SAKSHAMJAIN
21BDS0239

Tamper Detection in Academic Credentials

1. Introduction

In an increasingly digital academic ecosystem, verifying the authenticity of official documents is crucial to combat fraudulent degrees and certifications. This project presents a prototype system for **detecting tampering in academic documents** (degrees, transcripts, certificates) using a combination of **PDF metadata inspection**, **layout similarity comparison**, and **OCR-based text extraction**.

The system supports the detection of unauthorised modifications and provides alerts when discrepancies are found between original and tampered versions.

2. Methodology

2.1 Document Preparation

We manually created mock datasets with both original and tampered PDF versions of:

- **Degree Certificate**
- **Academic Transcript**
- **Course Completion Certificate**

Tampering operations included:

- Modifying names, GPA, course titles, and dates.
- Changing layout (e.g., fake seals, institution names).
- Altering metadata (modification dates or software used).

2.2 Metadata Analysis

Tool Used: `PyPDF2`
We extracted PDF metadata and compared fields like:

- `/CreationDate`
- `/ModDate`
- `/Producer`

Heuristic: If `/ModDate` > `/CreationDate`, the document is flagged as potentially modified. Additionally, suspicious `/Producer` values (e.g., "Fake PDF Editor") may indicate tampering.

2.3 Layout Comparison via SSIM

Tools Used: `pdf2image`, `OpenCV`, `skimage.metrics`
Steps:

1. Convert both original and tampered PDFs to PNG images (first page).
2. Convert images to grayscale.
3. Use **Structural Similarity Index (SSIM)** to compare the visual layout.

Threshold:
SSIM score < 0.95 implies layout inconsistencies. Tampered documents (e.g., fake seals, repositioned logos) will often show a noticeable drop in SSIM.

2.4 OCR-Based Text Extraction

Tools Used: `pytesseract`, `pdf2image`
Steps:

1. Convert PDF pages to images.
2. Extract text using Tesseract OCR.
3. Compare extracted text (original vs tampered) using manual inspection or Python's `difflib`.

Focus Points:

- Changed names (e.g., "Alex Carter" → "Alex Smith")
- Added or removed courses
- Altered grades or GPA
- Modified dates

This step detects subtle tampering even when the layout remains unchanged.

3. Assumptions

- Original and tampered files are named in a predictable format (`*_original.pdf`, `*_tampered.pdf`)
- Files are in English and contain standard fonts readable by OCR
- Tampering always reflects in one or more: metadata, layout, or content

- The documents are single-page or have most relevant data on the first page

4. Challenges & Trade-offs

Challenge	Description	Workaround
OCR Inaccuracy	Noise or poor scan quality may reduce accuracy	Used high DPI (300) image conversion
SSIM Limitations	Small changes may not affect SSIM drastically	Consider pixel-level diff or perceptual hashing
Metadata Falsification	Easy to spoof using editors	Combine with other checks (e.g., digital signatures)
Manual Text Comparison	Automated diff is not perfect with OCR errors	Tolerate small mismatches or apply fuzzy matching

5. Suggestions for Improvement

1. Advanced Text Comparison
Use NLP techniques (e.g., sentence embeddings) to compare document semantics and detect paraphrased content.

2. Batch Automation
Extend the tool to handle folder-based comparison and generate CSV reports summarizing tampering results.

3. Web Dashboard
Create a simple Flask/Django interface for users to upload documents and view comparison results visually.

4. Digital Signature Verification
Incorporate X.509 digital signature checks for documents signed by trusted issuers.

5. ML-Based Tamper Detection
Train a supervised ML model to classify tampered vs. genuine documents based on features from layout, metadata, and text.

6. Conclusion

The developed system demonstrates an effective approach for detecting tampering in academic documents using open-source tools. It covers multiple layers of inspection—metadata, layout, and content—which complement each other. While the prototype works well in a controlled environment, real-world deployment will require handling diverse file types, noisy scans, and adversarial attempts.

With improvements in automation, UI, and tamper classification, this solution can serve as a valuable tool for academic verification offices and recruitment agencies.