## **Technical Report: Tamper Detection in Academic Credentials Prototype**

**Internship Assessment Submission**

**Date:** May 19, 2025

### **1. My Approach, Assumptions, and Methodology**

My goal for this prototype was to explore how we can automatically spot potential tampering in digital academic documents like certificates or transcripts. Since I needed a lightweight start, I focused on checking the hidden information inside the PDF file itself, called metadata, and doing a quick look at the text on the first page.

My approach was to build a simple Python script using the pdfplumber library. This library is good because it can easily read the metadata and pull out the text from a PDF, even telling you where the text is on the page (though I mostly used it for just getting the text content in this basic version).

**Methodology:**

The script works by taking a PDF file path and doing two main things:

1. **Checking the PDF's Metadata:** Every PDF has details embedded in it, like when it was first made (CreationDate), when it was last changed (ModDate), and which software created it (Creator, Producer). My script reads these specific pieces of information. I check if the 'Created' and 'Modified' dates are different. If they are, it could mean the file was edited after it was originally created. I also look at the 'Creator' and 'Producer' fields to see if they mention software that's usually used for editing images, like 'Photoshop'. If I see something like that, it's a bit suspicious because a real, official document is usually made with document software or a specific university system.
2. **Checking the First Page's Text:** I also pull out the text from the very first page of the PDF. I then check if this text contains some basic words you'd expect to see in an academic document, like "university", "degree", "transcript", or "certificate". If these key words are missing, it might suggest the document isn't what it claims to be, or maybe the text couldn't be read properly.

Based on these checks, the script collects a list of potential "flags" if it finds anything suspicious, like a date mismatch or missing keywords.

**Assumptions:**

For this basic prototype, I made a few assumptions:

* I assumed that the PDF documents would actually *have* some metadata fields like CreationDate, ModDate, Creator, and Producer. Sometimes these can be removed.
* I assumed that if a document is tampered with, the metadata might show signs of it (like a changed date or suspicious software used for editing).
* I assumed that important keywords like "university" or "degree" would be present and readable on the first page of a legitimate academic document.
* I assumed that the first page is relevant and representative of the document's type.

### **2. Challenges and Trade-offs**

Building even this simple prototype highlighted some challenges and required making trade-offs:

* **Metadata Isn't Always Reliable:** The biggest challenge with relying on metadata is that it can often be easily changed, removed, or even faked using simple tools. A clever fraudster might make the 'Created' and 'Modified' dates the same, or change the 'Creator' field to look like legitimate software.
* **Date Formats are Tricky:** The date format in PDF metadata (D:YYYYMMDDHHMMSS±HH'MM') is not standard and needs careful parsing for accurate comparisons. My prototype does a very basic check; a real system would need robust date parsing.
* **Text Extraction Limitations:** Pulling text from PDFs using libraries like pdfplumber works best on digitally created PDFs. If a document is a scanned image saved as a PDF, the text might not be directly extractable, or the extraction might have errors. This would cause the keyword check to fail even on a real document.
* **Basic Checks Miss Lots of Tampering:** My prototype only looks at a few specific things. It doesn't check the visual appearance at all – it can't tell if a seal is forged, if text has been moved or changed visually, or if logos are fake. It also doesn't understand the actual *structure* of a certificate or transcript (like where the name should be, or how grades are listed).
* **Trade-off: Lightweight vs. Comprehensive:** I chose to keep the script lightweight and focus on easily accessible information (metadata, basic text). The trade-off is that it can only catch very simple or specific types of tampering and would likely miss more sophisticated forgery.

### **3. Suggestions to Improve or Scale the System**

To make this tamper detection system more effective and scalable, I would suggest several improvements:

* **Add Visual Analysis:** This is crucial. Use libraries like OpenCV to convert PDF pages to images and then analyze the image. This could involve comparing the document's layout to known templates, detecting the presence and correct position of university logos, seals, and signatures, and analyzing areas around these features for signs of image editing or pasting.
* **Implement Robust Text and Structure Analysis:** Go beyond just checking for keywords. Use libraries that can extract text with precise position information (pdfplumber can do this with its layout functions, or PyMuPDF). Define expected locations for key information (student name, degree, date, grades) for different document types and check if the text is found in those areas. For transcripts, specifically analyze tables to verify the format of course lists and grades.
* **Integrate OCR for Image-Based Text:** Use OCR (like Tesseract) to read text from image-based parts of the PDF (like scanned signatures or seals with text) or from scanned documents. Compare this OCR text with the text extracted from the PDF's text layer to spot discrepancies that might indicate hidden or altered text.
* **Develop More Advanced Anomaly Detection:** Use statistical methods or machine learning. Train a model on a large dataset of *legitimate* academic documents to understand what "normal" looks like (e.g., typical file sizes, metadata patterns, text densities, layout structures). Then, use the model to flag documents that are statistically unusual or deviate significantly from the norm.
* **Use Templates and Rules:** Create a system for defining templates or rules for specific institutions or document types. This would allow the system to perform more targeted checks (e.g., "For a University X Transcript, the logo should be in the top left corner, and the grades table should have exactly 4 columns").
* **Explore Digital Signatures and Blockchain:** For future scalability and reliability, investigate how digital signatures embedded in PDFs or hashes of legitimate documents stored on a blockchain could be used for verification. This provides a strong layer of trust outside the document content itself.
* **Build a Verification Database:** Ideally, integrate with a database from issuing institutions to verify key details like student ID, degree awarded, and date of issue directly.

Implementing these suggestions would move from a basic metadata check to a multi-layered system capable of detecting a much wider range of sophisticated tampering techniques. This would require more development effort, computational resources, and potentially access to datasets of legitimate documents and templates.