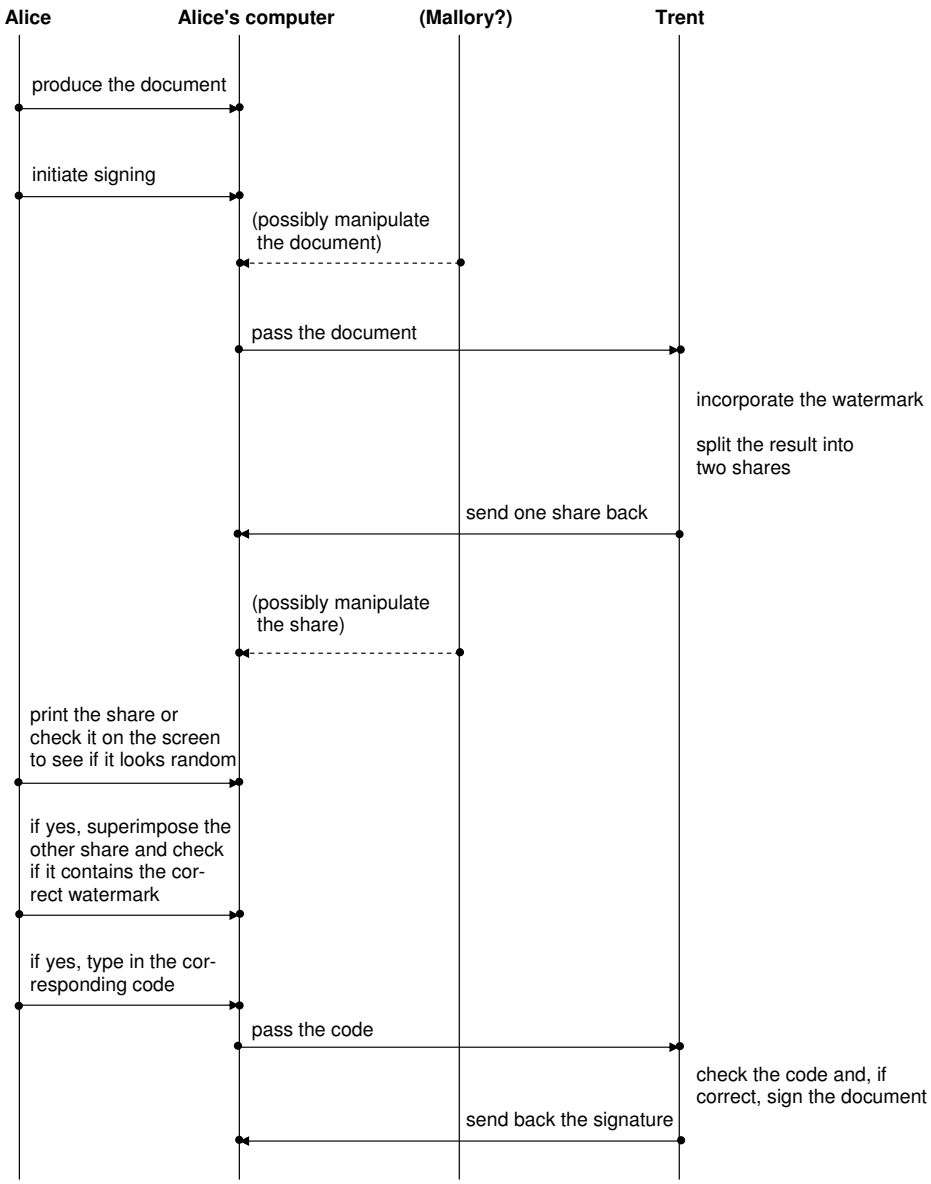

Fig. 1. Document authentication protocol with possible attack poinst for Mallory

Fig. 2. An example of visual authentication through watermarking: the secret key (transparency) (top), the Trent's share (middle), and the watermarked and encoded document (bottom).

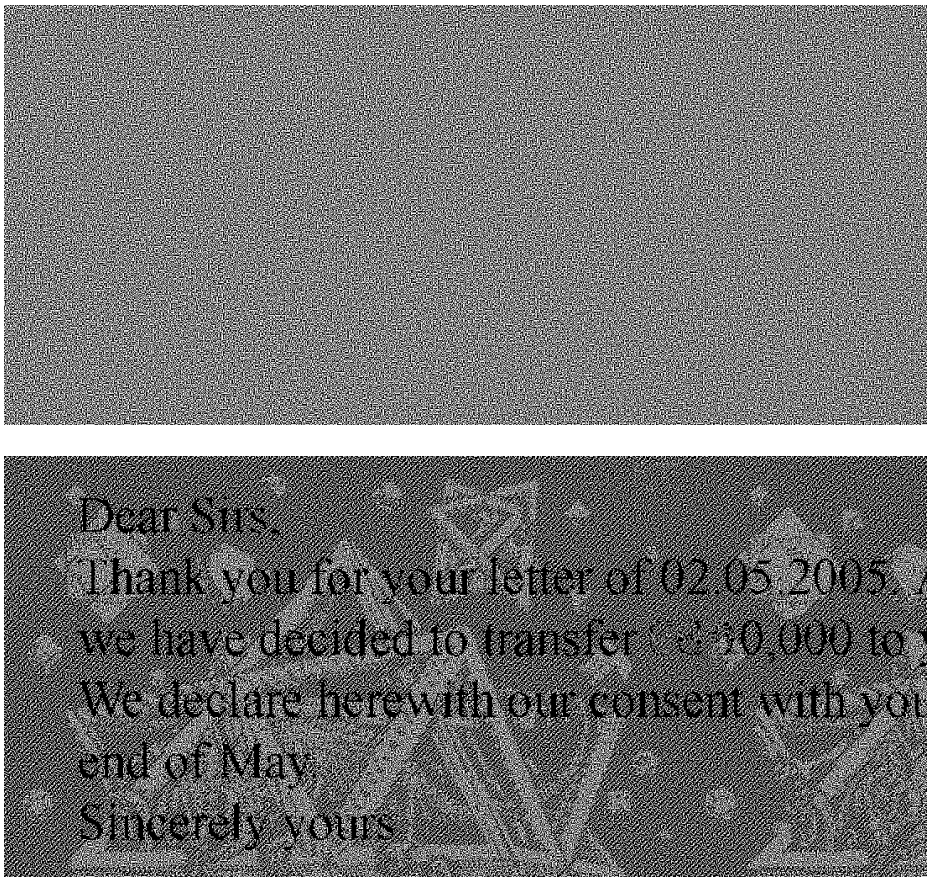

Fig. 3. An example attack with randomly inverting shares of pixels that need to be manipulated. The forged document share (top) is indistinguishable from the original one, but the superposition result clearly reveals tampering.

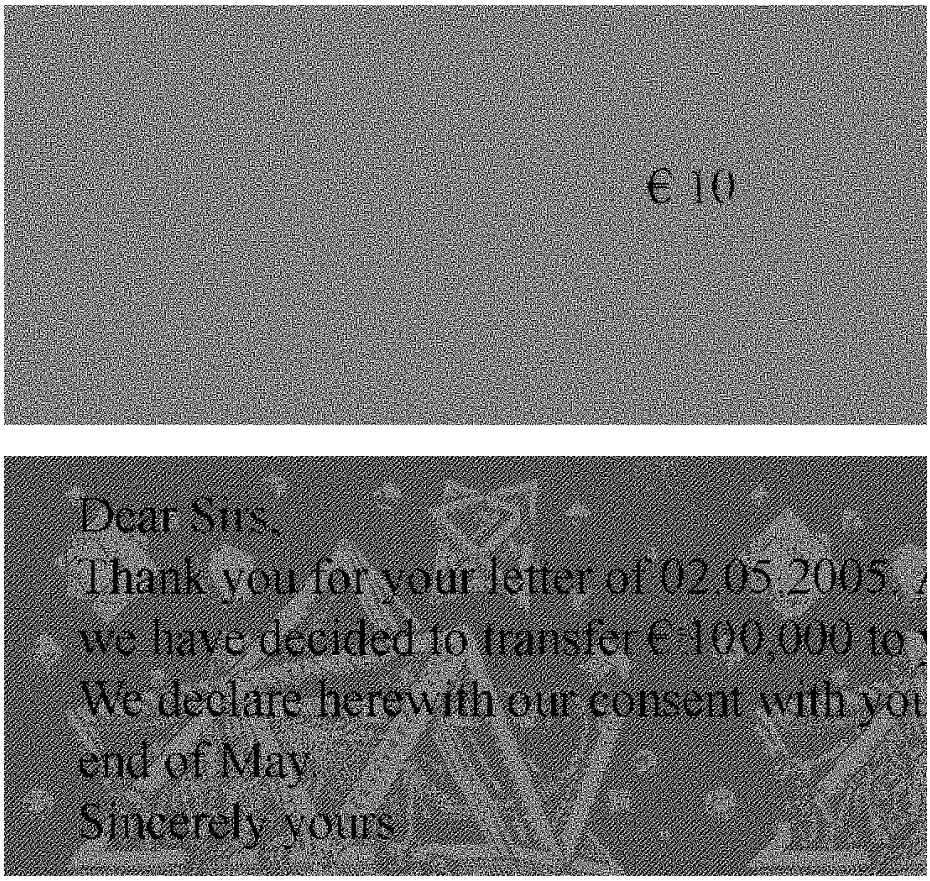

Fig. 4. An example attack, where the attacker, for the pixels he wants black, forces the complete pixel shares to be black. The result is better than in the previous example, but the forged document share (top) is clearly illegal.