

Authenticating Binary Text Documents Using a Localising OMAC Watermark Robust to Printing and Scanning

C. Culnane, H. Treharne, and A.T.S. Ho

Department of Computing, School of Electronic and Physical Sciences,
University of Surrey, Guildford, Surrey, GU2 7XH
c.culnane@surrey.ac.uk

Abstract. In this paper we propose a new authentication and localisation scheme to produce a watermark which can be embedded in a limited capacity binary text document and that will work in a print and scan environment. The scheme utilises Message Authentication Codes (MAC), specifically OMACs, which create a cryptographic fixed-length summary of a document. An OMAC must be truncated to a form part of our watermark and is used during authentication. The remainder of the watermark is used during localisation. We have created over 2,000,000 watermarks in controlled experiments to evaluate their ability to authenticate a document and localise any changes. In addition, we have embedded an authenticating watermark into seven different documents and authenticated them after printing and scanning.

1 Introduction

In our previous work [1] we increased the capacity available and robustness when embedding in binary text documents robust to printing and scanning. As we stated in our previous work, our goal is to increase the capacity sufficient to enable us to embed a meaningful authenticating watermark. We aim to only embed an authenticating watermark because it is not appropriate to embed a copyright based watermark since text documents can easily be OCR'd (Optical Character Recognition) or even manually re-typed to remove the watermark. There have been recent developments in using natural language watermarking [2], which change the sentence structure and sometimes actual words in order to create the watermark. These still suffer from easy attack, since someone is permitted to use an alternative synonym and this presents the problem of making changes to users' documents. In legal, medical and creative writing it would be unacceptable to make such changes.

The use of Message Authentication Codes (MACs) in authenticating text is not new. In [3] Villán et. al. proposed an embedding system providing far greater capacity, providing between 1 and 3 bits of capacity per character. However, the robustness of the scheme to printing and scanning relies on being able to use the same co-ordinate system for embedding and detection. It is unclear how the same co-ordinate system can be used after the distortions of printing and scanning.

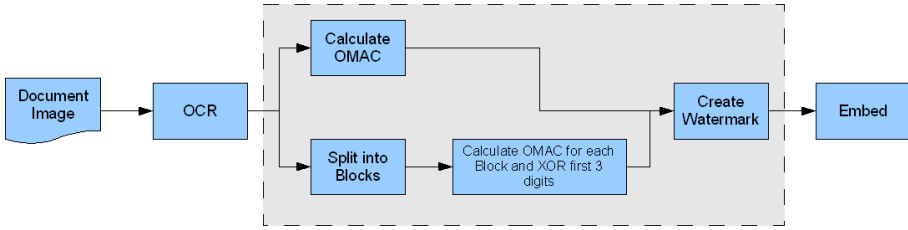


Fig. 1. Embedding Process

Creating an authenticating watermark for a binary text document robust to printing and scanning presents a unique set of problems. One problem is the distortion that occurs during printing and scanning and the other is what information can be represented in the limited capacity watermark and used as a basis of authentication. The way to counter the distortions is to abstract the image to a more constant value. This can be achieved by using OCR to convert the image of the document back into text. It does not matter if the position of the letter has moved a few pixels in some direction, or if a few pixels have been flipped from black to white or vice-versa. The OCR process is flexible enough to counter most of the distortions seen during printing and scanning. Whilst this can still be susceptible to some distortion, modern OCR packages allow user input in order to accurately correct any mistakes and to learn from those mistakes. The area most susceptible to distortion is the recognition of whitespace. It is for that reason that during the processing of the OCR output we strip all the whitespace. The removal of the whitespace will have no impact on the meaning of the document. Figure 1 is an overview of our authentication system and involves performing an OCR of the document as the first part of the embedding process.

The first contribution of the paper is the creation of an authenticating and localising watermark which can be embedded within a document with limited

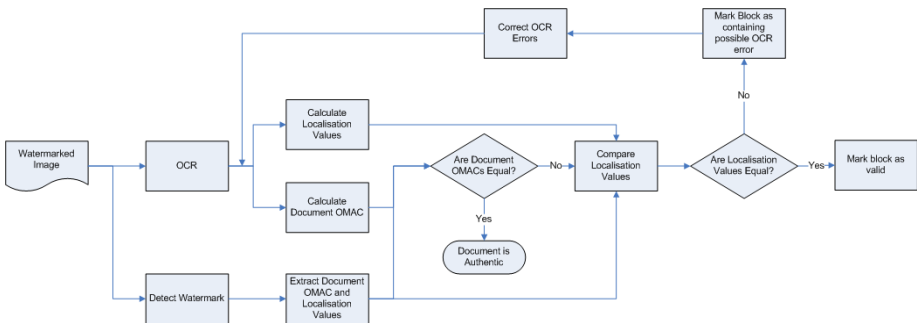


Fig. 2. Flow Chart for the Detection Process

capacity. This creation process is highlighted by the dotted box in Figure 1 and detailed in Section 3. Section 4 describes the structure of the watermark. Once a watermark is created it can be embedded into a document. This embedding process is summarised in Section 2 and based on our previous work.

The detection process is outlined in Figure 2. In contrast to traditional image authentication methods [4] that localise the document and then authenticate the localised blocks, we authenticate the document as a whole and subsequently attempt to localise any changes, and this will be an iterative process. We have to take this approach in order to comply with the capacity limitations.

Section 5 explains how we localise any errors during the detection process and this is another contribution of the paper. Section 6 summarises the experiments we conducted on authenticating documents and localising any errors.

2 Watermarking

In [1] we proposed a method for watermarking binary documents in a print and scan robust way. This was an extension of our work in [5] which was based on the work by Zou and Shi in [6]. Data is embedded by creating sets of word spaces, the whitespace between words, and making changes to the widths of these spaces to create a detectable difference between two sets. In [1] we further increased the capacity by embedding in a continuous line, instead of the inefficient line by line method used previously. Even with the improvements the average capacity is still only 98 bits. The combination of limited capacity and the need to handle printing and scanning are constraining factors in the identification of an authenticating watermark. However, in this paper we have succeeded to work within these constraints and developed a new authentication and localisation scheme.

3 What Is an OMAC

An OMAC is a One-key Message Authentication Code (MAC) equivalent to a CMAC (Cipher-based MAC) [7]. A MAC takes a message of arbitrary length and a secret key, it outputs a fixed length tag (MAC) that can be used to verify the integrity and authenticity of the message. Due to the MAC being created with a symmetric key, MACs do not offer any proof of non-repudiation as seen in digital signatures.

3.1 Alternative Authentication Methods

The traditional approach to authentication is to create a hash. A cryptographic hash function is a way of getting a fixed length output, called a digest, from an arbitrary length input in a manner whereby an attacker cannot create a second message that generates the same output or find a previously unseen message from a given digest. The two most common cryptographic hashes are SHA-1 and MD5. MD5 was shown to have security weaknesses many years ago and is rarely used in cryptography today. SHA-1 is still very much in use although

recent research has demonstrated potential weaknesses, see [8]. We cannot use either algorithm because of the size constraints. MD5 produces 128 bit hashes, whilst SHA-1 produces 160 bit hashes. Both of which easily exceed our 100 bit capacity requirement. SHA-1 is a cryptographic hash function, but it is not equivalent to an OMAC or indeed a HMAC. The SHA-1 algorithm is well known and as such anyone can generate a SHA-1 hash of anything. Typically a SHA-1 hash is encrypted using the private key from an asymmetrical encryption system. When someone wants to authenticate the message they decrypt the block using the public key, re-calculate the hash and compare them. SHA-1 can be combined with an HMAC which is similar to an OMAC. The reasons for not using such a scheme are given in Section 3.2. We also considered the use of Cyclic Redundancy Checks (CRCs) although these are easily attacked and are used more for error detection as opposed to authentication. The advantage of CRCs was the range of output sizes they provided 16, 32 and 64 bits. However, their lack of security meant they were no comparison to an OMAC.

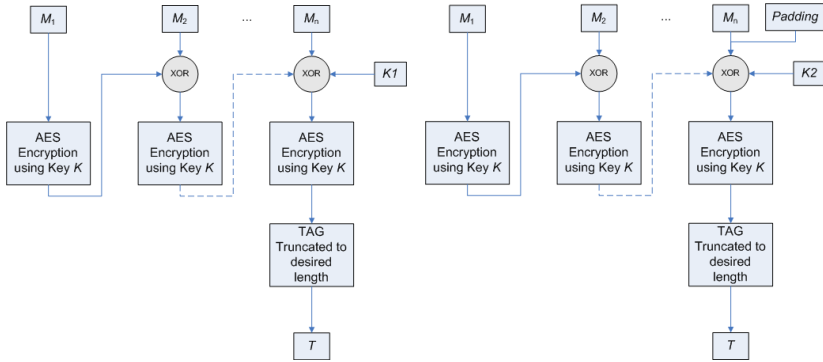
3.2 Selection of OMAC as Opposed to HMAC or SHA-1

We selected the OMAC algorithm for a number of reasons. Its equivalent, the CMAC, has been approved by NIST (National Institute of Standards and Technology, USA). There is a Java implementation of OMAC in the public domain and there are no known patent claims over the methods used. The OMAC can use any block based cipher and does not require the use of a cryptographic hashing algorithm. This allows us some flexibility in selection of the block based cipher. OMACs use a cipher block to encrypt the data as it is processed. This means someone without the key cannot create an OMAC and hence the key is required during authentication. There is never any decryption, thus allowing the OMAC to be truncated. This is not something that would be possible in a system that required decryption. The HMAC relies on a cryptographic hash function and typically uses SHA-1, however, some security concerns have been raised about SHA-1 [8] and truncating it may create greater weaknesses. The SHA-1 algorithm is due to be phased out in 2010 [9] and therefore its use in a new system is not prudent. Thus, an OMAC implementation based on an AES key is appropriate for our use because we will need the flexibility of truncation.

3.3 Overview of How OMACs Are Calculated

As stated above OMACs are based on the use of a block cipher, in our case AES. This block cipher requires a key K and operates on a block whose length is b , in this case 128 bits. Two subkeys $K1$ and $K2$ are derived from K . Both $K1$ and $K2$ are the same length as the block. During the process of generating the subkeys a pre-determined bit string is used. This bit string is based on the block length. In our case, 128 bit length, the bit string is defined as follows: $R_{128} = 0^{120}10000111$. The message from which to generate an OMAC is denoted as M and is $Mlen$ long, in terms of bits. The output of the OMAC generation is termed a TAG and denoted by T .

The full notation of the OMAC/CMAC algorithm is available from [7]. Our aim below is to provide an overview as opposed to a full definition.



(a) Last Message Block is Complete (b) Last Message Block is Incomplete

Fig. 3. Illustration of the OMAC/CMAC generation process (Images adapted from [7])

SubKey Generation. A string of ‘0’ bits of the same length as the AES block b is encrypted using the AES key K , the output is denoted as L . If the most significant bit of L is a 0 the subkey $K1$ is equal to L bit shifted to the left by 1, otherwise $K1$ is equal to $L1$ bit shifted to the left by 1 and XOR’d with R_{128} . (Bit shifting involves removing the left most bit and appending a 0 zero bit to the right.) The generation of $K2$ is in essence the same except instead of using $L1$ we use $K1$. As such, if the most significant bit of $K1$ is a 0 the subkey $K2$ is equal to $K1$ bit shifted to the left by 1. Otherwise, $K2$ is equal to $K1$ bit shift to the left one and XOR’d with R_{128} .

MAC Generation. Given that we have already obtained K , $K1$ and $K2$, if the length of the message M is 0 then $n = 1$ otherwise $n = Ceiling(Mlen/b)$. The *Ceiling* refers to the first integer not less than the expression. So *Ceiling*(1.2) is 2 whilst the *Ceiling*(3) is 3. The value of n is used to break the message up into blocks M_1, \dots, M_n . The last block of the message is then processed as follows: if it is a complete block, contains 128 bits, M_n is made equal to M_n XOR $K1$. Otherwise M_n is made equal to M_n padded to the correct length XOR $K2$. Details of the padding are available in [7].

C_0 is defined as the output from the next step. It is initially set to be a bit string of zeros equivalent to the block length. For $i = 1$ to n , let C_i be equal to the encryption of C_{i-1} XOR M_i . In essence each message block is XOR’d with the output from the previous block’s encryption and encrypted again. T is a truncation of the bit string to the desired length, taking the most significant bits.

Figure 3a and Figure 3b show the process in a graphical format. These images were adapted from [7].

4 Watermark Structure

Figure 4 is a diagram of the basic watermark structure. We use OMACs in two ways, firstly to authenticate the document and secondly to localise any errors

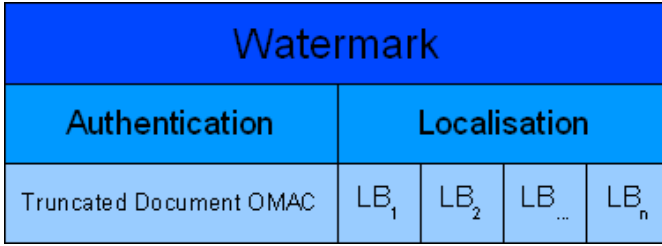


Fig. 4. Watermark Structure Diagram

found. We recognise that the OCR process may not be flawless and as such we want to be able to provide a guide as to where any error may have occurred. It is possible that OCR errors could occur during the embedding phase. We assume that should any such an errors occur it is likely they will re-occur during detection and therefore not cause a problem. An OMAC is generated for the whole document and then another OMAC is created for each localisation block. See Section 5 for more details on this. The principle of our system is that the secret key used in the OMAC will not be distributed, but held in a central location or shared between two trusted parties in advance. If someone wishes to verify a document they either email the document or bring it for verification. If the key was released it would allow anyone to change the document, generate another OMAC and embed it. Thus, defeating the security. It is important to record and potentially limit the number of times a document can be verified due to the truncation of the OMAC.

As can be seen from Figure 4 the watermark capacity is split in two. One half for authentication and one half for localisation. The authentication half contains the OMAC calculated over the entire document truncated to fit into the available size. The localisation half is further split into localisation blocks, each of which is 4 bits long.

4.1 Why Split the Watermark in Half?

How the watermark is divided needs to be decided in advance and the same for all watermarks created within that system. We do not have the capacity within the watermark to embed structural information such as where the division between authentication and localisation takes place. We choose to split the watermark in half as we considered localising any errors as important as authentication and are willing to accept the reduced security of a smaller truncated watermark. The impact of reducing the localisation size is that localisation blocks become larger and it becomes more difficult for a user to find and correct any OCR errors.

5 Localisation

As was mentioned above the OCR may not be flawless and there may be a need to correct any OCR errors if the authentication fails. In order to do this we

need to localise where the error is, to allow the user to quickly find and correct any errors. The biggest challenge to localising authentication errors is the small amount of capacity we have available. As was discussed in Section 4 we devote half our capacity to localisation. This is still a very small amount, 50 bits over approximately 50 lines. The method we propose is a balance between successful localisation and reduced space.

5.1 Division of the Document into Localisation Blocks

Once the document image has been OCR'd we have a text document containing the lines of OCR'd text. We read this document in and create an array of strings, each item in the array containing one line. We then count the number of lines we have and the available capacity. We allocate 4 bits per localisation block. We experimented with allocating 5 bits but there was no noticeable improvement in results. We divide the available capacity by 4 (the number of bits per block) and this gives us the number of localisation blocks we can create. We then divided the number of lines in the document by the number localisation blocks, giving us how many lines will be in each block. As we process each line during the OMAC calculation we also create the localisation blocks and calculate an OMAC per localisation block. We can express this process using the following equations:

$$L_b = (W_c/2)/4$$

$$L_s = D_l/L_b$$

where D_l is the number of lines in the document, W_c is the capacity of the watermark, L_b is the number of localisation blocks, and L_s is the number of lines in a localisation block.

Obviously, we cannot store the whole OMAC for each localisation block, since we only have 4 bits of capacity. We could just choose to store one of the OMAC digits, for example the first or last digit. However, this would not be the most efficient use of the 4 bits of capacity. It also would be prone to errors, whereby two different OMACs are treated as the same. We therefore decided to XOR the first three digits. We experimented with XORing all the digits but again saw no improvement. This is most likely due to the inherent limits of the XOR operation.

5.2 Example

Let $D_l = 50$ and $W_c = 100$, then we can calculate the following:

$$L_b = (100/2)/4 = 12$$

$$L_s = 50/12 = 5$$

Note that the rounding on the number of blocks is always down. It is better to have unused bits than not enough space to embed the final localisation block data. Also the rounding is always up on the block size calculation. This is to ensure all lines are part of some block.

The following is an example of an OMAC value for a localisation block: 11624770511598628777770661211885184331 (and corresponding to 128 bits and we only have 4 bits available). If we then take the first three digits and XOR them we get the following (we XOR the first two, then the XOR the result of that with the third):

Table 1. OMAC values to be XOR'd and XOR value

Decimal	Binary
1	0001
1	0001
6	0110
	0110

Suppose we change one character in the first block (an 's' to a 't') we then obtain the following OMAC value for that block: 3782238743673572791382206625495005089. This gives use the following XOR output:

Table 2. OMAC values to be XOR'd and XOR value after character change

Decimal	Binary
3	0011
7	0111
8	1000
	1100

Thus, during detection when comparing the two XOR values, they will not be equal and the block will be marked as having changed.

5.3 Localisation of a Document

Before the watermark is created an XOR value for each localisation block is calculated. These are then combined into the complete watermark, using the structure discussed in Section 4. During the detection an XOR value of the OMAC for each localisation block is calculated (see Calculation of Localisation Values box in Figure 2). The values stored in the watermark are retrieved and are compared with the computed value. If they are not the same the block is marked as potentially containing an error. The system is not perfect, since it is possible that a block cannot be localised, this can occur because there is not a 1:1 mapping between possible digit combinations and XOR values. This is due to the capacity limitations. However, a localisation block should never be indicated as changed when it has not changed. Also the OMAC for the entire document will still function correctly even if the localisation fails.

6 Experimentation and Analysis

Our experiments were done in three stages: we examined the theoretical limits of using a truncated OMAC, conducted a number of controlled experiments, and performed print and scan tests.

6.1 Theoretical Analysis

How secure is the authentication? The level of security provided by an OMAC has been proved using cryptographic proofs in [10], it is beyond the scope of this paper to re-examine those proofs. However, those proofs are based on using a full length OMAC. As we stated in Section 4 we use a truncated OMAC and hence need to evaluate the impact of such a reduction in the security.

It should be noted that a weakness of the OMAC is if an attacker has access to difference OMACs generated with the same key as was shown in [11]. We could protect against this attack by using a different key for each OMAC.

Birthday Paradox and Birthday Attack. The birthday paradox is part of probability theory, it states that given a group of 23 or more people the probability of two of them having the same birthday is at least 50%. This is important as it forms the basis of the birthday attack [12]. The birthday attack is a cryptographic attack that determines how many times we will need to brute force attack a hash before we find a collision. In our case a collision is when two different documents produce the same OMAC.

If our OMAC has a output length of m bits then to a collision we will need to calculate $2^{m/2}$ different OMACs in order to find a collision. This allows us the calculate a quantitative measure of how secure our OMAC is depending on the length of the truncated OMAC. For example, with a length of 50 bits we would need to calculate $2^{50/2}$ OMACs in order to have a high probability of finding a collision. However, a collision may be found within fewer calculations.

6.2 Controlled Experiments

As part of our evaluation of the authentication and localisation schemes we conducted five controlled experiments outside of the print and scan environment. These involved making controlled changes to the text in order to evaluate the effectiveness of the authentication and localisation schemes. The experiments resulted in over 2,000,000 tests.

In each of the experiments we took the equivalent of a page of text and created a watermark for it. We then made controlled changes to the document, re-calculated the watermark and compared the OMACs. We expected the OMACs to be different and the aim was to localise the changes. We made the following five types of changes:

1. Single Character Change
2. Dual Character Change with Same Character Change
3. Dual Character Change with Same Size Change

Table 3. Experimentation Results

Experiment Name	Number Hash Values Calculated	Collisions	Localisation Errors
Single Character Change	424080	0	27648
Dual Character Same Character Change	424080	0	82625
Dual Character Same Size Change	424080	0	79928
Dual Character Random Change	424080	0	83759
Dual Character Same Size Oposite Change	424080	0	80887
Total	2120400	0	354847

4. Dual Character Change with Random Character Change
5. Dual Character Change with Opposite Size Change

Further details of each experiment and their results are given below. When making changes to characters we used characters with ASCII values between 33 and 126 (inclusive). This is basically the range of human readable characters, excluding whitespace. The sample document consisted of 19 pangrams repeated to form a document consisting of 4560 characters (excluding whitespace). We choose pangrams to ensure we had a range of all characters. Therefore, 424080 watermarks were calculated, $(126-33)*4560$, when conducting each of the following experiments.

Single Character Change. This experiment is designed to mimic the small changes we are likely to see during OCRing process. It involves iterating through each character in the document and changing it to each one of the characters within the predetermined range, as shown in Figure 5. After each change the watermark is re-calculated. It is unlikely that someone could meaningfully attack a document based on changing just one character, unless it was a number. Table 3 shows that during this experiment there were no collisions. However, there were a number of localisation errors. The number of localisation errors should be viewed in the context of having performed over 424000 tests. The error was correctly localised in over 93% of the cases. The failure of the localisation method in no way affects the authentication process.

The quick brown fox jumps over the lazy dog
 ↓
 The quick hrown fox jumps over the lazy dog

Fig. 5. Single Character Change Diagram

Dual Character Change with Same Character Change. This experiment is to test how well the method works with two seemingly unconnected changes. This experiment also involves iterating through each character in the document and changing it to each of the characters in the predetermined range. For each change we also randomly select another character from the document and change that to the same character. For example both the characters ‘b’ and ‘o’ change to ‘h’ in Figure 6. This also allows us to further test that the localisation method can pick up changes in two different locations. Table 3 shows that during this

experiment there were no collisions. There were more localisation errors, but this is due to more blocks being changed and thus the chance of localisation errors increases. Even so the localisation method worked in over 80% of the tests.

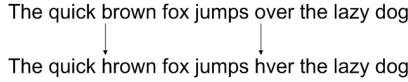


Fig. 6. Dual Character Same Change Diagram

Dual Character Change with Same Size Change. In this experiment we iterate through the characters in the document changing it to each of the possible characters. We calculate the size of the change then pick another character at random and make the same size change to it. For example in Figure 7 both changes increase by six. It is possible that in some cases the character that is picked at random may become a non-human readable character since it will be outside of the ASCII range. We choose to allow this since Java uses Unicode to represent characters and as such the value will not be outside of the acceptable range of characters. Again, Table 3 shows that there were no collisions. The number of localisation errors was similar in size to the one found in the Dual Character Change with Same Character Change.

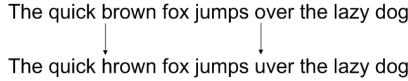


Fig. 7. Dual Character Same Size Change Diagram

Dual Character Change with Random Character Change. This experiment is an extension of the second experiment with more randomness. Figure 8 shows an example of the type of change undertaken. We iterate through each of the characters and change it to each of the possible characters. We then pick another character at random and change that to another random character. We can see from Table 3 that no collisions occurred and the localisation errors remained in the same region as the other similar experiments.

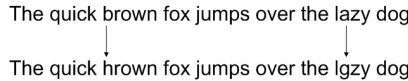


Fig. 8. Dual Character Random Change Diagram

Dual Character Change with Opposite Size Change. This is designed to test a deliberate attack on the system. By making an opposite change it means that the byte values will remain balanced. Since one is increased and another is decreased by the same amount. This experiment follows the same lines as the one described in Dual Character Change with Same Size Change except we make the

opposite change to the second character. Figure 9 illustrates the type of change we make. Table 3 shows that again there were no collisions. The number of localisation errors was inline with the other experiments. The localisation errors are not critical, since they are used as a way of helping the user to correct any OCR errors or to see where any attack may have taken place.

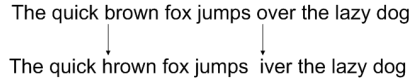


Fig. 9. Dual Character Opposite Size Change Diagram

6.3 Print and Scan Experiments

The above experiments demonstrated that a watermark based on OMACs successfully works given our capacity constraints. We also conducted experiments to evaluate the new method in a print and scan environment by embedding authenticating watermarks into seven documents, each formatted using a different font. The document images were at 300dpi and the font size in each case was 12pt. The fonts tested were: Arial, Arial Narrow, Comic Sans, MS Sans Serif, Tahoma, Times New Roman and Verdana. These were the same fonts as were used in our previous work [1]. The documents were printed and scanned on a HP PSC 2110 and we used ReadIris Pro 9 to do the OCRing. We could not include the Courier New font because it could not be OCR'd correctly. This was due to the large whitespaces in the document causing it to be fragmented into columns by the OCR Package.

Table 4 shows the results from the printing and scanning experiment. The first, second and third pass reflect the iterative nature of the process. Once a document has been authenticated we do not authenticate it again in any further passes. The values in the cells indicate whether the document was authenticated (**Valid**), contained OCR errors (e.g. 1 OCR) or contained noise (Noise). The noise is a result of the printing and scanning process and as is mentioned in [1] needs to be removed manually.

Table 4. Print and Scan Results

Font	W_c	D_l	L_s	Pass		
				First	Second	Third
Arial	105	50	4	Noise	Valid	
Arial Narrow	122	48	3	1 OCR	Valid	
Comic Sans	82	50	5	1 OCR	Valid	
MS Sans Serif	108	51	4	Noise	2 OCR	Valid
Tahoma	100	47	4	Valid		
Times New Roman	114	49	4	3 OCR	1 OCR	Not Valid
Verdana	87	47	5	1 OCR	Valid	

The Tahoma document was authenticated without the need for any corrections. The MS Sans Serif document and the Arial document both contained noise that had to be manually removed in order to correctly recover the watermark. Once the noise had been removed the Arial document authenticated correctly. The MS Sans Serif document contained two OCR errors each error was correctly localised to four lines in the 51 line document. After correcting those errors the document authenticated.

Comic Sans, Arial Narrow and Verdana documents all contained OCR errors that were correctly localised to within L_s lines, as shown in Table 4. Once these OCR errors had been corrected, they too authenticated. Even though all the OCR errors were corrected the Times New Roman document could not be validated due to a watermarking error. One bit was flipped due to distortion and that resulted in an incorrect OMAC being extracted from the watermark. This prevented the document from authenticating. This highlights the fact that should the watermark be lost or damaged it operates in a fail-safe manner and does not authenticate the document. These results show that it is an iterative process to remove the OCR errors before a document can be authenticated. Further work is needed to counter the problems caused by detection errors.

7 Conclusion

In this paper we have introduced an authentication and localisation scheme that can be embedded in a limited capacity watermark. The watermarks are typically less than 100 bits. We authenticate the document as a whole and cannot authenticate individual localisation blocks because this would require storing the entire OMAC for each localisation block which would exceed our capacity limitations. The ability to localise errors to within at most 5 lines allows the scheme to handle the distortions introduced during printing and scanning.

capacity is split in two, half for authentication and half for localisation. The larger the capacity in the document the greater the level of authentication security and the fewer the lines in the localisation block. The size of the watermark is calculated dynamically and therefore is adaptive to the capacity of the document. We could further emphasise security or localisation by changing how we split the watermark structure. For a more secure authentication the localisation can be reduced and a larger document OMAC embedded. Likewise if localisation is more important then authentication can be reduced.

We conducted over 2,000,000 million controlled tests and successfully authenticated in all of them. It should be noted that this to be expected since we would need to conduct $2^{50/2}$ tests in order to have a high probability of finding a collision. In over 83% of cases the changes were correctly localised. We have demonstrated that the authentication of the document is still sound, even with a truncated OMAC. We also demonstrated that even if the localisation fails it in no way affects the authentication.

We used the same authentication and localisation scheme in a print and scan environment. We successfully localised all of the documents and authenticated six out of the seven. The one failure was caused by a watermark detection error.

8 Future Work

As we discussed above the watermark is split in a 50-50 manner, however, that may not be ideal for specific applications. It was sufficient as a proof of concept, but we would like to develop the structure further so greater emphasis can be placed on authentication or localisation depending on the requirements of the user.

There are a number of different potential applications for the scheme we have proposed. The two most significant are authenticated archiving and authenticated message transfer. In authenticated archiving, the goal is to be able to archive the document and then at a later date authenticate that it is genuine. This is applicable to legal and medical situations. In this scenario there would be a trusted third party that would create the watermark and provide authentication services. None of the interested parties would have access to the AES key, therefore providing protection from malicious changes.

Authenticated message transfer aims to provide an authenticated link between two parties who trust each other. This is applicable to situations where printed text documents need to be sent through potentially hostile locations. It is particularly relevant to Emergency Procedure Plans and Work Procedures. These are often sent via couriers and there is currently no way of authenticating that the document received has not been changed. The two parties would need to agree on a shared AES key in advance. This could be done using established protocols currently used for generating session keys. Alternatively, the AES Cipher could be replaced with a Password Based Encryption (PBE) Cipher which generates the encryption key from a password. The two users could then exchange the password.

Acknowledgements. We would like to thank the reviewers for their helpful comments.

References

1. Culnane, C., Treharne, H., Ho, A.T.S.: Improving multi-set formatted binary text watermarking using continuous line embedding. In: IEEE International Conference on Innovative Computing, Information and Control (ICICIC 2007) (to appear, 2007)
2. Topkara, M., Topkara, U., Atallah, M.J.: Words are not enough: sentence level natural language watermarking. In: MCPS 2006: Proceedings of the 4th ACM international workshop on Contents protection and security, pp. 37–46. ACM Press, New York (2006)

3. 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.) *Security, Steganography, and Watermarking of Multimedia Contents IX*, February 2007. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, vol. 6505, p. 65051T (2007)
4. Miller, M.L., Cox, I.J., Bloom, J.A.: *Digital Watermarking*, 1st edn. Morgan Kaufmann, San Francisco (2002)
5. Cullane, C., Treharne, H., Ho, A.T.S.: A new multi-set modulation technique for increasing hiding capacity of binary watermark for print and scan processes. In: Shi, Y.Q., Jeon, B. (eds.) *IWDW 2006*. LNCS, vol. 4283, pp. 96–110. Springer, Heidelberg (2006)
6. Zou, D., Shi, Y.Q.: Formatted text document data hiding robust to printing, copying and scanning. In: *IEEE International Symposium on Circuits and Systems (ISCAS 2005)* (2005)
7. Dworkin, M.: Recommendation for block cipher modes of operation: The cmac mode for authentication. Special Publication 800-38B, NIST (May 2005)
8. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) *CRYPTO 2005*. LNCS, vol. 3621. Springer, Heidelberg (2005)
9. NIST. Nist brief Comments on Recent Cryptographic Attacks on Secure Hashing Functions and the Continued Security Provided by SHA-1. Internet (August 2004), http://csrc.nist.gov/hash_standards_comments.pdf
10. Iwata, T., Kurosawa, K.: Omac: One-Key CBC MAC. In: Johansson, T. (ed.) *FSE 2003*. LNCS, vol. 2887, pp. 129–153. Springer, Heidelberg (2003)
11. Mitchell, C.J.: Partial key recovery attacks on XCBC, TMAC and OMAC. In: Smart, N. (ed.) *Cryptography and Coding 2005*. LNCS, vol. 3796. Springer, Heidelberg (2005)
12. Schneier, B.: *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York (1996)