Detecting Modifications in Paper Documents: A Coding Approach

Yogesh Sankarasubramaniam, Badri Narayanan†, Kapali Viswanathan and Anjaneyulu Kuchibhotla

HP Labs India, Bangalore, India

ABSTRACT

This paper presents an algorithm called CIPDEC (Content Integrity of Printed Documents using Error Correction), which identifies any modifications made to a printed document. CIPDEC uses an error correcting code for accurate detection of addition/deletion of even a few pixels. A unique advantage of CIPDEC is that it works blind – it does not require the original document for such detection. Instead, it uses fiducial marks and error correcting code parities. CIPDEC is also robust to paper-world artifacts like photocopying, annotations, stains, folds, tears and staples. Furthermore, by working at a pixel level, CIPDEC is independent of language, font, software, and graphics that are used to create paper documents. As a result, any changes made to a printed document can be detected long after the software, font, and graphics have fallen out of use. The utility of CIPDEC is illustrated in the context of tamper-proofing of printed documents and ink extraction for form-filling applications.

Keywords: Paper documents, modification detection, error correction, ink extraction, content integrity

1. INTRODUCTION

Despite numerous predictions of a 'paperless world',[1] paper continues to thrive today. Paper documents are important not only in traditional government processes and social communities, but more so in hi-tech enterprises and businesses. The primary reason for the endurance of paper is its affordability and ease of use, which makes it irreplaceable for certain tasks. Even in today's 'digital age', government offices, financial firms, educational institutions, and business enterprises continue to issue and verify thousands of paper documents each day. However, the seamless inter-operation of the paper and digital worlds has posed considerable challenges.

One stumbling block in this regard has been the susceptibility of paper documents to forgery leading to fraud. Valuable documents like contractual documents and certificates can be easily manipulated by modifying small parts of the authentic document. This is referred to as the 'content integrity' problem. Another hurdle in bridging the paper-digital divide is the extraction of handwritten ink from printed material, for example in form-filling applications.

In this paper, we present a novel solution that holds much promise to address the above two challenges. We refer to this solution as CIPDEC: Content Integrity of Printed Documents using Error Correction. CIPDEC uses error correcting codes (ECC)[7,8] to not only detect any additions/deletions made to a printed document, but also locates and identifies these changes up to a pixel-level accuracy. For ease of understanding, we primarily describe CIPDEC as a 'content integrity' solution in Section 3. However, it will be evident that the basic idea of detecting modifications using ECC is applicable in several other scenarios. One such application of CIPDEC in ink extraction will be described later in Section 6.

The basic idea behind CIPDEC is illustrated in Fig. 1. There are two stages, namely: generation and verification. The generation stage occurs prior to printing of a paper document, while the verification stage ascertains the integrity of a printed and scanned document, which may be modified maliciously. Any pixel-level change made to the printed document can be caught during the verification stage. The ECC parities are

† Work done while the author was with HP Labs India

Further author information: (Send correspondence to Yogesh Sankarasubramaniam and Kapali Viswanathan)
E-mail: {yogesh,kapali}@hp.com

Document Recognition and Retrieval XVII, edited by Laurence Likforman-Sulem, Gady Agam,
Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 7534, 75340A · © 2010 SPIE-IS&T
CCC code: 0277-786X/10/$18 · doi: 10.1117/12.838122

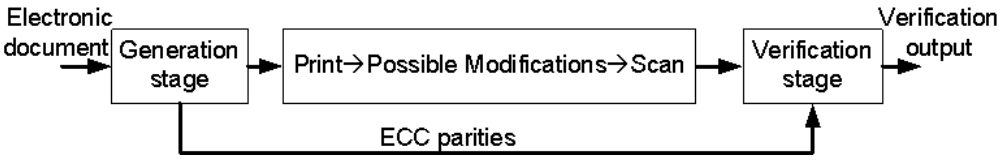

Figure 1. **B**road idea behind CIPDEC

fundamental for detection of pixel-level tampering. As seen from Fig. 1, the ECC parities are computed during the generation stage and used during the verification stage. Intuitively, any additions/deletions made to the paper document are treated as pixel errors, and are 'corrected' by the ECC.

The 'Electronic document' in Fig. 1 refers to the electronic document image to be printed on paper. The original electronic document (which may include text, figures, graphics, tables, and any other non-textual content) is treated as an array of pixels or a bitmap, which is available as the last step before printing the document. Thus, CIPDEC is independent of software, language, font and graphics used to create the paper document. Therefore, the content integrity of arbitrary documents, which may be printed using obsolete or unknown software, fonts and graphics, can still be verified. CIPDEC is also compatible with legacy paper documents that do not have a corresponding electronic original. A scanned image of the paper document can act as the 'electronic document' in such a case.

2. RELATED WORK

There have been wide-ranging efforts towards detection of fraudulent alterations to paper documents, including advances in information forensics and signature matching. However, these techniques have inherent limitations which a determined forger can easily exploit. For instance, studies have shown that around 6.5% of the time a forged document passes verification and 26.1% of the time an authentic document fails verification. Another stream of work relates to robust hash functions[2–4] and visual cryptography.[5,6] While the design of a robust hash for print/scan applications continues to be an open challenge, the requirement in CIPDEC is to detect additions/deletions at pixel-level precision, where even robust hash functions may not suffice.

Other traditional approaches to solve this problem are based on optical character recognition (OCR) and document image processing techniques. These solutions typically detect modifications by comparing document content against the original content. Thus, unlike CIPDEC, these techniques can be said to be 'non-blind'. Furthermore, such OCR/image processing based techniques do not offer the pixel-level accuracy of CIPDEC. Their ambit is usually limited to a given range of font sizes/languages/software, and they do not handle manual edits such as handwritten annotations. More recently, there have been other efforts at digitally signing a machine-readable version of the text. However, such solutions do not directly protect the entire printed content, which may include graphics, figures, and other important non-textual features.

CIPDEC fundamentally differs from all the above mentioned techniques. Firstly, a paradigm difference is that CIPDEC uses error correcting code (ECC) parities,[7,8] which can not only ascertain content integrity but also locate and identify any pixel-level modifications; and secondly, CIPDEC works 'blind', i.e., it does not require the original document during verification. Instead, it only requires the ECC parities, which are a fraction ($x = 5\%$ to 30%, typically) of the original document size. This allows the ECC parities to be carried along with the printed document in a secure machine-readable form. The ECC parameter 'x' is tunable according to the required error correction level.

To our best knowledge, this is the first effort at using error correcting codes to detect pixel-level modifications on printed documents. The advantages of such an ECC-based approach include font/software/language independence, blind verification, and precise detection of malicious modifications of even a few pixels. On the other hand, one additional requirement of the CIPDEC as compared to other non-blind solutions is the use of certain markers on the document. These markers are inserted prior to printing the document, and are crucial to the high pixel-level fidelity.

3. CIPDEC: CONTENT INTEGRITY OF PAPER DOCUMENTS USING ERROR CORRECTION

We introduce the following abbreviations to aid our discussion: ODI - Original document image: this is the same as 'Electronic document' shown in Fig. 1; MDI - Marked document image; CPD - Candidate paper document; SDI - Scanned document image; RDI - Reconstructed document image; CDI - Corrected document image.

A detailed flow-diagram of CIPDEC is given in Fig. 2. The steps involved in generation and verification stages are indicated using dashed and dotted lines, respectively. Let us first take a closer look at the generation stage. The electronic original document image (ODI) is treated as an array of pixels, for e.g. bitmap representation, and error correcting code (ECC) parities are computed on the ODI pixels. Two types of fiducial markers - corner markers and dot markers - are then added to the ODI to yield the marked document image (MDI). Corner and dot markers are shown for one example in Fig. 3 on the left. The corner markers are especially designed to serve a dual purpose - they provide coarse location for the dot markers, and they also offer auto-calibration across printers and scanners. The dot markers are specifically chosen to be unobtrusive in the paper world, and are crucial to the high pixel-level precision of the document image reconstruction (DIR) step during verification.

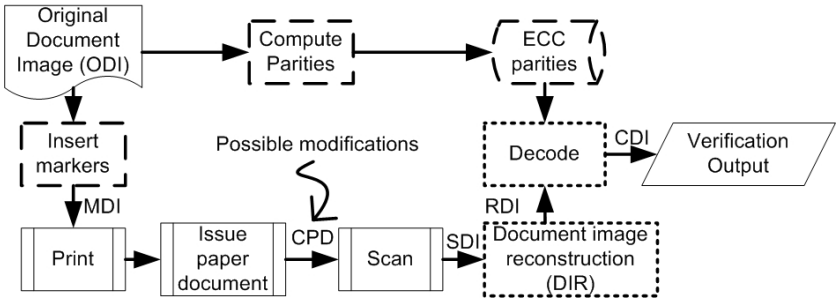

Figure 2. CIPDEC flow diagram

Each step of the generation stage is now formally listed down as follows. For reasons of simplicity, we assume that the ODI and RDI have a binary pixel level (either black (0) or white (1)). Extension to include multiple pixel levels proceeds along similar lines.

1. Let the ODI be an $N \times M$ array of pixels. Perform suitable cropping, smoothening, removal of unwanted data, and other 'clean up' operations if/as required.

2. Compute error correcting code (ECC) parities over the ODI pixels - our implementation uses RS codes[7–9] over GF(512), as described in Section 5

3. Store the ECC parities in an easily retrievable, secure form - for example, as digitally signed machine-readable data along with the issued paper document, or in a secure database.

4. Divide the ODI into cells - each cell is a square $T \times T$ array of pixels. In the example of Fig. 3 we have $N = M = 510$ pixels and $T = 30$ pixels.

5. Insert dot markers on the cell corners - any suitable scheme can be used for this purpose. In the example of Fig. 3, the ODI is scaled to an 8-bit representation (black=0 and white=255), and the dot markers are single pixels of gray-value 150. We found that this offers both unobtrusiveness and easy detection. Alternatives for dot markers include using other suitable pixel levels including black, certain suitable shapes, or any other markers that are visually unobtrusive but easily detected upon scanning.

6. Insert corner markers of size $C \times C$ pixels on the four corners of the dot-marked ODI. For the example of Fig. 3, $C = 15$ pixels.

The document image at the end of the generation stage is called the marked document image (MDI). The MDI is then printed (using a suitable resolution printer), and makes its way through the paper world with possible

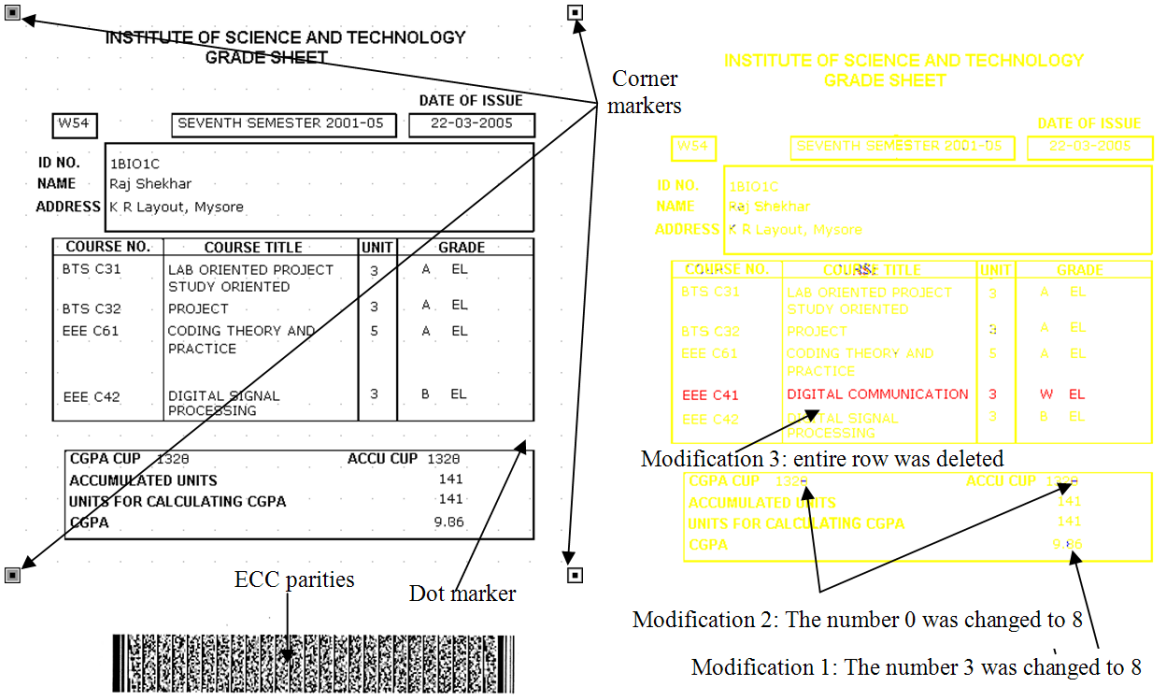

Figure 3. CPD with the dot markers and corner markers is shown on the left. On the right is shown the color coded verification output (CCO), which captures the modifications. The yellow (used to indicate unmodified content) may appear as a faint shade of gray on b/w prints.

additions/deletions, manual edits, annotations, wear and tear associated with paper handling and storage - such as photocopying, folds, stains, marks, staples, bruises etc. When the paper document is ready to cross back into the digital world, it is referred to as the candidate paper document (CPD).

The verification stage is the return-leg of CIPDEC. As seen from Fig. 2, the CPD is first scanned to yield the scanned document image (SDI), which is again an array of pixels, but now distorted by the printing/scanning/paper-world modifications. The printing/scanning distortions are first tackled using the document image reconstruction (DIR) step. Here, a pixel-precise digital image is reconstructed with the aid of the dot markers. The resulting digital image is referred to as the reconstructed document image (RDI). By virtue of the high fidelity of the DIR step, the modified pixels in the RDI can then be identified using the ECC parities by the corresponding ECC decoding procedure. The decoded RDI is referred to as the corrected document image (CDI). The 'difference' between CDI and RDI captures the modifications made to the paper document, which is then suitably displayed as the verification output to a human verifier. We now formally list out each step in the verification stage.

1. Scan the CPD using sufficient resolution to distinguish the pixels and dot-markers: this yields the SDI - For example, we used the HP Scanjet 2400 set at 8-bit gray-scale scanning at 300 ppi. The following Steps 2 through 5 constitute the document image reconstruction (DIR) step of Fig. 2.

2. Locate the four corner markers in the SDI - We use a zig-zag search and marker pattern matching to achieve this.

3. Auto-calibrate the pixel levels using corner markers - Statistical data from the SDI corner markers are used to compute detection thresholds, which are then used in Step 4 for dot-marker detection, and then in Step 5 for pixel decisions. Such an auto-calibration makes CIPDEC printer and scanner independent.

4. Locate the dot markers in the SDI - There are several possible ways of doing this. We describe here the method that was used in our implementation, which yielded high pixel-level accuracy for the DIR step.

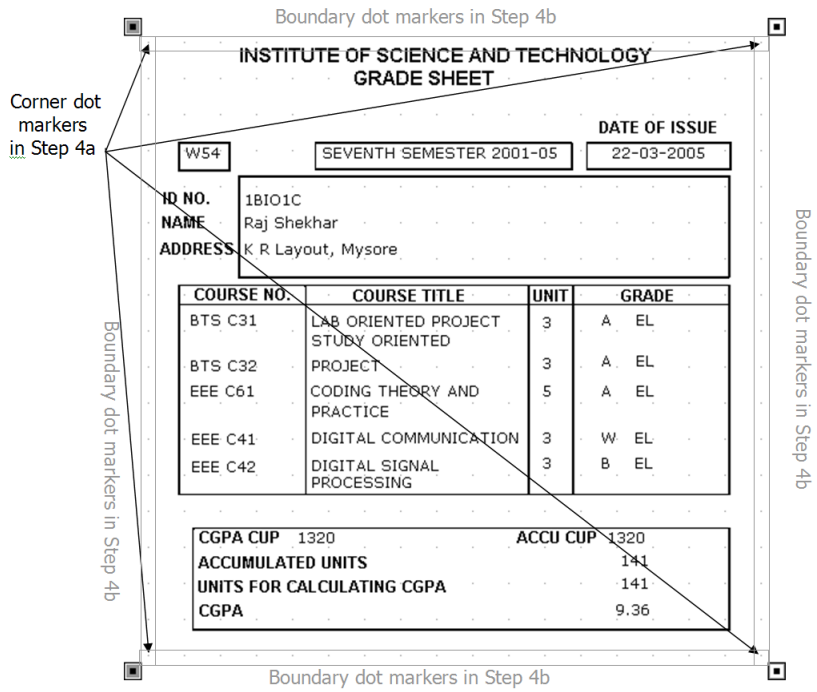

Figure 4. **I**llustration of Steps 4a and 4b in the verification stage.

(a) First locate the four dot markers nearest to the four corner markers (see Fig. 4). This is done by initially computing a coarse location from the corner marker positions (known from Step 3), and then searching a small neighborhood around the coarse location. The second step fine-tunes the coarse location and helps achieve high pixel-level precision.

(b) Next locate the other boundary dot markers (see Fig. 4) using the four detected dot markers from Step 4a. Once again, a coarse location for each dot marker is computed first, and then fine-tuned by searching its neighborhood. The coarse location is computed using a bilinear transform.

(c) Finally, locate the remaining dot markers throughout the document using the boundary dot markers detected in Steps 4a and 4b. A similar two-step (coarse and fine) procedure is adopted.

At the end of these three steps all the dot marker positions are known.

5. Reconstruct the document image by detecting the pixels within each cell - dot-marker positions from Step 4 are used for this purpose. This consists of the following two steps

   (a) Pixel location synchronization: The corner positions of each $T \times T$ cell are known by virtue of the dot marker detection in Step 4. The individual location of each of the T^2 pixels within a cell is found using a standard bilinear transform.

   (b) Pixel level determination: Each of the T^2 pixels must now be classified into one of the pixel levels - white or black. The pixel value is first computed using bilinear interpolation. Next, a hard-decision rule is applied on the pixel values to classify it as either black or white. The threshold for the hard decisions is obtained from the auto-calibration Step 3. Finally, pixel levels for the dot marker locations (which might have overwritten the document pixel) are estimated using a simple prediction algorithm that uses contextual neighborhood.

   At the end of Step 5, all the document pixels are detected, and the reconstructed document image (RDI) is obtained. This concludes the DIR step.

6. Retrieve the stored ECC parities and decode the RDI - The RDI along with the retrieved ECC parities are fed into the corresponding ECC decoder (details are in Section 5). The decoder corrects pixel errors

that are within its error correcting capability, to yield the corrected document image (CDI). Since modifications are also treated as errors by the ECC decoder, the 'difference' between CDI and RDI now reveals modifications/tampers made to the paper document.

The verification output displays the caught tampers using a color coded output (CCO). A human verifier then looks at the CCO and ascertains whether the indicated modifications are malicious or inadvertent, and accordingly makes his/her decision on the veracity of the paper record. The CCO uses: red to show deletions, blue for additions, and yellow for unchanged content. Fig. 3 shows the CCO for an example CPD (CPD is shown on the left, and the corresponding CCO is shown on the right). Three malicious modifications are caught in this example: 1) the number 3 was changed to 8, whereby the CGPA of 9.36 was modified to show 9.86; 2) the number 0 was changed to 8, whereby the CGPA CUP was increased from 1320 to 1328; 3) an entire row was deleted. In the first two cases, only a few pixels were added, and yet CIPDEC was able to catch the tamper and display it as blue (additions) to the verifier. It is also seen from Fig. 3 that there is a small fraction of residual 'salt-and-pepper' pixel noise. The human verifier can use his/her discretion in such cases to ascertain whether it is indeed a malicious modification or a random pixel error.

4. EXPERIMENTAL EVALUATION

We have extensively tested CIPDEC across different ODIs with various font sizes, generation software, ODI sizes, different printer/scanner combinations, and various paper-world artifacts like folds, stains, tears, marks, staple, annotations, manual edits, photocopying etc. In particular, we created a test set comprising of 50 documents with various types of malicious modifications and distortions. Our experiments have revealed that CIPDEC can accurately detect all pixel-level additions/deletions in over 97% of the test cases. In fact, most of the remaining 3% were constitued of extreme cases where entire chunks of content were blacked out, whereby the modifications exceeded the error correcting capability of the RS code.

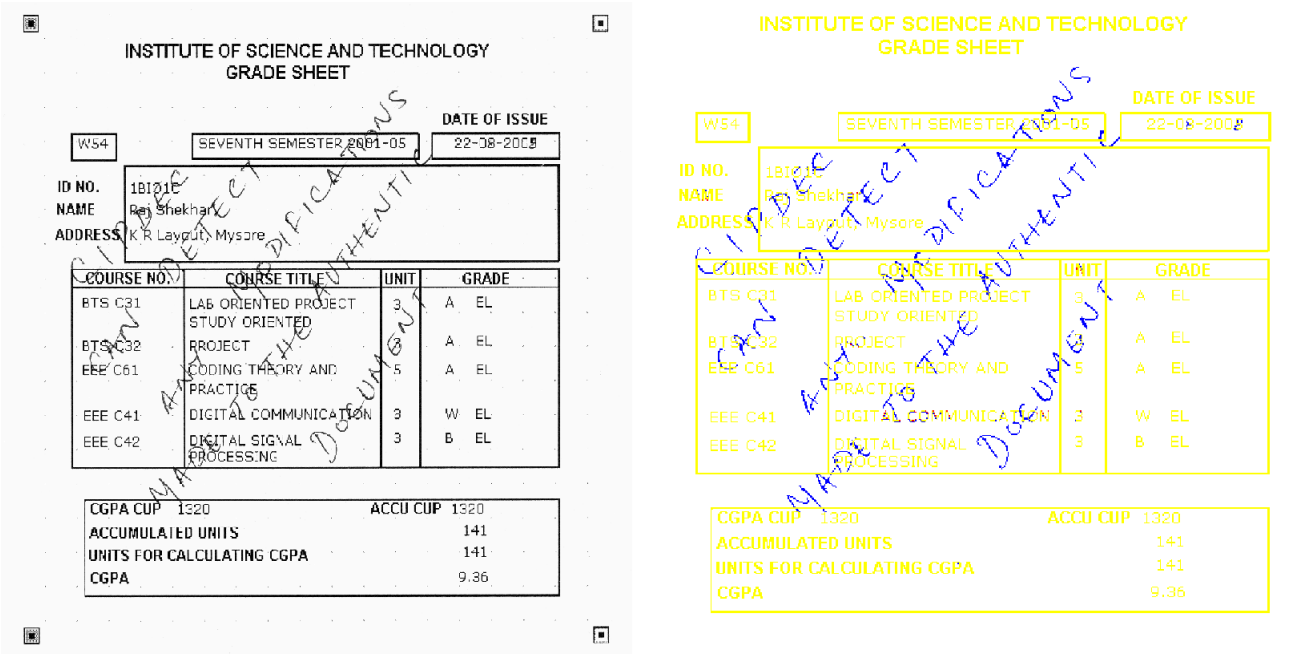

Figure 5. **C**IPDEC SDI (left) and verification CCO (right) showing the overwritings.

As illustrations, we show in Figs. 5 and 6, two test cases with the SDI on the left and the CIPDEC verification CCO on the right. Note that in Fig. 6, not only is the blackout region identified, but CIPDEC also points out the text that was blacked-out! Fig. 7 shows the CPD for three other test cases with tears, smudges, and folds, respectively. All these passed the CIPDEC verification, in spite of being heavily abused. Fig. 8 shows the

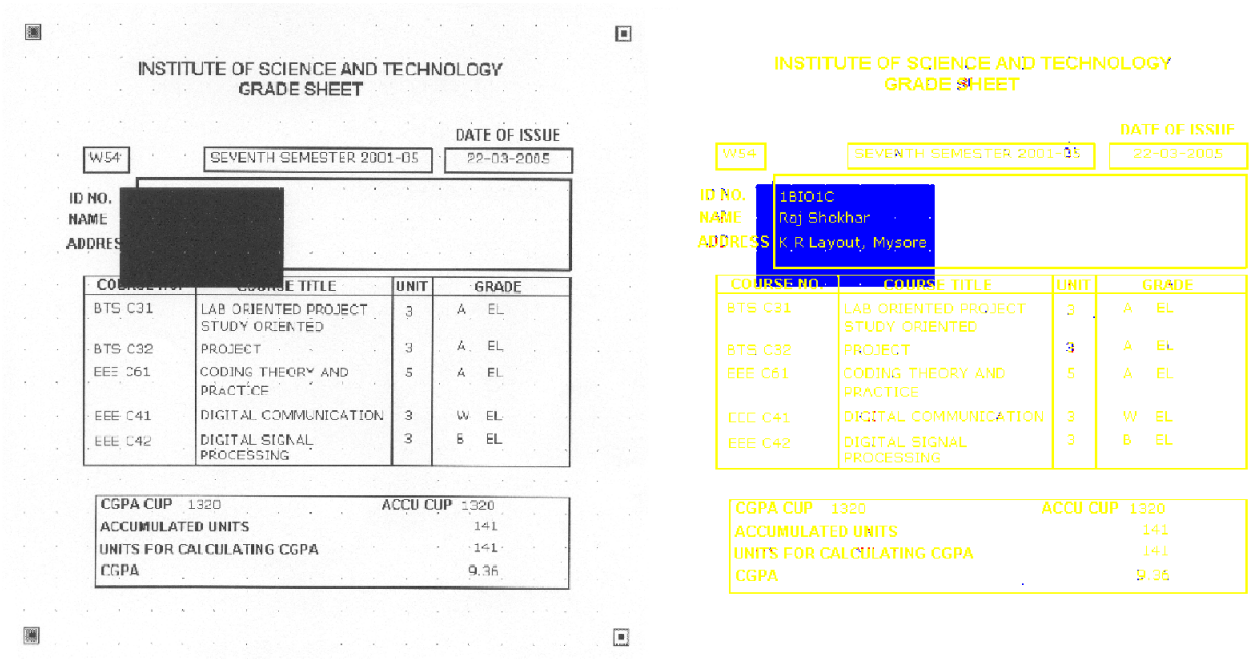

Figure 6. **C**IPDEC SDI (left) showing a blackout and verification CCO (right) showing the text recovered from blackout.

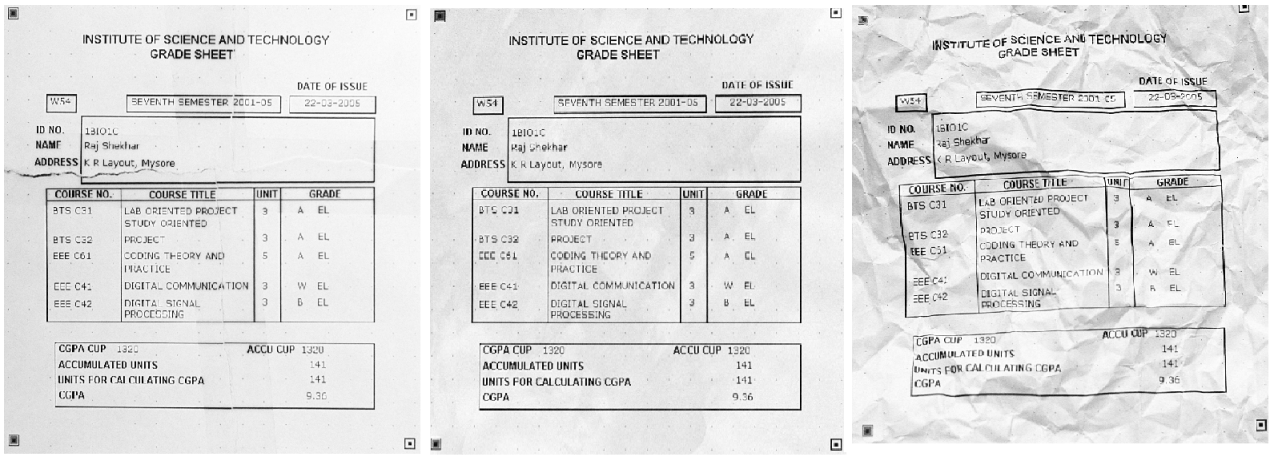

Figure 7. **C**IPDEC CPDs for three heavily distorted cases: torn and taped (left), smudged with coffee (center), and crumpled (right). CIPDEC verification was successful for all the three.

CIPDEC verification CCOs under repeated photocopying. Finally, Fig. 9 shows two test SDIs where CIPDEC could not verify their integrity. CIPDEC is unable to recover from the huge blackout in the first case, whereas the corner markers have been erased in the second case. However, these two tampers are clearly visible to the human eye, and one expects that they can easily be caught by an attentive human verifier.

Finally, we analyze the residual print/scan pixel errors left over from the DIR step (steps 2 through 5 in the verification stage), and assess whether this could potentially hamper CIPDEC verification. For this purpose, we study the raw pixel error rate (PER) in unmodified documents. A 'pixel error' is said to occur if the RDI pixel level is not the same as the corresponding ODI pixel level. The PER is defined as the fraction of pixel errors in the RDI. Thus, PER is an indicator of the residual print/scan errors after the DIR step. For illustration, the RDI for an unmodified document is shown in Fig. 10, with the residual pixel errors indicated in a dark shade. Our testing across various printer and scanner combinations yielded an average PER of 10^{-3}. We have found that this range of PER is quite tolerable, and does not interfere with CIPDEC verification. If necessary, the

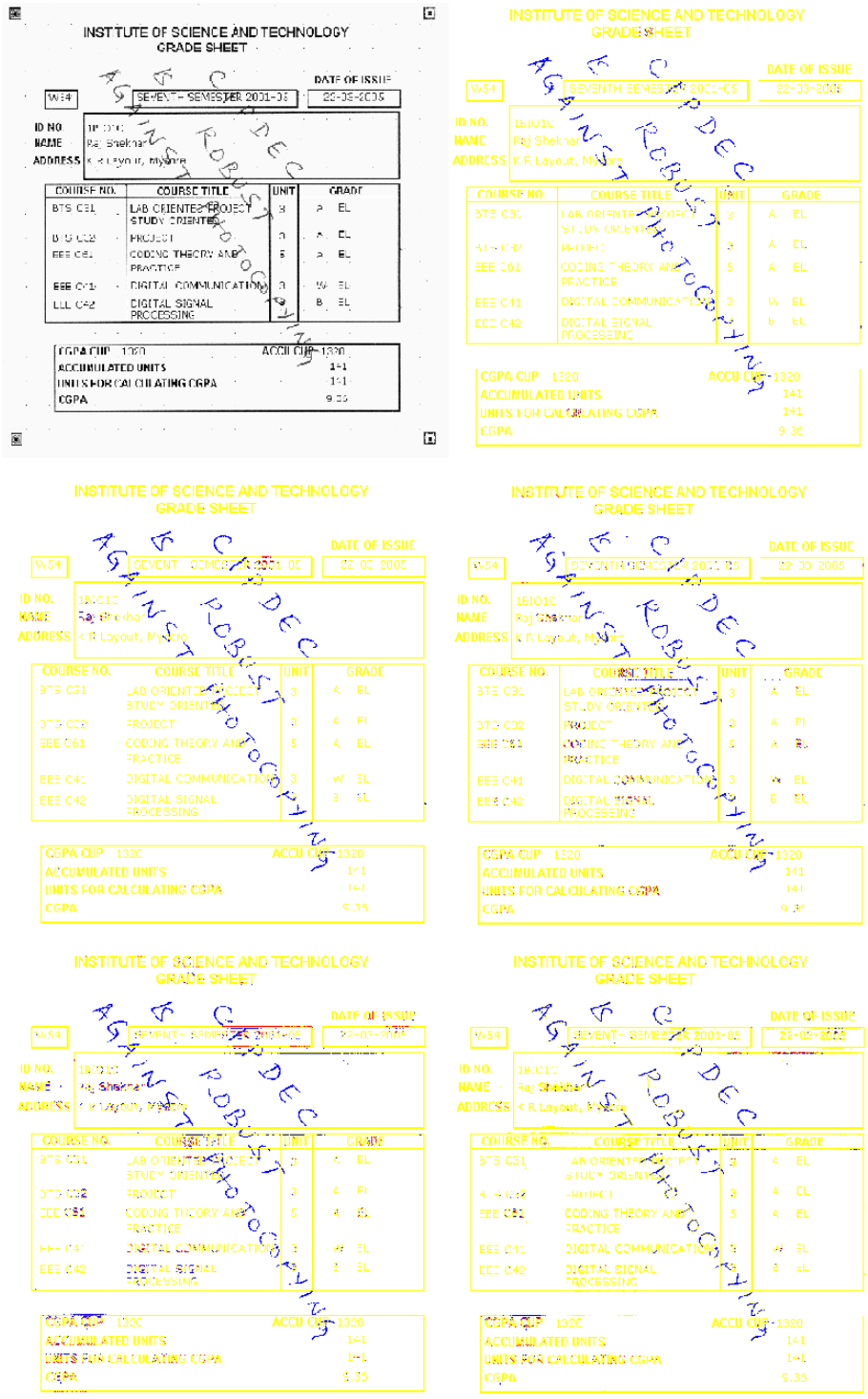

Figure 8. CIPDEC verification CCOs showing the overwritings for 1) SDI (top left); 2) no photocopying (top right); 3) first photocopy (middle left); 4) second photocopy (middle right); 5) third photocopy (bottom left); and 6) third photocopy with two scans (bottom right). The yellow (used to indicate unmodified content) in the CCO may appear as a faint shade of gray in b/w prints. Note that residual salt-and-pepper pixel errors increase with repeated photocopying. The CCO on the bottom right shows how using two scans for the same CPD can reduce these residual pixel errors.

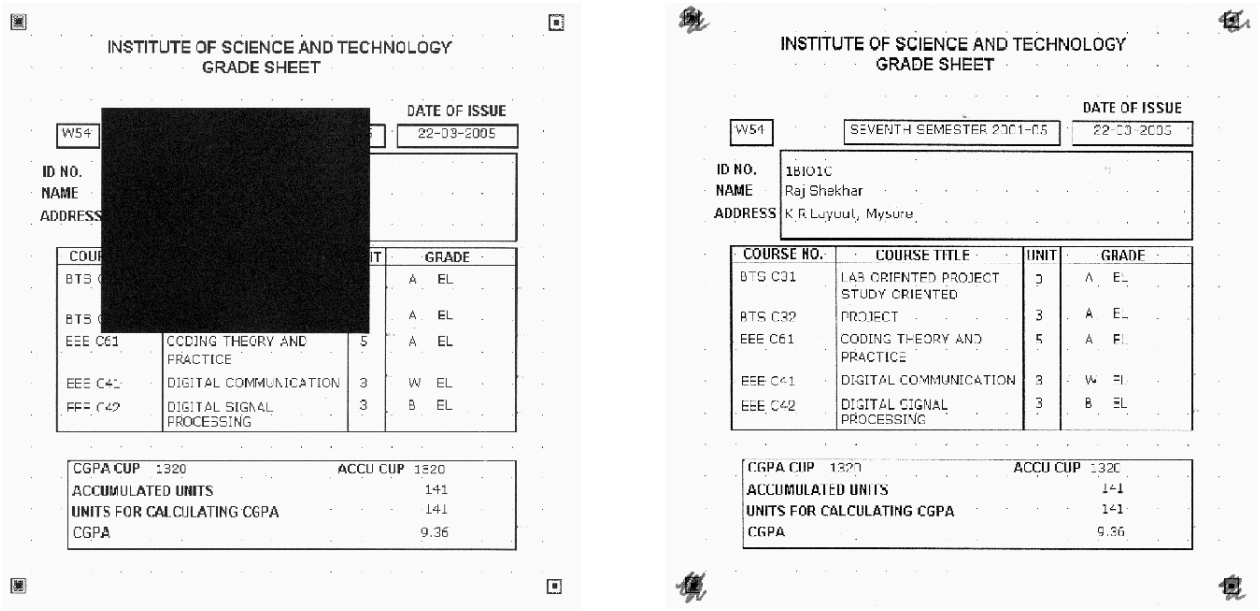

Figure 9. **C**IPDEC SDIs where verification failed: heavy blackout (left) and corner-marker tampering (right).

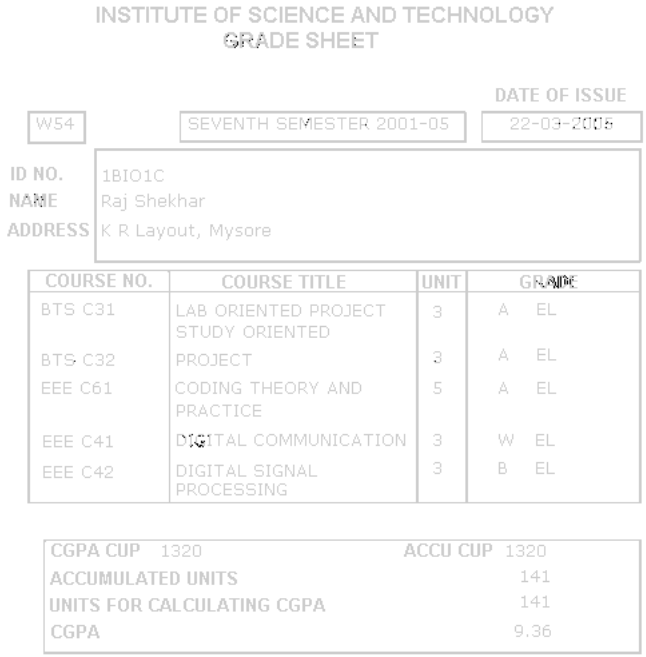

Figure 10. **R**DI with residual pixel errors shown in a dark shade.

effect of PER can be further reduced by taking multiple scans of a single CPD and combining their CIPDEC verification CCOs. One example of how using two scans for the same CPD can reduce residual pixel errors, is shown in Fig. 8 on the bottom right.

5. ERROR CORRECTING CODE (ECC) IMPLEMENTATION

The aim of using ECC in CIPDEC is to generate parities that can be used during the verification stage to identify any pixel-level changes. The implementation described here is intended to serve as an illustration, and yields

ECC parity size close to $x = 30\%$ of the ODI size. We have taken $x = 30\%$ as the conservative upper limit, which can correct even blackouts of reasonable sizes, as seen from Fig. 6. In practice, the choice of parameter x takes into account the required tamper-detection level and available storage size for the ECC parities. Lower values of x can be obtained by simply reducing the rates of the specified RS codes, while retaining the same construction as described below.[7,8]

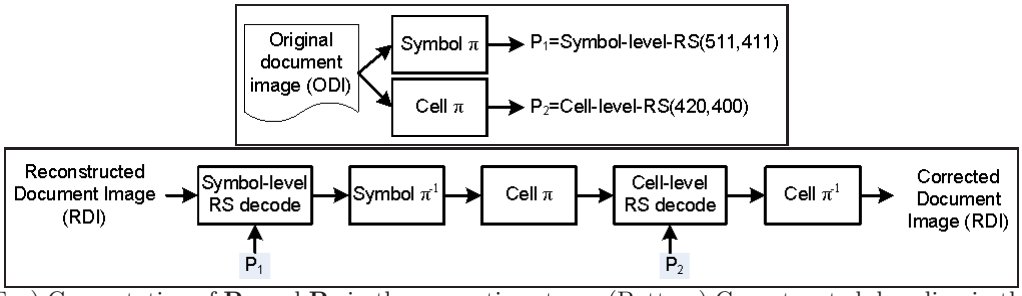

Figure 11. (Top) Computation of $\mathbf{P_1}$ and $\mathbf{P_2}$ in the generation stage. (Bottom) Concatenated decoding in the verification stage.

Let $\mathbf{P} = \{\mathbf{P_1}, \mathbf{P_2}\}$, denote the ECC parities, where $\mathbf{P_1}$ and $\mathbf{P_2}$ denote the individual parities generated by the two distinct RS codes that are used. The complete encoding block diagram is shown in Fig. 11. For computing $\mathbf{P_1}$, the ODI is divided into square blocks 3×3 pixels. Since each pixel is binary (either white (1) or black (0)), each 3×3 block can be viewed as a symbol over the finite-field GF(512). The symbols are then spatially interleaved ('Symbol π'), and fed into a systematic $(511, 411)$ Reed-Solomon (RS) encoder.[7,8] The entire array of output parities form $\mathbf{P_1}$. For computing $\mathbf{P_2}$, the ODI is divided into square blocks of 30×30 pixels, which for our implementation is chosen to precisely match the cell size (see Section 3). The parity computation for $\mathbf{P_2}$ proceeds in a similar fashion, but by using a $(420, 400)$ RS code with a cell-level interleaver ('Cell π'). The two interleavers, 'Symbol π' and 'Cell π' play a critical role in identifying spatially contiguous modifications. They effectively disperse a burst of modified symbols/cells so as to enable decoding in the verification stage. This is a practical requirement since most meaningful modifications consist of addition/deletion of contiguous symbols.

To sum up, we use two distinct RS codes - the $(511, 411)$ RS code at the symbol level, and the $(420, 400)$ RS code at the cell-level - which carry a total redundancy of 29.33%. $\mathbf{P_1}$ is designed to identify most of the modifications, and hence holds a large chunk of the redundancy - 24.33%, while $\mathbf{P_2}$ is used mainly to ensure a meaningful visual display of the verification output (CCO display, see Section 3). The reason is that any decoding failures are more meaningful to a human as large-enough cells rather than tiny symbols.

During the verification stage, the RDI is decoded using the concatenated structure shown in Fig. 11. First, the RDI is decoded using $\mathbf{P_1}$ as the parities. This is shown as 'Symbol-level RS decode', where the decoding is performed using the Berlekamp-Massey (BM) algorithm[7,8] on the $(511, 411)$ systematic RS code. Next, the output of this first RS decoder is de-interleaved using the inverse of the 'Symbol π' interleaver, and then interleaved using the 'Cell π' interleaver. Subsequently, the 'Cell π' interleaver output is fed into the cell-level RS decoder which uses $\mathbf{P_2}$ as the parities. This time the BM algorithm for the $(420, 400)$ systematic RS code is used for decoding. Finally cell-level de-interleaving is performed to yield the corrected document image (CDI).

6. OTHER POTENTIAL APPLICAITONS

While we have thus far focused on 'content integrity' using CIPDEC, the main idea of detecting pixel-level modifications has a wider range of applications. Fig. 12 shows an illustration where CIPDEC was used to extract the handwritten ink filled into a form. The handwritten ink extracted by CIPDEC is shown in dark, while the rest of the form content is shown in a light shade. The principle of CIPDEC is the same here - namely detecting modifications using ECC parities - but the so-called modifications in this case actually correspond to the handwritten ink. The extracted ink can then be passed on to a suitable recognition engine, which recognizes the meaningful data and passes it on to a further application. A similar setup can be used to capture manual edits made to printed documents, and several such other intelligent extraction and processing of printed documents.

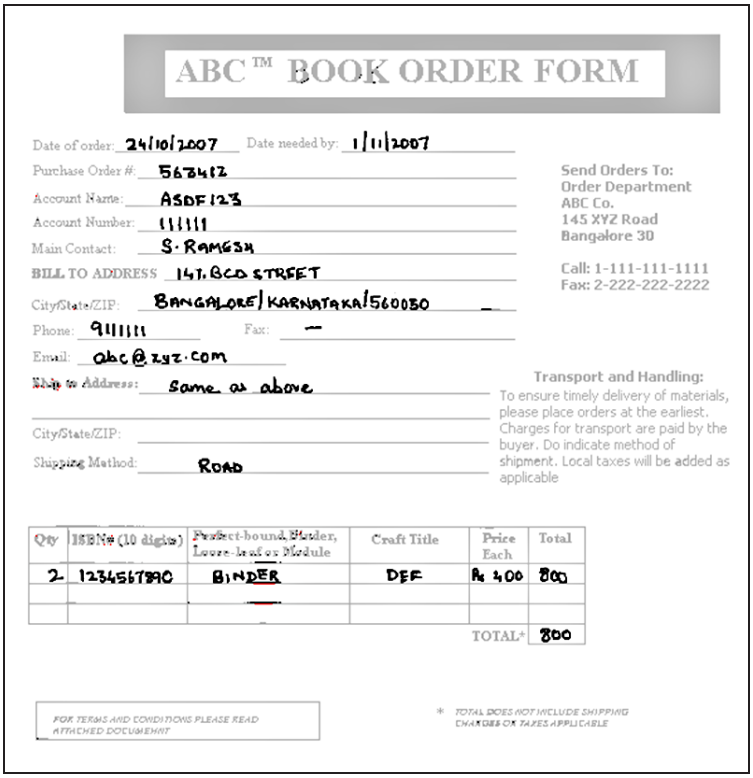

Figure 12. **A**nnotation extraction and form-filling example.

7. CONCLUSION

We presented a novel technique, called CIPDEC, for detecting modifications on paper documents. The core component of CIPDEC is the error correcting code (ECC), which helps detect any forgery and fraud of even a few pixels. Such a pixel-level, ECC-based approach provides several advantages including blind verification, font/language/software independence, and backward-compatibility with legacy documents. Extensive testing has revealed that CIPDEC is robust to photocopying, folds, tears, stains, etc. and also works across different printers and scanners by virtue of its auto-calibration. On the other hand, two requirements of CIPDEC are that the printed document must carry a barcode with the ECC parities, and that the document must be marked with corner and dot markers for a pixel-precision fidelity. For the first requirement, the size of the ECC parities is only a fraction of the document size (as opposed to the entire document size in most 'non-blind' solutions), and this size is also tunable according to the application requirements.

REFERENCES

[1] A. Sellen and R. Harper, "The Myth of the Paperless Office", *MIT Press*, 2001.

[2] J. Fridrich and M. Goljan, "Robust hash functions for digital watermarking", *in Proc. International Conference on Information Technology: Coding and Computing*, Mar. 2000.

[3] M. Jiang, E.K. Wong and N. Memon, "Robust document image authentication", *in Proc. IEEE International Conference on Multimedia and Expo*, July 2007.

[4] V. Monga, D. Vats and B.L. Evans, "Image authentication under geometric attacks via structure matching", *in Proc. IEEE International Conference on Multimedia and Expo*, July 2005.

[5] S. Huang and J.K. Wu, "Optical watermarking for printed document authentication", *IEEE Trans. Inform. Forensics and Security*, vol. 2, no. 2, June 2007.

[6] W. Yan, D. Jin and M.S. Kankanhalli, "Visual cryptography for print and scan applications", *in Proc. IEEE ISCAS*, 2004.

[7] *Error Control Systems for Digital Communication and Storage*, S.B. Wicker, Prentice-Hall Inc., 1995.

[8] *Error Control Coding: Fundamentals and Applications*, S. Lin and D.J. Costello, Prentice-Hall.
[9] I.S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Indiustrial Appl. Math.*, vol. 8, pp. 300-304, 1960.