When Document Security Brings New Challenges to Document Analysis

Sébastien Eskenazi, Petra Gomez-Krämer, Jean-Marc Ogier

When document security brings new challenges to document analysis

Sébastien Eskenazi, Petra Gomez-Krämer, and Jean-Marc Ogier

Laboratoire L3i, Université de La Rochelle
{sebastien.eskenazi,petra.gomez,jean-marc.ogier}@univ-lr.fr
l3i.univ-larochelle.fr

Abstract. It is very easy to ensure the authenticity of a digital document or of a paper document. However this security is seriously weakened when this document crosses the border between the material and the digital world. This paper presents the beginning of our work towards the creation of a document signature that would solve this security issue. Our primary finding is that current state of the art document analysis algorithms need to be re-evaluated under the criterion of robustness as we have done for OCR processing.

Keywords: Hybrid security, Text hashing, OCR hashing, Document analysis robustness

1 Introduction

With the ever increasing digitization of our world, people and companies now have to ensure the authenticity of many documents. For instance, an easy way to get a fraudulent identity card is not to forge one but to obtain a real one with fraudulent documents such as a fake electricity bill and a fake birth certificate [1]. Some of these documents are in a paper format and some are in a digital format. Ensuring the security of these two kinds of documents and of documents that can change format is called hybrid security. So far there is no other choice but to use two authentication systems: one for the paper documents and one for the digital documents. Even though watermarks or fingerprints can bridge the gap between the paper and the digital world they do not ensure the authenticity of the content of the document. They only ensure the authenticity of its support i.e. the file or the paper material [2].

The digital world has a very efficient tool to ensure the authenticity of a file: a hash key [3]. It cannot be used as-is on a paper document because of the noise of the scanning process which would always produce different files. We have the idea of extracting the content of the document and to hash this content instead of the raw scan of the document. This raises the immediate question of the content we want to extract. Is it just the text or does it also include the images, the handwritten signatures, or even a combination of those? And with which precision do we want to extract it? Is it plain text or do we also want to have the font, the font size and the emphasis? While investigating

this question, our first experiments prove that most document analysis tasks need to be thought again under a completely new paradigm in this community: robustness. It appears that current content extraction algorithms produce very different results with only a slight amount of noise. This makes it impossible to produce a reproducible signature. We started studying this robustness with the base algorithm of our signature: OCR. We plan to study the other necessary algorithms after we obtained a suitable OCR.

This paper is organized as follows. In Section 2, we introduce the general problem of hybrid signature and the performance objectives. Section 3 presents the workflow we intend to build to solve it and the issue of robustness. Then, we present our analysis for the task of OCR analysis in Section 4. Finally, Section 5 concludes our work.

2 Hybrid Signature

Before presenting the hybrid signature itself we will present the hash algorithms on which it is based.

2.1 Digital Hash Algorithms

A digital hash algorithm computes a digest for a message. A very good overview of digital hash algorithms can be found in [3].

The first kind of hash algorithm was cryptographic hashing. It was made in order to be able to control the integrity of the message content without having to read the entirety of the message. This was useful at the beginning of the Internet because of network transmission errors. Cryptographic hash algorithms are now mostly used for security applications and content retrieval. They have several features. The first one is that any small change in the message will change the digest with a very high probability. This is reflected in the collision probability, which is the probability of two different messages having the same digest. Another important feature of hash algorithms is the inability to recover the original message from its digest (hence the name "cryptographic"). The main consequence of this requirement is that any smallest change will completely change the digest. This allows the authentication of a confidential message without having to compromise the confidentiality of the message. The current standard cryptographic hash algorithm is SHA-256 as defined in the Federal Information Processing Standard FIPS PUB 180-4 [4] and FIPS PUB 186-4 [5]. A security analysis of SHA-256 can be found in [6].

The next kind of digital algorithm is fuzzy hash algorithms. Contrarily to cryptographic hashes, a small change in the message will only change a portion of the digest. This allows the retrieval of different messages that have similar message parts, but that are not completely identical (hence the name "fuzzy"). For this reason, the content of the message is not as protected as in a cryptographic hash algorithm. The most common fuzzy hash algorithm is ssdeep [7]. Fuzzy hash algorithms are also called perceptual hash algorithms especially in

the image processing community as the fuzzy hash is related to the content of the message and to the way a human perceives it. They are widely used for message retrieval, where the message can be of any kind such as an image [8], a text [9] or even raw bits of data on a hard drive [10]. A security analysis of perceptual hash algorithms can be found in [11].

Figure 1 shows the differences between these different hash algorithms and the proposed one.

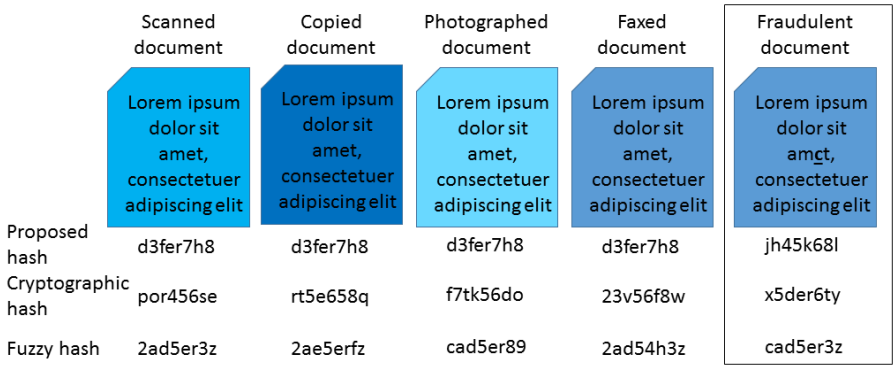

Fig. 1. Comparison of proposed, cryptographic and fuzzy digests on several copies of the same document and a fraudulent version of it. The cryptographic digests have nothing in common while the fuzzy ones all contain small variations making it impossible to identify which document is fraudulent.

2.2 Hybrid Security Algorithms

Villán et al. [12] elaborated two hashing algorithms. The first one is based on the combination of an OCR software (Abbyy Finereader) and a cryptographic hash (SHA-180). The second one is based on a random tiling hash, which computes the average luminance value on 1024 random rectangles for each word. They used an Arial font with a font size of 10 and no emphasis. The text was simple English text. While the first algorithm performs rather well with only two errors out of 64, the random tiling algorithm cannot differentiate one character from a similar one such as "o" and "e".

Tan et al. [13] developed a perceptual text hasing algorithm, which is very interesting as it is based on the skeleton features of each character. It has the drawback of removing all punctuation from the text.

The most recent work on hybrid security was made in two projects: the ANR Estampille [14] [2] and the European project SIGNED [15].

The Estampille project aimed at developing a 2D-barcode that could be used as a fingerprint. The barcode is printed with extreme precision in a specific printing process. This fingerprint is supposed to be impossible to be reproduced

or to be copied without detection. The latest results actually prove that having a dozen copies of an authentic fingerprint is enough to forge one. This does not include the fact that the document content is not included in the fingerprint. Thus once a forged fingerprint is made, it is possible to render any document authentic-like.

The SIGNED project was more ambitious and corresponds to what we can call a "hybrid signature". The goal of the project was to produce a digest of a document that allows the detection of any modification. It is based on a fuzzy hash algorithm, which has the disadvantage of breaching the possible confidentiality of the document, but this also allows for the localization of the modification. The document is analyzed at the signal level. They cut the document in tiles and use a Discrete Haar Wavelet Transform [16] on each of these tiles. For a document scanned at 600dpi, the tiles have a size of 64 by 64 pixels. Then a cryptographic hash is applied on each tile and all the digests are concatenated to create a fuzzy digest. The fuzzy digest is then printed on the document. During the verification process the digest of the scan is computed and compared with the one that is printed on it. A distance is computed between these two digests and if it is too large the document is considered to be fraudulent.

The results of the SIGNED project had to meet six performance indicators:

1. Probability of missed detection and false alarm
   (a) Probability of false alarm (PFA) below 0.001
   (b) Probability of missed detection (PMD) below 0.001 with $PFA < 0.001$
      - for the replacement of digits in Arial 8, 10 and 12
      - for the replacement of dots by commas in Arial 10 and 12
2. Collision probability below 0.001
3. Minimum area size to detect a manipulation: 42×42 pixels at 600dpi
4. Throughput below 5 seconds per page
5. Size of the document digest below 4 kB
6. Compatibility with current scanners and printers

A false alarm occurs when a document is detected as fraudulent while it is not and a missed detection is the contrary. The minimum area size means that a manipulation has to be bigger than a square of 42 by 42 pixels to be detected.

The project met all these requirements except for:

- The PMD for the replacement of dots by commas that required a bigger font than Arial such as Verdana
- The minimum area size that is of 64×64 pixels at 600dpi
- The throughput that was not achieved for the verification phase (no figure was given)
- The size of the document digest that was between 4.8 and 170 kB depending on the required precision

The performance indicators were given by the industrial partners of the project and as such represent a reasonable goal to reach. We can also notice that if we want to use OCR processing the minimum area size performance indicator is not relevant anymore.

3 Proposed Solution

Since a hash algorithm cannot be applied directly on a document image because of the variability and the acquisition noise, our idea is to extract the content from the document image, to format this content properly and to apply the hash algorithm on it. This will drastically reduce the amount of data to hash and will allow to hash only the significant (semantic) content of the image. The interest of extracting the content is that this is exactly what is not supposed to – and what we do not want to – change. This will eliminate the noise and so the variability of the data, and will allow the use of a cryptographic hash. Thus, we can expect this technique to be applicable also in the case of a camera captured document.

As the notion of "semantic" can have several meanings we will attempt to define it now. Usually a hash algorithm is applied directly on the raw data, the bits of information. We intend to apply it on the content extracted from the raw data. Hence we consider a semantic hash on the contrary to a normal hash. We do not take into account the level of abstraction of the content on which we apply the hash. For instance, applying a hash algorithm on a sequence of characters without any understanding of their division in words still makes a semantic hash. What matters is that the data on which is the hash algorithm applied to is the one that contains only the information that is meaningful for a human (and in our case that is not supposed to change).

3.1 Signature Workflow

Figure 2 shows the basic workflow for the signature computation. There are three main phases: the preliminary phase (orange) extracts the different elements such as text, images, tables and diagrams from the document image. The second phase (gray) extracts the content of each of these elements. The last phase (green) reconstructs the document and computes its digest.

The first step of the preliminary phase removes any geometric distortions such as skew and warp. We then proceed with the segmentation phase and the tagging of each segmented element according to its content (text, image, table, etc.).

During the second phase, the elements are analyzed by a content extraction algorithm. The algorithm is specific for each element tag such as an OCR for the text.

Finally all the extracted contents are used to rebuild the document according to the segmented and analyzed data. We apply SHA-256 to compute the document hash.

3.2 The Issue of Robustness

The objectives require a false positive rate below one in a thousand and the same for a missed detection.

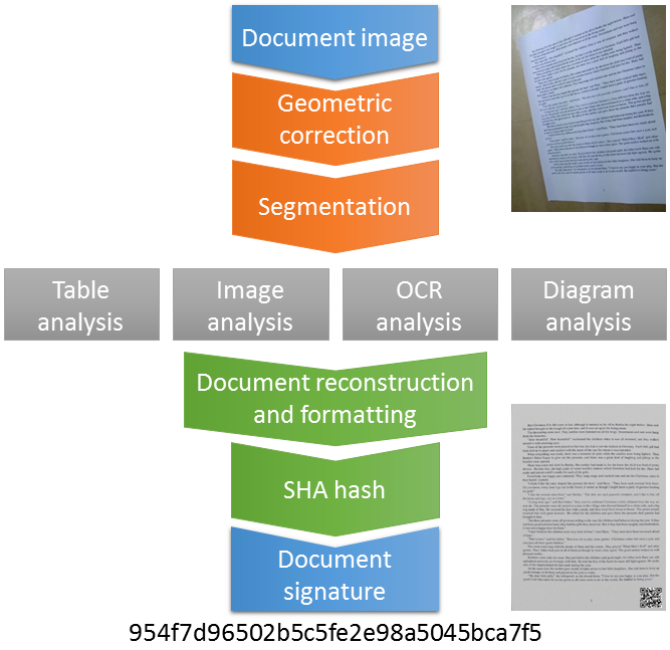

Fig. 2. The workflow of the signature computation.

The case of missed detection is not very important, except for some specific cases. For instance, in the case of undersegmentation, missed detections are mainly related to the precision of each content extraction algorithm. Should one algorithm not be good enough, we can either try to improve it, not use it and consider that the corresponding information is not secured or consider that the corresponding information is secured with a degraded level of security. Another solution is to only consider certain types of document images for which we have the desired accuracy. The precision of most document analysis algorithms is the most frequent criteria used to evaluate them. So the precision is usually quite good.

The case of false positives is very different. To prevent a false positive, we need to ensure that an algorithm will produce the same results for reasonably similar images of the same document and/or with some additional noise. This is the robustness of the algorithm and it is usually not analyzed. The issue with robustness goes even further than that: with a missed detection we accept a document that has a necessarily small modification (otherwise we would not miss it), while with a false positive we reject an entire document that is completely fine. From a practical point of view, if the rate of false positives is too high, there will be too many rejected documents. This means that either the document verification process is useless or it will require a lot of manpower to hand-verify each rejected document. False positives can be created at every step of the

signature process: the segmentation, the content extraction and the document reconstruction can all produce false positives that will have a direct impact on the signature.

This shows how critical it is to ensure the robustness of the algorithms used to compute the document signature. Unfortunately this robustness has hardly been studied. We did this work for OCR and present it in the next section.

4 Study of OCR Capabilities

We will first describe in detail our test dataset, our test protocol and our results of the OCR analysis. Finally we will discuss possible OCR improvements.

4.1 Dataset

Considering the quite stringent requirement for the OCR accuracy, we chose to use clean, text-only, typewritten documents. Tesseract can analyze documents with a single or double column layout, so we tested it with a combination of these. We used only and all the characters that it can recognize.

The dataset is made of 22 pages of text with the following characteristics:

- 1 page of a scientific article with a single column header and a double column body
- 3 pages of scientific articles with a double column layout
- 2 pages of programming code with a single column layout
- 4 pages of a novel with a single column layout
- 2 pages of legal texts with a single column layout
- 4 pages of invoices with a single column layout
- 4 pages of payslips with a single column layout
- 2 pages of birth extract with a single column layout

We created several variants of these 22 text pages by combining:

- 6 fonts : Arial, Calibri, Courier, Times New Roman, Trebuchet and Verdana
- 3 font sizes : 8, 10 and 12 points
- 4 emphases : normal, bold, italic and the combination of bold and italic

This makes 1584 documents. We printed these documents with three printers (a Konica Minolta Bizhub 223, a Sharp MX M904 and a Sharp MX M850) and scanned them with three scanners and at different resolutions between 150dpi and 600dpi as shown in Table 1. This makes a dataset of 28512 document images. Figure 3 shows an example of the images contained in the dataset.

4.2 Test Protocol

Most of the security algorithms are publicly available. For this reason we chose to use Tesseract [17] which is open-source. We used Tesseract version 3.02 with the default English training. We ran it on every image of the data set and used the ISRI evaluation tool [18] to compute the accuracy of the OCR and to analyze its errors. We also computed a hash digest of each OCR output file with SHA-256. We used these digests to compute the probability of false positives.

Scanner	150dpi	300dpi	600dpi
Konica Minolta Bizhub 223		X	X
Fujitsu fi-6800	X	X	
Konica Minolta Bizhub C364e		X	X

Table 1. Scanning resolution for each scanner

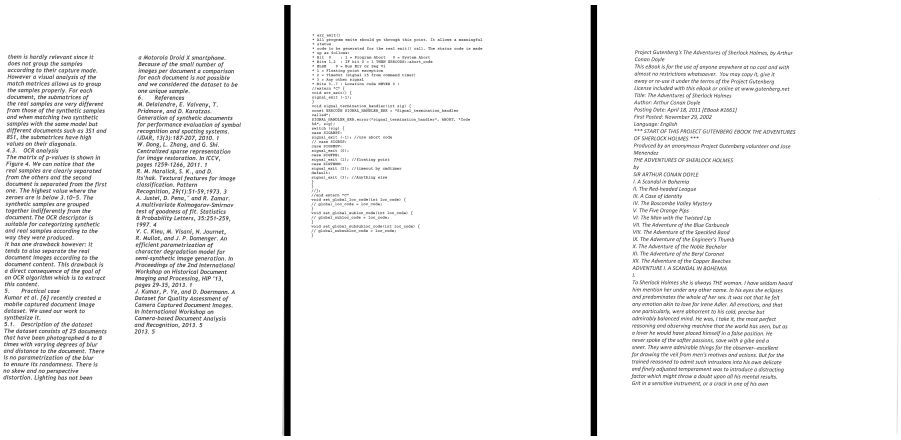

Fig. 3. An example of three document images of the dataset

Computation of PFA We will explain in the following the computation method for the probability of false positives. We consider a set of n documents. What a document exactly is will be defined for each computation. For each document there are m_i different digests, $i \in [1, n]$. Each digest is present s_{ij} times, $j \in [1, m_i]$. This creates a pseudo matrix S_{ij} containing all the s_{ij} values. For any i, m_i is the number of non null values on the i^{th} row and the number of images is the sum of all the values contained in the matrix.

The signature verification process can be modeled by the successive draw without replacement of two digests among all the digests available for the document. The draw is without replacement as we consider it impossible to produce twice exactly the same image. A false positive happens when the two digests that are drawn are not the same. Hence the probability of a false positive for the i^{th} document is given by:

$$P_i = 1 - \sum_{j=1}^{m_i} \left(\frac{s_{ij}}{\sum_{k=1}^{m_i} s_{ij}} \times \frac{s_{ij} - 1}{\sum_{k=1}^{m_i} s_{ij} - 1} \right) . \quad (1)$$

The global false positive probability is the mean of the probabilities for each document.

Post Processing After some preliminary tests we noticed that it is unreasonable to require the OCR to distinguish some characters such as a capital "i" and

a lower case "l" or a capital "o" and a zero. Tesseract regularily mistakes on for the other and it would be the same for a human. For this reason we added an alphabet reduction as post processing step. Table 2 summarizes the alphabet reduction.

Character	Replacement
Empty line	Removed
Tabulation and space	Removed
— (long hyphen)	- (short hyphen)
', ‘ (left and right apostrophes)	' (centered apostrophe)
", “, ” (left and right quotes, double apostrophe)	" (centered quote)
I, l, 1 (capital i, 12^{th} letter of the alphabet, number 1)	|(vertical bar)
O (capital o)	0 (zero)
fi (ligature)	fi (two letters f and i)
fl (ligature)	fl (two letters f and l)

Table 2. Alphabet reduction

4.3 Results

We evaluate the results of our proposed method in terms of precision, probability of false positives and collision probability, which are detailed in the following. Table 3 sums up most of our results.

Precision The OCR precision ranges from 14.5% to 100% with an average of 98.08%. The worst results are obtained for the pages of code as tesseract does not handle well the code text syntax. The other low precision results are due to segmentation errors.

The precision increases seriously when using a resolution of 300dpi rather than 150dpi as it increases from 91.58% to 99.33%. The improvement is much less significant when using 600dpi instead of 300dpi with a precision reaching 99.45%.

The font size and emphasis also play an important role. The average precision is 95.83%, 99.02% and 99.38% respectively for a font size of 8, 10 and 12 points. The use of italic worsens the accuracy results as, by instance, the vertical bars "|" can be misrecognized as division bars "/".

The choice of the font has a similar effect. Courier and Times New Roman get the worst results and the best ones are obtained for Verdana. This is explained by the fact that Courier characters have a very special shape, Times New Roman is the smallest font and Verdana the biggest.

Even though the accuracy results are not perfect they allow for us to meet the probability of missed detection criteria.

False Positives The probability of false positives is quite different. As the OCR extracts only the text without taking into account the font, font size or emphasis we can consider that our dataset as 1296 copies of 22 document pages. A document is then only defined by its textual content. This produces a probability of false positives of 92%!!

We then decided to consider a document as being defined by its textual content, the font, font size, font emphasis and scanning resolution. Our dataset is then composed of between 3 and 9 copies of 4752 documents. This produces a probability of false positives of 84%.

This is where increasing the scanning resolution has a real interest as the probability of false positives decreases from 99% to 79% and to 73% for a resolution of 150, 300 and 600dpi respectively. The other influencing factors are the same as for the probability. This is understandable as the less errors there are, the less variation possibilities there are and the lower the false positive probability. However one could imagine that the same errors could be done every time thus reducing the false positive probability. This is not the case.

The best case scenario is when we consider a scanning resolution of 600dpi, a font size of 10 or 12 points, no italic emphasis and any font but Courier and Times New Roman. This allows us to reach a probability of false positives of 57% and even 53%, if we do not take into account the pages of code. The OCR precision reaches 99.79% and 99.83% in these two cases respectively.

We can see that the issue of robustness has not been resolved at all. If we consider a text of 2000 characters and we want to have less than one different OCR text out of one thousand copies of the text; this means an error rate of one in two million.

Collision Probability The collision probability is the probability of having two different documents producing the same digest. It can be due to three factors: the OCR, the alphabet reduction and the hashing algorithm.

The OCR introduces a collision probability when it recognize two different characters as the same. This probability can be considered equal to its error rate which is about 0.002.

The alphabet reduction introduces a collision when two different characters are replaced by the same character. However the alphabet reduction was made so that only characters that a human could mistake for one another are replaced. This means that it does not actually create collisions except for reference numbers combining letters and numbers and for the replacement of capital "i" by an "l". We applied the alphabet reduction algorithm on the Aspell English dictionnary in order to see how many similar words it would produce. As expected the only collisions were due to the replacement of a capital "i" by an "l". If we take the font case into account we obtain a collision probability of 0.0002.

Finally SHA-256 has a negligible collision probability usually below tens of orders of magnitude below the OCR and alphabet reduction collision probabilities. No collision has been found so far for SHA-256 [6].

Hence, the most critical part is the OCR precision which brings the collision probability up to 0.002. It should not require much work to bring it below 0.001.

Criteria	General case				Best case scenario
	150dpi	300dpi	600dpi	All resolutions	
Precision	91.58%	99.33%	99.45%	98.08%	99.83%
Probability of false positive	99%	79%	73%	84%	53%
Collision probability	0.0084	0.007	0.004	0.019	0.002

Table 3. Average values of the results of the OCR testing

Other Performance Criteria All the other performance criteria are met. The probabilities of missed detections are equal to the OCR error rate, which is below 0.1% in the case of digits, commas, and dots. The minimum area size is not relevant for an OCR algorithm. The throughput is about 2 seconds on average which is well below 5 seconds. SHA-256 produces a digest whose size is 256 bits, which is well below the 4kB limit. Finally, our system was tested with three printers and three scanners which proves its compatibility with current scanners and printers.

4.4 Possible Improvements

Kae et al. [19] created a document specific OCR for each document. They ran Tesseract on the document to detect a set of reliable characters. Then they used SIFT as a character descriptor and an SVM classifier to OCRize the document. Unfortunately this requires training the classifier for each document and prevents the algorithm from performing in real time, which is necessary for our application. They used a test set of 10 documents and reduced Tesseract's error rate by 20%.

Reynaert [20][21] created a lexicon of word variants to compensate for OCR errors. He applied a space reduction similar to our alphabet reduction and searched for the word in the lexicon in order to find its correct form. This allows for the correction of any errors within a Levenstein distance of 2 e.g. a maximum of two edition operations. The algorithm called TICCL can detect between 55% and 89% of OCR errors on a corpus of Dutch newspapers.

Finally, Niklas [22] combines the work of Reynaert with a new word hashing algorithm called OCR-Key. He achieves an error reduction rate between 39% and 75% on a corpus made of several issues of The Times newspaper between 1835 and 1985. However for these last two works, the use of a lexicon is inapplicable for names or reference numbers combining letters and numbers especially those occurring in invoices, playslips etc.

We also started preliminary testing with Abbyy Finereader which is considered to be more robust than Tesseract by the document analysis community. We

expect an improvement between one and two orders of magnitude for the error rate.

5 Conclusion

We presented our beginning work on securing a hybrid document by creating a document signature based on its content. The issue of robustness is the most critical and needs to be studied for every task involved in the computation of the document signature. This could also be a criteria not to include some document features in the signature. If the extraction algorithm for a document feature is not robust enough, the probability of false positives will be too high and hence should not be included until it has been improved.

We found that the robustness of Tesseract is quite low. While an accuray of 99.83% is quite good; a probability of false positives of 53% is not acceptable. The objective is to have it below 0.1%. Moreover, these figures are obtained in a constrained best case scenario. The scanning resolution has to be at least 600dpi and the font size no less than 10 points. There can be no italic emphasis and no small fonts such as Courier or Times New Roman. The content of the document must also respect some sort of human readability e.g. it cannot be programming code. Another finding is that increasing the scanning resolution from 300dpi to 600dpi does not improve the accuracy much but it significantly reduces the probability of false positives. The collision probability at 0.002 is nearly acceptable and the other performance criteria are satisfied.

One can also wonder about the robustness of segmentation algorithms and of the retranscription of the document structure. Any quantization or randomness will most likely lead to a fragile and/or not reproducible algorithm.

We have shown that the current state of the art does not perform sufficiently well on a seemingly easy task when studied under the angle of robustness. This issue of robustness is of great interest as it could lead to a new document security technology. Considering the work at hand any help or collaboration from the document analysis and security community would be welcome.

References

1. Smith, A.: Identity fraud: a study. Technical Report July, Economic and Domestic Secretariat Cabinet Office (2002)
2. Baras, C., Cayre, F.: Vers un modèle de canal réaliste pour l'analyse de la sécurité du processus d'authentification par code matriciel 2D. In: XXIVème Colloque GRETSI. (2013) 2–5
3. Zauner, C.: Implementation and benchmarking of perceptual image hash functions. PhD thesis, University of Applied Sciences Hagenberg (2010)
4. Bryson, J., Gallagher, P.: Secure Hash Standard (SHS) (2012)
5. Kerry, C.F., Gallagher, P.: Digital Signature Standard (DSS) (2013)
6. Gilbert, H., Handschuh, H.: Security analysis of SHA-256 and sisters. In Springer Berlin Heidelberg, ed.: Selected Areas in Cryptography. Springer Berlin Heidelberg (2004) 175–193

7. Kornblum, J.: Identifying almost identical files using context triggered piecewise hashing. Digital Investigation **3** (September 2006) 91–97
8. Hadmi, A., Puech, W., Said, B.A.E., Ouahman, A.A.: Perceptual image hashing. In Das Gupta, M., ed.: Watermarking - Volume 2. InTech (2012) 17–42
9. Belazzougui, D., Navarro, G., Valenzuela, D.: Improved compressed indexes for full-text document retrieval. Journal of Discrete Algorithms **18** (January 2013) 3–13
10. Winter, C., Schneider, M., Yannikos, Y.: F2S2: Fast forensic similarity search through indexing piecewise hash signatures. Digital Investigation **XXX** (September 2013) 1–11
11. Koval, O., Voloshynovskiy, S., Beekhof, F., Pun, T.: Security analysis of robust perceptual hashing. In Delp III, E.J., Wong, P.W., Dittmann, J., Memon, N.D., eds.: Proc. SPIE 6819, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X, SPIE (February 2008) 1–10
12. Villán, R., Voloshynovskiy, S., Koval, O., Deguillaume, F., Pun, T.: Tamper-proofing of electronic and printed text documents via robust hashing and data-hiding. In Delp III, E.J., Wong, P.W., eds.: Proc. of Security, Steganography, and Watermarking of Multimedia Contents IX. (February 2007) 65051T–65051T–12
13. Tan, L., Sun, X., Zhou, Z., Zhang, W.: Perceptual text image hashing based on shape recognition. Advances in Information Sciences and Service Sciences(AISS) **3**(8) (2011) 1–7
14. Baras, C., Cayre, F.: 2D bar-codes for authentication: a security approach. In: Proc. of 20th European Signal Processing Conference (EUSIPCO), IEEE Sign. Proc. Soc. Press (2012) 1760–1766
15. Malvido Garcià, A.: Secure Imprint Generated for Paper Documents (SIGNED). Technical Report December 2010, Bit Oceans (2013)
16. Haar, A.: On the theory of orthogonal function systems . Mathematische Annalen **69**(3) (1910) 331–371
17. Google: Tesseract (2013)
18. Rice, S., Jenkins, F., Nartker, T.: The fifth annual test of OCR accuracy. Technical Report April, Information Science Research Institute (1996)
19. Kae, A., Huang, G., Doersch, C., Learned-Miller, E.: Improving state-of-the-art OCR through high-precision document specific modeling. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Comput. Soc. Press (2010) 1935–1942
20. Reynaert, M.: Non-interactive OCR post-correction for giga-scale digitization projects. Computational Linguistics and Intelligent Text Processing **4919** (2008) 617–630
21. Reynaert, M.W.C.: Character confusion versus focus word-based correction of spelling and OCR variants in corpora. International Journal on Document Analysis and Recognition (IJDAR) **14**(2) (November 2010) 173–187
22. Niklas, K.: Unsupervised post-correction of OCR Errors. PhD thesis, Leibniz Universität Hannover (2010)