Secure Notarization of Paper Text Documents

Matthias Ruhl* Marshall Bern† David Goldberg†

Abstract

We present a method to notarize a paper document. This method computes a set of features from the page, inputs these features into a hash function, digitally signs the output, and then prints the resulting signature on the page. The notarized document is self-contained: verification does not require reference to online information.

Introduction. Authentication of digital data is a well-studied problem in computer science [8], but many important, long-lived documents are still available only on paper. Paper is traditionally authenticated either through elaborate printing techniques (as in money) or through trusted signatures and stamps (as in wills or contracts). Current technology, however, has made it easier to counterfeit paper documents. In this work, we show how to use digital methods to authenticate paper documents, and introduce concepts that could be used for other cases of authenticating analog data.

We formalize the problem as follows. Given an analog document D, the *signer* computes a digital signature $s(D)$. Given a document/signature pair (D,s), the *verifier* checks whether s is a valid signature $s(D)$ for document D. The document signature should have the following security properties: The verifier accepts a valid (D,s) pair with high probability; the verifier rejects a pair $(D',s(D))$ with high probability if D and D' differ "significantly"; and only the signer can compute signatures that are accepted by the verifier. Moreover, the signature $s(D)$ should be small, so that it can be printed onto the document in a machine-readable form such as bar codes or "data glyphs" [3], thus avoiding the need to keep information online in perpetuity.

The desired properties pose a difficult problem, and it was initially unclear whether a practical solution was even possible. The solution we propose here uses *features* computed from the document, essentially the locations and shapes of repeating components (e.g. letters) on the page. These features should suffice for any document composed of repeating components, even if the components cannot be recognized as known symbols. Although these features are fairly robust to scanner variation, we needed another key idea to make our method work: an *assist channel*, essentially error correction bits for analog-to-digital conversion.

Most previous methods [1, 4] authenticate a document by comparing its features to a known valid copy; unlike our method they require an online copy. One exception is the work of O'Gorman and Rabinovich [6], who present a method of authenticating small photos. Their technique, however, writes out the features themselves rather than a hash of the features, and hence does not scale well to large images or text pages. Adding hashing is not just an incremental change. Hashing forces the feature computation to be entirely reproducible, because a change in even one bit will change the hash output.

Algorithm. Consider the following method, which is a straightforward transfer of digital technology to paper documents: the trusted signer scans the document, passes the bits through a one-way hash function, signs the hash using public key cryptography, and finally affixes the signature to the document in machine-readable form. Unfortunately this simple scheme does not work. When the verifier scans the document, he will not get exactly the same bits as the signer and hence does not obtain the same hash. The verifier needs an assist channel: a string of bits (also signed and printed on the page in machine-readable form) that enables him to recreate the same data each time. The idea of an assist channel is implicit in [5] and is currently being explored in detail by Greene [2].

We now give an example of a very simple type of assist channel. Imagine a feature that is a continuous function of the input page and takes real values in the range $[0,1]$, and assume that scanner variation adds a small amount of noise to the feature. One might discretize the feature value to two bits to represent the ranges $[0,\frac{1}{4})$, $[\frac{1}{4},\frac{2}{4})$, $[\frac{2}{4},\frac{3}{4})$, and $[\frac{3}{4},1]$, and use a one-bit assist channel. An assist bit of 0 means that discretization should be done by directly comparing the measured feature value to the ranges, and a 1 means that $\frac{1}{8}$ should be added to the feature value before this comparison in order to move it away from $\frac{1}{4}$, $\frac{2}{4}$, or $\frac{3}{4}$. So long as the noise is never as large as $\frac{1}{16}$, this single-bit *rounding hint* enables robust recomputation of the discretized feature.

A 300 dpi binary scan of a text page is roughly 1 MByte and varies significantly from scan to scan, so it is not possible for a small assist channel to remove all variation. Hence we are led to the problem of extracting document

*MIT Laboratory for Computer Science, Cambridge, MA 02139, USA, ruhl@theory.lcs.mit.edu

†Xerox PARC, Palo Alto, CA 94304, USA, {bern, goldberg}@parc.xerox.com

features that can distinguish a valid scan from a subtle forgery, yet are not as variable as the bitstream itself. At the same time, we must devise an assist channel that ensures feature robustness from scan to scan.

We now sketch what we came up with. We describe the signer's and verifier's algorithms in parallel, although in practice verification would take place after signing.

First, both signer and verifier scan the image at 300 dpi in gray-scale mode. Then the verifier gives his image an initial coarse registration using registration marks pre-printed in the corners of the document. The verifier also performs histogram equalization using a gray-level histogram of the signer's image read in from the assist channel.

Second, both signer and verifier compute connected components (cc's for short), meaning maximal sets of adjacent sufficiently dark pixels. In order to obtain agreement between signer and verifier cc's, the signer uses two different darkness thresholds, one that tends to merge and one that tends to split ambiguous cc's. The signer puts into the assist channel the bounding box of each merged cc that could possibly split up when scanned by the verifier. The verifier then knows to merge cc's that fall within each of these boxes.

Third, signer and verifier compute the xy-coordinates of cc's. Even after initial registration, pixels in the middle of the page may still be misregistered. Luckily the verifier can use the bounding boxes from the last step to correct this problem. All of the verifier's cc's that are not in assist-channel bounding boxes are moved by a weighted sum of the translations observed at nearby boxes.

The coordinates of centroids of cc's are rounded to the nearest multiple of 16 pixels. The signer puts a list of all coordinates that are close to a rounding boundary into the assist channel, together with a bit indicating the direction it should be rounded. Using these hints, the verifier can reliably obtain exactly the same rounded coordinates.

Fourth, the signer and verifier must agree on an ordering for cc's. Subscripts, formulas, broken characters, etc. make it difficult to find a robust ordering rule. Our method orders cc's in "chunks" (roughly text lines), approximately in English reading order both within and between chunks. Coordinates of initial cc's in chunks and cc's with ambiguous order within a chunk are put into the assist channel.

The fifth and final step is to compute shape features that characterize the gray-scale cc's. In order to keep down the size of the assist channel, cc's are combined into clusters using a tokenizing compressor [7], and shape features are computed for only the first member of each cluster. We include the cluster number for each component in the assist channel. (Clustering also enables further security checks: the verifier could check that a cluster's members are indeed very similar to each other.) The shape features themselves are computed by convolving cc's with a number of *test functions*, finding the maximum value in each of these convolutions, and rounding these values to a small number of bits. If an original value is close to a rounding threshold, an appropriate rounding hint must be added to the assist channel. In our experiments, we used 24 test functions including horizontal, vertical and diagonal lines, T- and L-shapes, and dots and rings.

Document	Chemistry	Physics	Software
Histogram	261	226	204
Bounding Boxes	2952	2020	1346
Locations	856	540	395
Ordering	2593	1924	1299
Shape Features	4956	4728	4327
Total	11,618	9438	7571

Results. The method worked successfully on three different makes of scanners and on various documents, including complicated physics and chemistry journal pages. The security appeared to be quite good; even tiny changes to the page gave different hash values.

The table shows the size of the assist channel in bytes. The hash itself is negligible—only 128 bits. Cryptographically signing the data (hash and assist channel) does not expand the size appreciably, so the table gives a fair indication of the overall size of the signature. Even though our encoding is rather crude, the signature is slightly smaller than token-compressed scans of the pages. To put the sizes in perspective, with a 600 dpi printer it is possible to write dense but readable glyphs at about 0.8 KByte per square inch, giving a signature of 8–12 square inches. We are currently investigating ways to reduce the size of the assist channel without compromising security. An open theoretical problem is to determine lower bounds on the size of an assist channel for analog-to-digital conversion.

References

[1] S. Bhattacharjee and M. Kutter. Compression tolerant image authentication. *Proceedings ICIP '98, Chicago*, 1998.

[2] D.H. Greene. Assist channel coding and processing. *Manuscript*, 1999.

[3] D. Hecht. Embedded data glyph technology for hardcopy digital documents. *SPIE Proceeedings*, 2171:341–352, 1994.

[4] C.-Y. Lin and S.-F. Chang. Generating robust digital signature for image/video authentication. *Multimedia and Security Workshop at ACM Multimedia '98*, September 1998.

[5] D.P. Lopresti and J.S. Sandberg. Certifiable optical character recognition. *Proceedings 2nd ICDAR*, 1993.

[6] L. O'Gorman and I. Rabinovich. Identification of documents via pattern recognition and public-key cryptography. *IEEE Trans. PAMI*, 20(10):1097–1102, 1998.

[7] W. Rucklidge and D. Huttenlocher. A flexible network document imaging architecture. *SPIE Elect. Imaging*, 2000.

[8] B. Schneier. *Applied Cryptography*. Wiley, 1995.