SealClub: Computer-aided Paper Document Authentication

Martin Ochoa
ochen@zhaw.ch
Zurich University of Applied Sciences
Zurich, Switzerland

Jorge Toro-Pozo
jorge.toro@sdx.com
SIX Digital Exchange
Zurich, Switzerland

Hernán Vanegas
hdvanegasm@unal.edu.co
Universidad Nacional de Colombia
Medellín, Colombia

David Basin
basin@inf.ethz.ch
ETH Zurich
Zurich, Switzerland

ABSTRACT

Paper documents, where digital signatures are not directly applicable, are still widely utilized due to usability and legal reasons. We propose a novel approach to authenticating paper documents by taking short videos of them with smartphones. Our solution combines cryptographic and image comparison techniques to detect and highlight semantic-changing attacks on rich documents, containing text and graphics. We provide geometrical arguments for the security of our novel comparison algorithm, and prove that its combination with a cryptographic protocol is secure against strong adversaries capable of compromising different system components. We also measure its accuracy on a set of 128 videos of paper documents and a set of 960 synthetically generated warped documents, half containing subtle forgeries. Our algorithm finds all forgeries accurately with no false positives. The highlighted regions are large enough to be visible to users, but small enough to precisely locate forgeries.

KEYWORDS

Document authentication, Forgery detection

1 INTRODUCTION

Despite advances in digitalization, printed paper is still widely used for documents of various degrees of sensitivity. In the US alone, 129 billion physical mails were sent in 2020 [53]. In enterprise environments, departments like human resources, legal, and accounting are still predominantly paper-centric [25]. Moreover, research on the impact of media on reading outcomes consistently shows that reading comprehension and efficiency improve when reading from paper compared to reading from screens [13, 15, 39].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

ACM Reference Format:

Martin Ochoa, Hernán Vanegas, Jorge Toro-Pozo, and David Basin. 2023. SealClub: Computer-aided Paper Document Authentication. In Annual Computer Security Applications Conference (ACSAC ’23), December 04–08, 2023, Austin, TX, USA. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3627106.3627176

The vision of fully digital offices and services has been articulated since the 1970s [8], with predictions such as “by 1990, most record-handling will be electronic.” Countries like Denmark, who have pursued the digitalization of public services for decades, have realized that, even in 2022, about a quarter of the adult population is “digitally exposed” [26] and rather than being forced to use digital services, they should be given easy access to analogue alternatives. It thus seems unlikely that paper will be completely replaced by electronic communication in the foreseeable future.

As a consequence, the forgery of printed documents is, and will continue to be, a widespread problem [18]. For example, it is believed that a single degree mill company sold over 200,000 fake educational diplomas worldwide for $51 million in 2015 alone [5]. Germany’s Federal Criminal Police Office has detected about 70,000 document forgeries per year over the last decade [20].

This situation raises a natural question: can we leverage standard digital security mechanisms, such as cryptographic signatures, to prevent attacks on paper documents? This question has inspired various proposals [2, 17, 31, 56] such as adding QR codes to documents containing information like signed hashes of the document’s content. This annotation allows interested parties to verify a printed document’s authenticity, assuming they possess the signer’s public key. It is now also common to see documents with QR codes or URLs pointing to an authentic version of the document on the issuing institution’s website [23, 24, 52]. These approaches and other related work are discussed in more detail in Section 6.

Such techniques have strong limitations. A human verifying a document’s authenticity must still manually compare a printed document against an online version or against text stored in a QR code and displayed on a PC or smartphone. While this can help humans spot obvious forgeries, more subtle attacks such as removing a few characters are challenging to spot manually. It is well known, for instance, that a single comma can alter the meaning of a sentence [16]. Consider the sentence “Stop clubbing baby seals” versus “Stop clubbing, baby seals”.1 In a recent legal case in the US, the lack of a comma on a labor law regulating paid over-time was used to rule in favor of a group of drivers who received a $5 million settlement for unpaid work [34].

Intuitively, advances in Object Character Recognition (OCR) and machine learning should be directly applicable in this context to spot such subtle semantic attacks. However, as we will see in Section 2.5, such approaches also have inherent limitations. On the one hand, OCR cannot spot differences in documents containing rich

1Also a meme of internet fame.
text, such as simple graphics and handwritten signatures, causing both false positives and false negatives in such scenarios. On the other hand, machine learning requires large data sets for training, it is hard to reason about their generalizability to previously unseen document types, and the results lack explainability.

**Approach.** We propose a system, called SealClub, to digitally authenticate printed documents. SealClub is based on the following two components: (i) cryptographic mechanisms are used to guarantee the authenticity and privacy of reference document pictures and (ii) an image comparison algorithm is used to detect subtle forgeries on a short user-taken video of a paper document with respect to an authentic reference document. We use a video instead of a single picture for both usability and accuracy reasons: superimposing the outcome of the analysis on each video frame (augmented reality) provides users with immediate feedback and comparing multiple images against the reference image improves accuracy and helps filter out false forgery alarms.

![Figure 1: SealClub’s usage on an example scenario.](image)

Our image comparison algorithm is based on approximate similarity that automatically highlights noticeable differences between two document images. Our algorithm marks regions that differ by more than a pre-configured threshold and displays the differences to end users in a meaningful way, enabling them to make informed decisions about a document’s authenticity. Crucially, our algorithm works in realistic conditions using off-the-shelf smartphone cameras and is tolerant to routine distortions, for example arising from document folding or adverse lighting conditions, thereby reducing false positive rates while maintaining high forgery detection rates.

**Example.** Figure 1 illustrates an example of how this works and comes from an ongoing collaboration with the administration of a major city. Consider a scenario where a government authority issues paper documents to citizens. The issuer has a public/private key pair. With the private key, the issuer signs a URL pointing to a digital copy of the document to be authenticated together with a hash of the image’s content. The printed version of the document contains a QR code with this signature, the URL, and the document’s hash. Attackers may be motivated to forge such documents when showing them to a third party. For instance, they may want to modify the content of a debt collector registry when applying for a loan or rental. Third parties who wish to verify the authenticity of such printed documents install an authentic app on their phone, which has pinned the issuer’s public key. With this key, the app can verify the signature in the QR and download the digital copy and check its authenticity. The app runs our image comparison algorithm, which compares the authentic digital copy of the document against a short video of the physical document. Afterwards, the app either declares the physical document as authentic or highlights those regions that have been potentially tampered with.

We have designed SealClub to be secure even against strong attackers. As described in Section 3, it uses multiple keypairs and other security measures to be secure even against attackers who can compromise individual components or the components' private keys. Our approach provides message-origin authentication and, as we will see in Section 4, timestamp-based non-repudiation of origin under key revocation.

We also provide privacy guarantees by employing symmetric cryptography to protect against data breaches on the cloud storage, which could expose sensitive documents. Namely, SealClub stores digital copies of the authentic documents encrypted with symmetric keys that are also included in the printed QR, but not stored by the issuer. These keys can be used by end users to decrypt the retrieved documents from the remote server.

To reason rigorously about our image comparison algorithm’s security guarantees, we geometrically model smartphone captures of paper documents. Moreover, we formally model the protocol underlying SealClub and prove that it ensures document authenticity when composed with our image comparison algorithm. We also implement our approach and evaluate its accuracy on videos of different document types and taken with multiple devices. Our evaluation shows that our approach detects all forgeries with no false alarms on the user-taken videos. Moreover, highlighted regions are large enough to be easily seen by human users, but small enough to precisely pinpoint where the forgeries occur.

**Contributions** We are the first to provide a solution combining cryptography and image comparison to authenticate printed documents in the physical world. Our solution is secure against strong attackers and it is privacy-preserving, even in the presence of compromised components and server-side data breaches.

We give novel geometric arguments to provide security guarantees for our image comparison algorithm in the presence of subtle forgeries in paper documents. We formalize our system design and provide formal guarantees for the cryptographic authentication protocol when used with the image comparison algorithm.

We also show that our method is more suitable than OCR and machine learning approaches for the problem at hand by detecting forgeries with high precision in rich-text documents printed on warped paper. Different from machine-learning approaches, our method does not rely on datasets with thousands of samples. This is relevant in our setting because, to our knowledge, there is no dataset for document forgeries that contains all the characteristics required to obtain a robust model.

Finally, we implement our solution and show that it provides high assurance in real-world scenarios by evaluating it on 128 document videos and 960 synthetically generated warped documents. Our algorithm detects all forgeries with relatively few false positives on the individual frames. False positives can be further refined by using a majority voting algorithm over a set of frames, yielding perfect accuracy (no false positives) on our evaluation videos.

**2 PRELIMINARIES**

Before describing our technical solution, we elaborate on the problem statement, the system and attacker models, and the technical
requirements. We also introduce relevant notation.

**Problem Statement.** Is it possible to build a system that supports users in authenticating paper documents and recognizing forgeries?

### 2.1 Requirements

**Functional Requirements.** Issuers can share paper documents with users. Documents may be sent by postal mail, and may be folded to fit in envelopes. Users can access a digital copy of the physical document they receive and automatically compare them to detect differences using their standard, off-the-shelf smartphone. Frames in videos of documents taken using smartphones may introduce lighting, focus, and geometrical distortions [32]. For example, unfolded documents may fail to lie flat, which changes the geometry of the original digital document. Also, the picture quality may change across smartphone models and users.

**Security and Privacy Requirements.** If an attacker modifies or replaces an existing paper document issued by a legitimate issuer before it reaches an end user, the phone-based app should alert the user to modifications in the forged document. Unauthorized parties should not be able to create new paper documents on behalf of a legitimate document issuer. If they attempt to do so, the app should inform end users of such spoofed documents. Only users in possession of a legitimate paper document can retrieve the corresponding original digital counterpart. In case of a breach on the storage service, which stores the original documents in digital form, attackers should not be able to learn the content of the documents unless they possess the corresponding paper documents. In the event of an issuer’s key compromise, users in possession of issued documents before the key is revoked should still be able to authenticate them (non-repudiation of origin under key revocation [7]). Moreover, the compromise of individual system components (issuing or storage) should not suffice to successfully create forgeries.

**Usability Requirements.** The system must tolerate a wide range of uncontrolled distortions on the video taken by end users with their smartphones. This means that the system must produce a low number of false alarms on legitimate documents, but still provide high accuracy in detecting forgeries. This is challenging because a semantic-changing forgery could be as (visually) small as a single comma, added or removed.

### 2.2 System and Attacker Model

**System Model.** There are two kinds of legitimate system users: *issuers* and *end users*. Issuers have an insecure channel, such as postal mail, for paper documents, but they can automatically distribute their public keys to users. Users connect to the storage service to securely retrieve encrypted copies of a document’s digital version. Documents have a secure timestamp that is provided by a time-stamping service, which allows end users to check when a document was created. Users also visually compare a short video of the paper document and the (original) digital version retrieved from the issuer’s servers. This comparison is assisted by an app that automates the visual highlighting of (potential) differences between frames in the video and the document’s reference image.

**Attack Model for Physical World.** Attackers may attempt to modify existing, legitimate, paper documents, by intercepting them in transit to end users or by creating new documents on behalf of issuers. To do so, they may use any forgery means, digital or physical, and may perform subtle attacks to increase their success chances. However, crucially, a forgery must be visible and semantic changing.

Note that it is hard to define semantic changing precisely in general. For instance, for some documents, like those referring to our baby seals, small punctuation changes suffice to change the document’s intended meaning. For other documents, changing the signature or a portion of a graph representing sales results might constitute a semantic changing attack. However, we assume that changing a region smaller than a punctuation mark e.g., a pixel or two, will not change the document’s intended content for human readers, but will appear to be noise.

**Attack Model for Digital World.** Attackers may attack communication channels between systems components (see Figure 1 and 5) and also the components themselves, for instance by compromising private keys or uploading data to the storage server. However, we assume end users can download an authentic app on their smartphones. In Section 4, we will show that our system is robust against very strong attackers compromising multiple, but not all, system components.

Note that if users download a fake app, this will trivially invalidate our approach’s security since the app may claim that non- authentic documents are authentic. This would also be the case if an attacker could modify the original app’s logic in a host-based attack on the mobile. We do not consider these attacks here given that they are generic attacks for which dedicated defenses are needed, such as app clone detection in app stores, anti-phishing countermeasures, and software protection.

Similarly, although our system is vulnerable to an attacker completely controlling all components on the document issuer, we consider this to be an extremely strong attacker model which is impossible to defend against. Note that such an attacker also would also invalidate the security of any distributed system for the digital signing of documents.

### 2.3 Video processing

In order to improve accuracy and to give users real-time feedback on the quality of the pictures the algorithm receives, our solution works with short videos of paper documents rather than single snapshots. However, the comparison with respect to the reference document will be performed on the individual video frames.

Given a video and a frame-by-frame comparison, we can define a notion of majority voting. Namely, for fixed numbers \( k, n \in \mathbb{N} \), we discard differences that appear in less than \( k \) out of \( n \) frames. For most of Section 3.1, we will thus describe single image (frame) comparison against the reference document. As we will discuss in the evaluation, the goal of this strategy is to distinguish actual differences in a document from sporadic false positives that may arise from temporary distortions such as a lack of focus.

### 2.4 Strict Forgery Detection

For the rest of the paper, we will consider grayscale documents and begin with some notation. Let \( n, m \in \mathbb{N} \) and \( D = [0, 1]^n \times [0, 1]^m \). A digital document \( d \in D \) is an \( n \times m \) grayscale matrix, where \([0, 1]\) is a fixed-point representation of fractions between 0 and 1 as used
in computer vision to represent pixel intensity. In the following, it will be useful to consider rectangles within an image, which we denote by two coordinates \((x_1, y_1), (x_2, y_2) \in [0, n] \times [0, m]\) such that \(x_1 < x_2\) and \(y_1 < y_2\).

A (perfect) printing function \(p: \mathcal{D} \rightarrow \mathcal{P}\) transforms the digital document into a paper version \(p\) that faithfully preserves proportion and intensity. Ideally, there exists an inverse function \(p^{-1}: \mathcal{P} \rightarrow \mathcal{D}\) such that \(d = p^{-1}(p(d))\). For instance, a perfect printing of \(d\) that is perfectly scanned should be pixelwise identical to the original \(d\).

Although forgery can happen at the digital or physical level, we can abstract away from where it occurs by comparing two digital images \(d\) and \(\hat{d}\), where \(\hat{d}\) may be \(p^{-1}(p(\hat{d}))\) (a scan of a document that was possibly digitally modified and then scanned) or \(p^{-1}(p(d))\) (a scan of authenticated printed document that was possibly physically modified). A neighbour of a position \((x, y) \in \mathbb{N}^2\) is a position \((x', y') \neq (x, y)\) such that \(\max(|x - x'|, |y - y'|) \leq 1\). In a connected set \(C \subseteq \mathbb{N}^2\), either \(|C| = 1\) (isolated point) or otherwise, all positions \((x, y) \in C\) have at least one neighbour in \(C\).

**Definition 2.1 (Ideal forgery locator).** Let \(d, d' \in \mathcal{D}\) be an authentic image and a potential forgery of \(d\), respectively. Then an ideal forgery locator \(F(d, d')\) returns a list of detected differences \(r_1, \ldots, r_k\), such that each \(r_j\) is the smallest rectangle containing a connected set of positions \(x_j, y_j\) with \(d(x_j, y_j) \neq d'(x_j, y_j)\) and the empty set otherwise.

This definition corresponds to the strictest security guarantees. In practice, however, it will be impossible to enforce such strict guarantees when working with paper documents as discussed in Section 2. Hence, we relax the definition of forgery and a forgery locator as we explain in the following.

**2.5 Limitations of existing approaches**

A natural approach to tackle the aforementioned problem statement would be to leverage existing tools such as object character recognition (OCR) or change detection algorithms such as [11, 14, 19]. However, as we shall illustrate, there are inherent limitations to such approaches, which make them unsuitable to satisfy the requirements specified in this section. Consider the OCR model from Google\(^2\) for text extraction from documents and the state-of-the-art tool for change detection presented in [4]. They propose a transformer-based Siamese network architecture, whose source code is publicly available.\(^3\) In both cases, we consider pre-trained parameters for the specific use case in which the model was designed. For the change detection model, we used the parameters trained on the LEVIR dataset [11] that contains pairs of satellite images and is labeled with the changes in the terrain between both images. We choose this because, to our knowledge, there is no publicly available dataset that considers document forgeries in the presence of warped paper. Moreover, to train such models, it is necessary to generate thousands of samples to obtain a model that generalizes correctly as in the change detection models.

In the case of OCR, we considered one of the samples used in the evaluation of the present work. After using the model, we find that the text content is interpreted properly. However, there are figures in the document that are interpreted wrongly as text, which creates false positives as shown in Figure 2. Moreover, some figures with distinct differences both produce an empty output using OCR, as shown in Figure 3a, which constitutes a false negative. In contrast, our work is agnostic to the type of document content, allowing it to detect any type of forgery, textual and graphical.

**3 SEALCLUB DESIGN**

SealClub combines algorithms for image comparison with cryptographic measures to enable end users to determine a document’s authenticity. We explain this in detail in the next sections.

**3.1 Image Comparison Algorithm**

We start by motivating the kind of algorithm required, based on the requirements of Section 2.

**3.1.1 Approximate Similarity.** In practice, when dealing with physically printed documents there will inevitably be pixel-wise differences. Definition 2.1, although appealing from a security standpoint, is too strict in practice as it will result in forgeries being detected even on pictures of documents that have not been modified (false alarms). To better capture meaningful differences between images, we introduce the notion of a difference map.

**Definition 3.1 (Difference map).** The difference map \(M\) of two images \(d\) and \(\hat{d}\) is the matrix, also in \(\mathcal{D}\), defined by:

\[
M(i, j) = |d(i, j) - \hat{d}(i, j)|, \text{ for all } (i, j) \in [1, n] \times [1, m].
\]

The difference map is a matrix containing the visual differences between two images, with more glaring differences (e.g. a black pixel compared against a white one) having higher intensity. When
Although Definition 3.3 is more tolerant to differences between (SIFT) [35] and Oriented FAST and Rotated BRIEF (ORB) [48]. We are, however, possible to filter using intensity thresholding.

To account for a tolerance to small differences, we relax our definitions of forgery and a forgery locator.

**Definition 3.2 (δ, τ)-detectable forgery.** Let \( d, d' \in D \). Let \( \delta \in (0, 1] \) and \( \tau \in \mathbb{N} \) be such that \( \tau \leq m \cdot n \). A \((\delta, \tau)\)-detectable forgery is a connected set of positions \( pos_1, \ldots, pos_k \in [1, n] \times [1, m] \) in the difference map \( M \) of \( d \) and \( d' \) such that \( M(pos_j) < \delta \), for each \( 1 \leq j \leq k \), and \( k > \tau \).

The motivation behind the choice of parameters \( \delta \) and \( \tau \) is that differences with low intensity (\( \delta \)) in the difference map correspond to less visible differences, such as blurred borders of text, and small regions (\( \tau \)) are usually noise. The concrete values of \( \delta \) and \( \tau \) must be set in a concrete implementation and may vary depending on a document’s intrinsic feature, for instance the font size. These values are defined by document issuers and are embedded in the paper document’s QR code as we will discuss in the next section. Note that setting \( \delta = 1 \) and \( \tau = 0 \) is equivalent to the stricter Definition 2.1. A \((\delta, \tau)\)-forgery locator points to regions that have been modified.

**Definition 3.3 (δ, τ)-forgery locator.** Let \( d, d' \in D \). Then a \((\delta, \tau)\)-forgery locator \( F(d, d') \) returns a possibly empty list of detected differences \( r_1, \ldots, r_k \), where \( r_i \) is the smallest rectangle containing a \((\delta, \tau)\)-forgery.

A practical challenge is that uncontrolled smartphone pictures of printed documents have unpredictable distortions as in Figure 4. Although Definition 3.3 is more tolerant to differences between images, while still providing some security guarantees, examples such as Figure 4 are challenging to handle, given that differences are large enough to be considered potential attacks. These will result in false positives. In the next section, we discuss how we tackle these challenges while still providing clearly defined security guarantees.

**3.1.2 Iterative Forgery Detection Algorithm.** An important precondition for document comparison is that the two pictures \( d, d' \) are aligned. This is non-trivial as smartphone pictures are likely to have angle and perspective differences with respect to the authentic image. A commonly used technique to rectify such differences is to apply a homography computed over the document’s estimated corners [22]. In our setting, we have an additional advantage when computing an accurate homography since we have access to the original image. We can thus use techniques to recognize known objects in new scenes. Popular algorithms to find descriptive points of an object are, for instance, scale invariant feature transform (SIFT) [35] and Oriented FAST and Rotated BRIEF (ORB) [48]. We can search for matches of such points in two images using automatic feature matching [38]. This allows us to better estimate the location of the authentic document in the printed document scan and then apply a homography to align both pictures. Figure 10 (in Appendix) illustrates the combination of both techniques (object finding) [40].

Our approach is as follows. Folds in the printed document may induce distortions that cannot be corrected with a global homography technique, given that they occur in 3D space as illustrated in Figure 4. However, if we zoom in to the neighborhood of a detected potential difference, we can attempt to recompute a homography in this limited region and check for differences. The rationale for this is that locally, in the neighborhood of a potential difference, the perspective is more consistent and thus more likely to be corrected.

**Algorithm 1 Iterative Forgery Detection Algorithm**

**Input:** Images \( d, d' \in D \) to be compared

**Output:** List of differences

1. \( d' \leftarrow \text{find}(d, d') \)
2. if \( d' \) is Null then
3. \( \Delta \leftarrow \{(0, 0), (n, m)\} \)
4. return \( \Delta \)
5. end if
6. \( d' \leftarrow \text{preprocess}(d') \)
7. \( \Delta \leftarrow 0 \)
8. \( \Lambda' \leftarrow \text{get differences}(d, d') \)
9. while \( \Lambda \neq 0 \) do
10. \( \Lambda \leftarrow \Lambda' \)
11. \( \Lambda' \leftarrow 0 \)
12. for \( r \leftarrow \Lambda \) do
13. \( \sigma \leftarrow \text{neighborhood}(\sigma) \)
14. \( z \leftarrow \text{find}(d(\sigma), d'(\sigma)) \)
15. if \( z \) is Null then
16. end if
17. \( \Lambda' \leftarrow \Lambda' \cup \text{get differences}(d(\sigma), z) \)
18. end for
19. end while
20. return \( \Lambda' \)

We now present our forgery detection algorithm. First we compute descriptive points and look for a homography, controlling for a rotation up to the angle \( \phi \). We then perform preprocessing steps similar to document scanning to improve light and contrast. Afterwards we perform a first similarity pass on the document obtaining a list \( \Delta \) of potential differences. We iterate over each difference and compare a neighborhood \( \sigma \) of the difference in the authentic image against a neighborhood of the image to verify. The comparison involves recomputing the descriptive points and recomputing the homography. We then recompute the similarity difference map in the region \( \sigma \) and store the newly found differences (if any). If no good homography is found, which can happen for instance if the printed document contains a region visually very different from the original (modified or new text/images), then we directly compare the two corresponding regions. The process can be repeated until a fixed point is reached.

Pseudocode for our algorithm is given in Algorithm 1. In this presentation, we assume that the parameters \( \tau \) and \( \delta \) are implicit to the function \( \text{get differences} \), that the size of neighborhood is implicit to \( \text{neighborhood} \), and that \( \phi \) is implicit to \( \text{find} \).

**Definition 3.4 (Similarity).** When the output of Algorithm 1 is the empty list (no forgeries found), we say that \( d' \) and \( d \) are \((\tau, \delta, \phi, \sigma)\)-similar.
Video processing. The algorithm’s output is thus a list of rectangles containing potential differences. Some of those differences may be false alarms, which occur when picture distortions cannot be automatically corrected. However, as discussed in Section 2, we used the output of single frames in the context of a video containing multiple frames to reach a verdict.

**Definition 3.5 (Majority voting).** Let \( k, n \in \mathbb{N} \). Let \( V \) an list of documents (frames) \( f_1, \ldots, f_n \). Let \( \Delta_1, \ldots, \Delta_n \) be the corresponding outputs of detected differences of Algorithm 1 with respect to a fixed reference image \( d \) and a given \((\delta, \tau)\) configuration. A rectangle \( r \) is in the majority voting list of differences of \( V \) if there exists \( k \) indexes \( j_1, \ldots, j_k \) such that \( r = \bigcap_{i=1}^{k} r_{j_i} \) and \( r_{j_i} \in \Delta_j \).

Even if false alarms can arise due to random distortions in a given frame, they are less likely to occur in a majority of frames. This voting helps improving accuracy as we show in Section 5.

### 3.2 Securing the Reference Document

Our forgery detection algorithm takes a reference image as input and it is thus essential to ensure this image’s authenticity. We refine the architecture of Figure 1 to highlight the various components and the communication channels between them in Fig 5. Documents printed by issuers are time-stamped and have a signed QR code on them that includes both the URL and the hash of the original document’s content; hence both are protected from modification and spoofing, as well as all relevant parameters. Documents signed with a revoked key are only authenticated by the app if the timestamp is older than the revocation date.

#### 3.2.1 Setup assumptions.

First, it is necessary to distribute keys so the parties involved can issue and verify documents. As usual, we assume an issuer \( I \) has generated a pair of asymmetric keys \((k_I, K_I)\), called *issuer keys*, and can sign digital documents using its private key \( k_I \). Moreover, a time-stamping service has generated an asymmetric key pair \((k_{TS}, K_{TS})\) of *time-stamping keys* used to provide secure timestamps.

The end-user app must possess the authentic public keys of the issuer and the time-stamping service. There are various standard ways how these public keys can be authentically distributed to end users. First, as discussed in the example in the introduction, users can download a legitimate app from a trustworthy app store, and the app ships with the pinned public keys. Alternatively, public keys can be scanned, in the form of a QR code, at a public physical location that is trusted by end users.

As indicated in Figure 5, we depict connections where the storage service and time-stamping service are secured using TLS. This also requires that these services have generated public-private key pairs and the app has access to the respective public keys. It is necessary however that an issuer authenticates itself with the storage server to store documents and for this either a client-side certificate can be used, or the issuer can be authenticated over the TLS channel using a pre-distributed credential authKey for this purpose. For simplicity of exposition, we assume the latter in our account below.

#### 3.2.2 Role descriptions.

**Issuer.** To issue a new document \( d \), issuer \( I \) first generates a symmetric key \( k_d \) for encryption. Afterwards it creates the QR code \( q \) to be printed together with the document \( p \). The QR code can be placed either on the document’s bottom or on its backside. After generating the QR code an adding it to \( p \), the issuer deletes both \( k_d \) and \( q \).

The QR code contains the following data:

- A URL \( U \) pointing to the encrypted reference document \( E = Enc_{k_d}(d) \).
- A hash of the original document \( h(d) \).
- The symmetric key \( k_d \) to decrypt \( E \).
- (Optional) Fallback text \( T \) describing the document for offline verification.
- The values of \( \tau, \delta, \phi \), and \( \sigma \) for this document.
- A signature \( S = Sign_{k_d}(h(U, h(d), k_d, T, \tau, \delta, \phi, \sigma)) \) produced by the issuer.
- A secure timestamp \( t \) of \( S \).
- The fingerprint of the issuer’s public key.

Here, \( U \) corresponds to the URL hosting \( Enc_{k_d}(d) \) in the storage service component, with which \( I \) can authenticate using authKey and build a secure channel using the storage service’s TLS certificate. The secure timestamp is a 3-tuple \( t = (S, time, Sign_{K_{TS}}(S, time)) \), consisting of the signature \( S \), the current time, and a signature with \( K_{TS} \) is obtained through a time-stamping service. \( I \) shares the paper document \( p \) together with the QR \( q \) with the end user.

**End user app.** Upon receiving a document \( p \) containing a QR code \( q \), the application running on the user’s smartphone looks up in its local storage the corresponding public key \( K_I \) using the fingerprint, and then verifies the signature \( S \). It also verifies the content of the timestamp \( t \) using the public key \( K_{TS} \) of the time-stamping service. It then checks whether the time used in the timestamp and the issuer key used for signing the document \( K_I \) are coherent with respect to the Certificate Revocation List. That is, if the key was revoked, whether the timestamp is older than the revocation date. If so, the app then downloads the encrypted document \( E \) establishing a secure channel with URL \( U \) (using the pinned key \( K_S \) of the storage service). It then decrypts \( E \) with the symmetric key \( k_d \) and verifies the hash of the original document. If the hash verifies, \( E \)’s plaintext is authentic and the app then passes the plaintext to the forgery detection algorithm described in Section 3.1. Alternatively, if internet connectivity is not available, the app can display the text \( T \) to the user and perform an Optical Character Recognition (OCR) on the document as a fallback mechanism.

**Storage service.** The storage service publicly hosts encrypted documents that can only be uploaded by the issuer. For this, issuers must register with the storage service, setting up a shared credential authKey to authenticate subsequent uploads.
**Time-stamping service.** The time-stamping service receives a signature $S$ from the issuer to be used in the QR code generation, and computes a secure timestamp triple $t = (S, \text{time}, \text{Sign}_{K_{TS}}(S, \text{time}))$. Anyone possessing the public key $K_{TS}$ can verify that $S$ was produced at time $t$.

3.2.3 Revocation. Revoked keys are stored, together with the time of revocation, in a publicly accessible Certificate Revocation List (CRL). This CRL is maintained by a trusted entity; this could be for example the issuer, the storage service provider, or a national certificate authority. The replacement of the revoked keys can be done on the app update. If an issuer’s private key were revoked, it is desirable that documents that were issued pre-compromise should be verifiable in order to avoid the necessity of reissuing them. This property, called non-repudiation of origin, can be achieved using secure timestamps [7, 58], which are provided by the time-stamping service. If the private key of the time-stamping service is also compromised, then non-repudiation cannot be achieved [7]. However, in Section 4, we will discuss how our system storage service provides a last line of defense against this threat.

3.2.4 Document Privacy. In SealClub, the original digital documents are available online for end users to verify the printed documents. Given that this poses the risk of a data breach, we store those copies encrypted with a uniformly random symmetric key $k_d$, unique for each document $d$. As discussed above, this key, generated by the document issuer, is stored in the printed document’s QR code, and then deleted after printing. Only users possessing the printed document or copies thereof can decrypt the original document stored in the storage cloud.

4 SECURITY ANALYSIS

In this section we analyze SealClub’s security guarantees. We examine the two main building blocks and, for each of them, explain how our system defends against different (potential) attacks that might be attempted given the threat model described in Section 2.2.

4.1 Security of Image Comparison Algorithm

To rigorously reason about the image comparison algorithm, we formulate a mathematical model of paper folds and the formation of its corresponding image taken by the camera. In the reasoning, we assume that the document verifier is not malicious. Also, we do not model lens distortion or light conditions for the following reasons. First, it is possible to mitigate lens distortions by controlling the distance a picture is taken from. This can be done by rejecting frames where the first step of the algorithm (document detection) finds a relatively small document, for instance occupying less than 70% of the image. Second, extreme light conditions, such as too little or too much light, will result in the algorithm not finding the document in the first step. Adaptive thresholding [47] can be used to handle the intermediate case where uneven lighting conditions generate shadows.

**Choosing an appropriate $\tau$.** Intuitively, the choice of $\tau$ represents the minimum size of an expected connected forgery. For instance, $\tau$ could be chosen by measuring the area of a comma in a given document, based on the font type and size used. Note that when considering a warped paper surface, the effect of perspective may impact how an area in the geometry of the flat paper sheet is seen by a mobile camera. In the following, we will argue that for every $\tau$, a conservative $\tau' = \rho \cdot \tau$ for $\rho \approx 1$ can be selected such that the algorithm will detect arbitrary $\tau$-forgeries in presence of folds in the paper.

Let a family of paper folds similar to the one represented in Figure 6 be given. Without loss of generality, we assume paper is folded in two in the page’s middle, warping it into two symmetrical half cylinders. The reasoning for more folds and folds in different parts of the paper is similar. This folding is described by a function $b$ such that the paper surface is defined as follows.

**Definition 4.1.** Let $b : \mathbb{R} \to \mathbb{R}$ a positive continuous function (i.e. one with positive range), such that $b$ is piecewise differentiable and continuous. We say that the paper is warped as a half cylinder determined by $b$ if the paper surface can be described as $P = \{(x, y, b(x)) \in \mathbb{R}^3 \mid (x, y) \in \mathbb{R}^2\}$.

For example, $b$ can be the function $ax \cdot e^{-\lambda x}$, but the reasoning can be applied to other differentiable functions as well. A thorough discussion on the choice of $b$ can be found in Appendix C.

Our strategy consists, first, in considering two rectangles with the same area in both the warped paper and the flat paper; then we project both in a “virtual plane” that resembles the image formation in a camera; finally, we compute the ratio $\rho$ of the areas of both projections. We then generalize this result to a forgery comprised of multiple adjacent rectangles. The details of the computation of $\rho$ are presented in the Appendix B. There, we present the complete geometric construction using the pinhole model [21, 43] and the detailed proof of the following theorem, as well as an analysis of the assumptions and limitations of our calculations. We omit in this section the theorem for the case of a forgery comprised of just one rectangle, which can be found in Appendix B.

Before formally stating Theorem 4.2, we provide intuition on its purpose. Suppose that we have a grid over the warped paper dividing the paper into rectangles. If we take one of these rectangles, we can project it into the virtual plane and obtain a shape with some given area. If we then flatten the paper, the same rectangle

![Figure 6: The plane view of a flat document vs warped.]

![Figure 7: Projection of a rectangle from a flat document (blue) and a warped document (red) into the virtual plane.]



169
can be also projected to the virtual plane to form another shape whose area may differ with respect to the projection of its warped version, as illustrated in Fig 7. We call $\rho_i$ the ratio of the areas of both projections associated with the $i$-th rectangle. We can compute this ratio for every rectangle in the grid and then take the minimum among all ratios, which will be called $\rho$. Now, consider a forgery that is comprised of multiple adjacent rectangles in the grid. As with the rectangles, we can project the forgery into the virtual plane using both versions of the paper: the warped and the flattened one. Theorem 4.2 says that if the algorithm detects the projected forgery in the flattened version using the parameter $\tau$, then the algorithm will detect it in the warped version using the parameter $\tau' = \rho \cdot \tau$.

**Theorem 4.2.** Let $\tau$ and $\delta$ be parameters for the Algorithm 1. Suppose that we have a paper warped as a half cylinder determined by the positive function $b : \mathbb{R} \to \mathbb{R}$, where $b$ is piecewise differentiable and continuous. Assume that we impose a grid over the plane comprised of disjoint rectangles $\{R_i\}_{i=1}^n$, such that the grid coincides with the borders of the warped paper and with the points where $b$ is not differentiable. Let $\{R_i\}_{i=1}^n$ a subsequence of adjacent rectangles and let $\alpha_i \in \mathbb{R}$. Define $\alpha_{w} \equiv \{(x, y, b(x)) \mid (x, y) \in \mathbb{R}\}$, and let $\alpha$ be the projection of $\alpha_{w}$ on the virtual plane. Let $\alpha'_w$ be the same region as $\alpha_{w}$, but considering the flat paper sheet. Consider the region $\alpha_{w} \equiv \{(x, y, b(x)) \mid (x, y) \in \mathbb{R}\}$ and let $\alpha'_f$ be the same region as $\alpha_{w}$, but in the flat paper sheet. Let $\rho_i$ be the area ratio between the areas of the projections of $\alpha_{w}$ and $\alpha'_w$ when projected in the virtual plane, and let $\rho = \min \rho_i$. Suppose that the projection of $\alpha'_f$ in the virtual plane is a $\delta, \tau$-detectable forgery. If $\tau' = \rho \cdot \tau$, then $\alpha$ is a $\delta, \tau'$-detectable forgery.

A proof is given in Appendix B.1. Note that once a printing resolution is fixed, we can impose an appropriate grid in which each rectangle in the real-world corresponds to the basic unit of a pixel. This allows us to express arbitrary forgeries as a connected set of rectangles. We now proceed to connect Theorem 4.2 to the main algorithm.

**Theorem 4.3.** Let $d$ be an original document and let $d'$ be a picture of a document $p$ containing a $(\delta, \tau)$-forgery $\alpha$ and folded according to the piecewise differentiable continuous function $b : \mathbb{R} \to \mathbb{R}$. Then there exists an $r_i$ in the output of Algorithm 1 with parameters $\delta, \tau'$, for $\tau'$ computed according to Theorem 4.2 for $b$, such that $\alpha \in r_i$.

Proof. By contradiction, assume the output of Algorithm 1 does not contain a rectangle $r_i$ such that $\alpha \in r_i$. There are two cases when this can happen: (a) $\alpha$ was not detected in the first global difference calculation (line 8), and thus the algorithm never enters the loop (line 9) or (b) $\alpha$ is found initially, but in some round $k$ it is deleted from $\Delta$.

Case (a). In this case, sufficiently many characteristic points of $d$ were detected in $d'$ and thus a homography was found, as otherwise the algorithm would have returned $\Delta = \{(0, 0), (n, m)\}$, which trivially contains $\alpha$. Under the assumptions of Theorem 4.2, arbitrary forgeries of size bigger than $\tau$ will be detected on the picture of the folded paper, which leads to a contradiction.

Case (b). In this case, we use a loop invariant to establish a contradiction. Here, we will assume that the homographies performed in small sections of the paper will result in images similar to the flat version of the document. So, if there is forgery $\alpha$ such that $|\alpha| > \tau$, a homography in a neighborhood of $\alpha$ will look very similar to the original forgery and it still holds that $|\alpha| > \tau$. The invariant we want is the following: Assume that $\alpha \in \Delta'$ is a result from Line 8. Every time the execution reaches Line 9, $\alpha \in \Delta'$. A detailed proof of this invariant can be found in Appendix D.

In summary, this argument shows that if a forgery $\alpha$ is detected in the initial scan, then it will be contained in some rectangle $r_i$ at the end of the execution (no false negatives). Central to this argument is that $\alpha$ is found in the first homography, which holds under the hypothesis of Theorem 4.2.

Note that it is important to choose $\phi$ so that the family of equivalent scans does not alter the document’s semantics. For instance, choosing $\phi = \pi/2$ could result in accepting semantics-altering forgeries, such as an inverted chart, whereas $\phi = \pi/20$ is less likely to do so.

*From single frames to video.* As discussed in Section 2, the verdict on a document video will be reached once a majority vote on $k$ out of $n$ frames has been performed. This lowers the likelihood of errors due to extreme random distortions, and otherwise preserves the security argument for individual frames.

### 4.2 Security of Reference Document

We now discuss the security guarantees that our system provides, under the assumption that Algorithm 1 provides a secure comparison of digital documents with smartphone pictures of paper documents. We start by formalizing SealClub’s main security goal.

**Definition 4.4 (Document Origin Authentication).** For every issuer $I$ with private key $k_I$ and public key $K_I$, all byte-strings $U, \kappa, T, t$, and all $\delta, \phi, \sigma \in \mathbb{N}$, and all $d, d' \in D$ such that:

- $d'$ and $d$ are $(\tau, \delta, \phi, \sigma)$-similar,
- $d'$ contains the signature $\mathcal{S}_{\text{sign}}(h(U, h(d), \kappa, T, t, \tau, \delta, \phi, \sigma))$ in its QR code,
- $\{d\}_K$ is hosted at the URL $U$, and
- If $K_I$ has been revoked, then the timestamp $t$ corresponds to $S$ and to a time older than the revocation timestamp in the CRL.

it holds that either $d'$ is a picture of a document issued by $I$, or both the private key of the issuer $k_I$ and the server hosting $\{d\}_K$ at $U$ have been compromised.

We have produced a formal, computer-checked proof that Definition 4.4 holds for our design, with respect to our attacker model, using the Tamarin model-checking tool [37]. Tamarin is a symbolic model-checker for security protocols and we use it to model the actions and interactions between different system components, including the attacker compromising them and learning their associated keys.

Following our attacker model, we model the physical channel between issuers and end users as an insecure channel, where tampering and spoofing of the document and the QR code can occur. We also model key $k_I$ compromise and revocation, which corresponds

---

4Unless angles are crucial for the document type. If so, flatbed scans would be advisable.

5 https://github.com/ldvanegasam/sealclub-artifacts.
to the adversary gaining control of the issuer’s long term secret. The compromise of the storage service means that attackers can upload and replace arbitrary documents on this service. Note that, following the attacked model defined in Section 2.2, we do not model the compromise of the end-user app; if this were compromised an attacker could falsely authenticate any document trivially.

Tamarin produces a proof of this property automatically in under one second on a commodity laptop. Intuitively this property holds because there is defense in depth where individual compromises are insufficient to violate this property. For example, compromising the issuer’s private key $k_I$ is not sufficient to forge a document, since if no corresponding digital version is stored on the storage service, the app will not be able to carry out the authentication. Moreover, compromising the storage service and only uploading a forged document without generating a valid QR code with the key $k_I$ will be detected by the app when verifying the signature $S$. Finally, if we consider the compromise of the the timestamping service, which is equivalent to ignoring the timestamp coherence check in the model, our approach is secure unless the attacker additionally compromises both the issuers key $k_I$ and the storage service. This result validates our design in the presence of strong adversaries who can compromise some, but not all, of the system components.

We provide next additional intuition on potential attack vectors and describe how they are prevented by our system design.

**Attacker forges QR code and redirects to its own URL.** This is detected by the system if there is no matching trusted (i.e. not revoked) public key $K_I$ for the legitimate issuer $I$ that can verify the signature $S$. This attack will only succeed if the attacker can obtain a digital signature for a trusted public key $K_I$ as well as compromising the storage service before $K_I$ is added to the Certificate Revocation List.

**Attacker uploads arbitrary data to storage service.** Our attacker model allows for the compromise of the individual system components in Fig. 5. Assume an attacker manages to compromise the storage service and replaces the (encrypted) reference document $Enc_{K_I}(d)$, pointed to by $U$, by some other version $Enc_{K'_I}(d')$. This attack will fail unless the attacker can generate a valid QR code matching the new $K'_I$ and the new document hash $h(d')$, for which they would also need to compromise the issuer’s key $k_I$.

**Other attacks.** As explained previously, we assume during setup that public keys are distributed authentically. Concretely, consider the bank issuer example in the introduction. In this setting, users have a legitimate app installed on their phones with an authentic issuer’s public key pinned. As a result, phishing attacks in the form of a malicious app are not possible because the app is delivered to users using trusted channels according to our system model.

## 5 EVALUATION

In this section we evaluate of our forgery detection algorithm, which is at the core of our approach. To do so, we implemented Algorithm 1 in Python using the OpenCV library [41]. We perform two kinds of evaluations. First, we consider a set of videos of printed documents that have been folded in halves or thirds. This reflects the typical scenario of users receiving a document from postal mail. In order to consider more diverse and challenging scenarios, we also consider a synthetically generated dataset where we randomly fold a digital version of a document in 3D space to simulate unexpected paper folds. Both datasets are balanced in terms of forged and legitimate documents. The set of evaluated videos and warped images with annotations are publicly shared.

### 5.1 Evaluation on videos of printed documents

**Dataset.** We evaluate our prototype’s performance on a set of 128 videos taken from 32 printed documents with 4 different smartphones, including 2 Android (Xiaomi redmi note 8 and 11) and 2 iOS devices (iPhone 6s and 12pro). Each video has a duration of approximately 5 seconds. Videos were taken under different lighting conditions and printed documents were either unfolded or previously folded in halves or thirds. For each of these document classes, there are 4 legitimate documents with different contents, dates, and signatures. For each of the 16 legitimate documents, we produce a forgery, ranging from very subtle (e.g. a minus sign before a number or altering one digit in a date) to relatively noticeable (e.g. removing a sentence or changing the name of the signing individual). The full dataset of original and forged documents is given in the repository.

**Parameters.** Before analysing the videos of the printed documents, for each document class we choose values for $\delta$ and $\tau$ such that the digital versions of the 4 forgeries in this class are correctly detected by the algorithm. This yields slightly different, but similar, values for all classes, which is explained by the fact that they have different font types, sizes, and character sets. The values of $\tau$ range from 15 to 20 pixels, which is about half the size of a comma in the reference document resolution. The value of $\delta$ is the same for all documents at 0.39. We set $\sigma$ to be the same for all documents at 60px × 60px and for efficiency’s sake we only perform two refinement rounds instead of computing a fixed point. We observe that additional rounds do not significantly improve the analysis for our dataset. We set $\phi = 30^\circ$.

**Results.** As discussed in Section 2.3, we perform a majority vote on the frames to help distinguish actual forgeries from sporadic false positives occurring on individual frames. Our analysis annotates a total of 89 regions as forgeries. These correspond exactly to all forgeries in the documents used for the evaluation. That is, for each rectangle containing a forgery in the ground truth, our method finds a rectangle that non-trivially intersects it. This intersection is on average 99% of the area of the ground truth rectangle. The area of the estimated rectangle by the algorithm is on average 0.94% the area of the document (i.e. less than 1%).

![Figure 8: Outcome of algorithm on a forged document.](image-url)
False alarms arise when there are misalignments with respect to the original documents that are not be resolved by our iterative technique, for instance because no good homography was found for that document region. Although sporadic false alarms occur in some frames of our videos, the majority voting analysis requiring 6 out of 7 matches prevents them from occurring in the final reported result. In the 856 frames processed in all videos, only 86 false positives are found by the algorithm (on average, approximately one false positive on 10% of the frames). However after applying the majority voting the final verdict contains no false positives.

Figure 8 shows a paper document and an original document together with the automatically added annotations. The running time of the individual frame analysis ranged from 0.5 to 1 second, depending on the number of initially estimated differences in the algorithm’s first round. For our evaluation, we perform an offline analysis at the rate of 4 frames per second. The rationale behind this sampling rate is that successive images should be sufficiently distinct but the analysis should converge as fast as possible. By choosing a majority voting on 7 frames, our analysis makes a decision based on 1.75 seconds of video.

5.2 Evaluation on synthetically generated 3D warped images

We evaluate our algorithm on multiple artificially warped documents as proposed by Ma et. al. [36]. In their work, authors generate a finite number of deformations over a mesh, and then, the mesh is applied over the image to obtain a warped version. We adopt the proportion of curving and folding deformations from the work of Ma et. al. Note that we do not simulate lighting conditions given that from our previous evaluation on videos of paper documents with various lighting conditions, we obtained very good results on removing shadows using adaptive thresholding [47]. As an example, Figure 9 shows an artificially warped document. We highlight foldings and curvings by adding a grid in an intermediate step.

For the evaluation we consider the same 16 documents and their corresponding forged version as in the dataset discussed in the previous subsection. For each document and each forgery, we generate 30 synthetically warped images using the technique mentioned above, resulting in a total of 960 documents.

In this evaluation, we set \( \tau = 15 \) and \( \delta = 0.39 \), as we did for the videos in the previous subsection. Our evaluation results show that our algorithm has no false negatives and 302 false positives. This is on average 1 false positive (of size less than 1% of the document size) on every third document. Note that here, the number of false positives occurring in individual frames is higher than in the case of the videos. This is because the random warps considered are geometrically more complex than those warpings normally occurring in papers folded in halves as in the evaluation with printed documents. Nevertheless, we still obtain no false negatives, which coincides with our security goal. Here we do not perform a majority voting since we want to test the performance of our algorithm on individual randomly warped images and we do not have multiple similar images to apply the majority voting as in the video evaluation. In practice this represents a worst case scenario where a human would need to manually compare the detected potential difference against the original. An evaluation on printed documents folded in random, challenging ways, where videos with multiple frames are available and thus a majority voting can be applied, is left for future work.

5.3 Discussion and limitations

Our evaluation on two distinct datasets shows that it is possible to configure our algorithm in a way that it detects subtle forgeries while keeping the false positive rate low. False positives can be extremely rare when considering videos of documents and a majority vote. We observe a higher false positive rate in the synthetic dataset. This is caused by more challenging foldings in the synthetic dataset with respect to the videos taken on printed documents.

Our evaluation has some limitations. In practice, users may produce videos under more challenging conditions, for example due to shadows, lack of focus, stains etc. Thus, a comprehensive user study where participants are asked to take their own videos on their devices should be designed and carried out; this is left for future work. Currently, we are collaborating with a major European city to deploy our technology in the field and this will allow us to get further insights into its usability.

In our problem domain, it may happen that the printed document has benign changes like water spillage or other unintentional marks. Our approach will then detect such marks as false positives and highlight for the user the benign changes. If we design an algorithm that ignores such changes, it may ignore similar marks in other situations where such differences are included with malicious intentions. Therefore, it is left to the user to judge whether those false positives are benign or not, and our approach will help the user to clearly identify them. The effect of such benign changes in the usability of the application is an interesting question for a user study as mentioned above.

Furthermore, note that choosing of an appropriate \( \tau \) is a challenging task that depends on the smallest expected forgery. This, in turn, may vary depending on the document type. It is in principle possible that an attacker may attempt to break down an attack into disconnected but close regions that are individually smaller than \( \tau \) in order to bypass our analysis. However it is debatable whether such attempts would actually constitute a semantic changing attack because some of them might look suspicious to the human eye or even be unnoticeable. Despite this, we can chose a lower \( \tau \) to detect such attacks at the price of potentially more false positives.
6 RELATED WORK
Extending paper documents with characteristics that facilitate digital authentication has been discussed by Eldrefawy et al. [17], Wang et al. [56], and Li et al. [31]. In contrast to these, our solution provides an algorithm for automatic image comparison, and gives a detailed analysis of its security guarantees. In the following, we compare with other approaches related to SealClub.

OCR approaches. OCR has known accuracy issues [30], stemming from problems similar to those addressed in this work, such as lighting, foldings, and overall picture quality. In Section 2.5, we show limitations of OCR for documents that include images. In contrast, our approach accounts for pictures’ and characters’ positions. Also, OCR assumes known character sets, whereas we can detect differences in new alphabets or character fonts. Ambadiyil et al. [2] present an approach that can be verified using OCR technology by hashing critical parts of a text document. Our approach is more general as it can detect forgeries throughout the document and on its non-textual parts.

Machine-learning approaches. The research most closely related to our image comparison technique is [3]. This work compares scanned documents and highlights visual differences to detect forgeries. Two fundamental differences with our approach are that their approach is tailored for textual documents and that they rely on high quality flatbed scans. Works such as Kim et al. [29] use machine learning to characterize differences between documents with a focus on text documents and version management.

It is challenging to attain generalizability of a deep neural network in our context given that one may not consider all possible interesting forgeries or types of authentic documents. Also, defenses against adversarial attacks on such solutions is still a topic of ongoing research [10, 12]. In our setting, we can mathematically characterize attacks and thus provide precise accuracy guarantees under reasonable assumptions.

Other forgery detection approaches. Van Beusekom et al. [54] propose a technique that creates signatures out of known authentic documents. This technique can detect forgeries where attackers scan and reprint original documents, since in this process distortions of known authentic subregions of the document tend to occur. This approach however would not detect sophisticated forgeries that, for instance, simply add a comma to an authentic document.

Similarly, the work of Picard et al. [42] on copy detection patterns raises the bar against forgery under the assumption that copying/reprinting variants of an original document also degrades their quality. Although this technique can detect some forgeries, it cannot pinpoint the specific regions that have been tampered with and its security guarantees are unclear [28].

Document dewarping. Research on document dewarping [49, 51] is related but orthogonal because it does not assume knowledge of ground truth (the authentic image). Often the goal of document dewarping is to improve OCR. In contrast, our goal is more general since forgeries could also happen in graphics like charts, signatures, and other images that cannot be parsed using OCR.

Alternative image similarity. It is possible to instantiate our image comparison approach with alternative image similarity functions, such as the structural similarity index (SSIM), [57]. SSIM was designed to measure visible differences between versions of an image from a human user’s point of view. We have preliminary experimented with SSIM, and although it offers advantages for reducing false positives, it poses challenges for forgeries that overlap content present in the original image.

Image change detection algorithms. Our work can be seen as a particular case of the change detection problem [44]. There exist approaches that use geometric adjustments like affine and projective transformations, among others, in pre-processing phases [59] of image change detection algorithms. However, our methodology builds upon these techniques to solve the comparison problem. Specifically, our method can be considered as a specific case of “simple differing” [27, 45, 46]. Given that our approach focuses on printed document authentication, we leverage the specifics of our use case to prove the security of our proposal using geometrical arguments. We did not find concrete implementations of methods based on simple differenting to make experimental comparisons with our approach. The most recent advances in change detection are based on machine learning [1, 9, 33, 55]. As discussed above, using machine learning models in our setting has generalizability and security implications, and to the best of our knowledge, there is no labeled dataset large enough that meet the requirements of our problem. Moreover, there are some accuracy issues of pre-trained networks as discussed in Section 2.5.

Non-reputation and revocation. Key revocation is challenging for non-reputation in the context of digital signatures [7]. Various solutions have been proposed, such as secure timestamping [58] or forward secure signature schemes [6, 7]. In SealClub, we use secure timestamps to avoid the necessity of reviving all documents signed prior to a key being revoked due to compromise. Commercial services such as [50] provide similar time-stamping guarantees. In our setting, a malicious storage administrator can delete the reference image corresponding to a given issued document, thus de-facto repudiating it. A decentralized storage solution would mitigate this threat, but its formal treatment is left for future work.

To the best of our knowledge, we are the first to propose a comprehensive approach to automatically verify the authenticity of rich paper documents with the help of commodity smartphones by using image comparison and cryptography.

7 CONCLUSIONS
We have presented SealClub, a novel approach that uses cryptographic and image comparison techniques to authenticate rich paper documents containing text and graphics. Our solution provides security guarantees along with rigorous proofs. A preliminary evaluation of SealClub shows that it significantly raises the bar against forgery in challenging scenarios. In the future, we plan to perform a large-scale user study to further support our claims and gain more insights regarding its usability.

ACKNOWLEDGMENTS
We thank the Werner Siemens-Stiftung for their generous support of this project, under the Centre for Cyber Trust. We would also like to thank Adrian Perrig and the anonymous reviewers for their helpful comments to earlier versions of this paper.
REFERENCES

[1] Pablo F. Alcantarilla, Simon Stent, Germain Ros, Roberto Arroyo, and Riccardo Gherardi. 2018. Street-view change detection with deconvolutional networks. Autonomous Robots 42, 7 (01 Oct 2018), 1301–1322. https://doi.org/10.1007/s10452-018-9734-5

[2] Sajan Ambadyul, V B Vibhath, and V. P. Mahadevan Pillai. 2016. Performance analysis and security dependence of on paper digital signature using random and critical content. In 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES). 722–725. https://doi.org/10.1109/SCOPES.2016.7955534

[3] Elena Andrevea, Vladimir V Arlaçov, Oleg Slavin, and Alexeys Mishev. 2020. Comparison of scanned administrative document images. In Tenth International Conference on Machine Vision (ICMV 2019), Vol. 11433. International Society for Optics and Photonics, 114333A.

[4] Wele Gedara Chaminda Bandara and Vishal M. Patel. 2022. A Transformer-Based Siamese Network for Change Detection. In IGARSS 2022 – 2022 IEEE International Geoscience and Remote Sensing Symposium. 207–210. https://doi.org/10.1109/IGARSS546834.2022.9883866

[5] BBC news. 2021. ‘Staggering’ trade in fake degrees revealed. https://www.bbc.com/news/business/61 (2023), 1–11. https://doi.org/10.1109/TGRS.2023.3277496

[6] Johannes Braun, Franziskus Kiefer, and Andreas Hülsing. 2013. Revocation and non-repudiation: when the first destroys the latter. In European Public Key Infrastructure Workshop. Springer, 31–46.

[7] Business Week. 1975. The Office of the Future . http://web.archive.org/web/20160508055917/http://www.bloomberg.com/bw/story/1975-06-30/the-office-of-the-futurebusiness:news-stocks-market:stock-and-financial-advice .

[8] Andrea Codegoni, Gabriele Lombardi, and Alessandro Ferrari. 2023. TINYCD: a (not so) deep learning model for change detection. In Proceedings of the 25th International Conference on Pattern Recognition (ICPR). IEEE, 3162–3167.

[9] Rodrigo Caye Daudt, Bertr Le Saux, and Alexandre Boulch. 2018. Fully Convolutional Siamese Networks for Change Detection. In 2018 25th IEEE International Conference on Image Processing (ICIP). 4063–4067. https://doi.org/10.1109/ICIP.2018.8451652

[10] Anirban Chakraborty, Manan Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2021. A survey on adversarial attacks and defences. Computer-Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 8-14, 2013. Proceedings. 696–701. https://doi.org/10.1007/978-3-642-39799-8_48

[11] Hao Chen and Zhenwei Shi. 2020. A Spatial-Temporal Attention-Based Method and a New Dataset for Remote Sensing Image Change Detection. Remote Sensing 12, 10 (2020). https://doi.org/10.3390/rs121021662

[12] Lu Chen, Jiao Sun, and Wei Xu. 2021. FAVA: Fast Adversarial Watermark Attack on Optical Character Recognition (OCR) Systems. In Machine Learning and Knowledge Discovery in Databases, Frank Hutter, Kristian Kersting, Jefrey Lijffijt, and Isabel Valera (Eds.). Springer International Publishing, Cham, 547–563.

[13] Virginia Clinton. 2019. Reading from paper compared to screens: A systematic review and meta-analysis. Journal of Research in Reading 42, 2 (2019), 288–325. https://doi.org/10.1111/1467-9817.12269

[14] Andrea Codigoni, Gabriele Lombardi, and Alessandro Ferrari. 2023. TINYCD: a (not so) deep learning model for change detection. In Proceedings of the 25th International Conference on Pattern Recognition (ICPR). IEEE, 3162–3167.

[15] Debraj Das, Debdeep Mukhopadhyay. 2021. A survey on adversarial attacks and defences. In Proceedings of the 25th International Conference on Pattern Recognition (ICPR). IEEE, 3162–3167.

[16] Chak Man Li, Pili Hu, and Wing Cheong Lau. 2015. Authpaper: Protecting paper-based documents using authenticated 2D barcodes. In 2015 IEEE International Conference on Sensing, Security and Cybernetics. IEEE, 740–746.

[17] Chanchal Dutta, Benjamin Schonhofer, and Martin Schuhmacher. 2015. Visibility Cues for Copy-Detection Patterns. In Proceedings of the 27th Conference on Computer Vision and Pattern Recognition. 4700–4709. https://doi.org/10.1109/CVPR.2015.7298867

[18] Marius Muja and David G Lowe. 2009. Distinctive image features from scale-invariant keypoints. International journal of computer vision 60, 2 (2004), 29–41.

[19] Simon Meser, Benedikt Schmidt, Cas Cremers, and David A. Basin. 2013. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 15-19, 2013. Proceedings. 696–701. https://doi.org/10.1145/3624591.3624712

[20] Hindustan Times. 2017. University of Mumbai degrees will now come with QR codes. https://www.hindustantimes.com/mumbai-news/university-of-mumbai-degrees-will-now-come-with-qr-codes/story-z2YXaXU1U5Z1OdWpnjQ8Q.html. Online; accessed December 17 2021

[21] 2021. PKS 2020 - Zeitrei-
A DOCUMENT DETECTION

Figure 10: Descriptive feature matching and estimation of perspective

Figure 10 illustrates how key-point matching and estimation of perspective allows one to find an estimated location of a document in a smartphone taken picture and retrieve it by adjusting the perspective back to a flat approximation.

B CHOOSING AN APPROPRIATE \( \tau \)

As discussed in Section 4, the key to computing a more conservative \( \tau \) is to compute the ratio \( \rho \). To do this, we will assume that the paper is modeled as a half cylinder in \( \mathbb{R}^3 \). The \( xy \)-view of the page will be modeled as a function \( b : \mathbb{R} \rightarrow \mathbb{R} \) that is differentiable for \( x > 0 \). This function models the right side of the paper with respect to the warp. The left side is a reflection of the curve described by \( b \) with respect to the \( y \)-axis. The setup is represented in Figure 6. Here, we are assuming that the initial (global) homography was done perfectly when the paper is scanned and the camera is at the top of the page. To analyze how the area changes in the presence of warping, we use a variant of the pinhole model, which is simple but useful to analyze the geometry of the problem. In this model, each point is projected over a cone with vertex \( S \) in a plane \( x'y' \) called the "virtual plane". For more details about this model, see [21, 43].

Consider a rectangle \( R = [x_0, x_1] \times [y_0, y_1] \), for \( x_1, x_0, y_1, y_0 > 0 \). This rectangle is in the \( xy \)-plane, and we can use the function \( b \) to embed this rectangle in the surface of the warped paper. The resulting embedding will be a small warped rectangle modeled as a cylinder positioned in the page sheet defined as

\[
R_w \overset{\text{def}}{=} \{(x, y, b(x)) \in \mathbb{R}^3 : (x, y) \in R\}.
\]

Let us take a point in \( R_w \) and compute its coordinates in the \( x'y' \)-plane (the virtual plane). The situation is depicted in Figure 11.

Due to the similarities of triangles, it holds that

\[
\frac{x'}{x} = \frac{SO'}{SO - b(x)}, \tag{1}
\]

and therefore,

\[
x' = \frac{SO'}{SO - b(x)} \cdot x. \tag{2}
\]

With a similar construction, we have that for the \( y' \)-coordinate,

\[
y' = \frac{SO'}{SO - b(x)} \cdot y. \tag{3}
\]

which results in

\[
y' = \frac{SO'}{SO - b(x)} \cdot y. \tag{4}
\]

Note on notation: throughout the text, if we refer some object (point, segment, length) in the virtual plane, we use tilda after its name. Also, we use the subscript "w" standing for "warped",...
When \( R \) is projected to the virtual plane in a rectangle \( A \), the area of \( A \) is the length of \( f \) as a function \( (x, y) \rightarrow (x, y) \).

Then, the area of \( A(R_w) \) is
\[
A(R_w) = \int_{R_w} |\det \mathbf{J}f|.
\]
In this case,
\[
D_f = \left( \frac{\partial}{\partial x} P(x, y), \frac{\partial}{\partial y} P(x, y) \right).
\]
After computing the partial derivatives, we have that
\[
|\det \mathbf{J}f| = \left| \frac{\partial}{\partial x} P(x, y) \right| \left| \frac{\partial}{\partial y} P(x, y) \right|
= \left( SO^2 \right)^2 \frac{|SO - b(x) + x \cdot b'(x)|}{|SO - b(x)|^3}.
\]
And finally,
\[
A(R_w) = \int_R |\det \mathbf{J}f| = \left( SO^2 \right)^2 \int_{R_w} \frac{|SO - b(x) + x \cdot b'(x)|}{|SO - b(x)|^3} dx.
\]
Now, let us find the projection of a flat rectangle in the virtual plane. This rectangle must be completely flat, and it must have the same area in the "real world" as \( R_w \). We will denote this rectangle as \( R \).

Let \( I = [x_0, x_1] \). Formally, the length of \( L_w \) can be computed as
\[
L(w) = \int_{x_0}^{x_1} \sqrt{1 + |b'(x)|^2} dx.
\]
Then, the area of \( R \) can be computed as \( A(R_w) = (y_1 - y_0) \cdot L_w(I) \).

When \( R \) is projected to the virtual plane in a rectangle \( A \), we can compute its area as
\[
A(R'_A) = \left( \frac{SO^2}{SO} \right)^2 \cdot A(R_i).
\]
Taking both Equation (9) and (11), we define its ratio as
\[
\rho_i \overset{\text{def}}{=} \frac{A(R'_w)}{A(R'_A)} = \frac{(y_1 - y_0)^2}{A(R_i)} \int_{x_0}^{x_1} \frac{|SO - b(x) + x \cdot b'(x)|}{|SO - b(x)|^3} dx.
\]
Now we can state the first theorem for rectangular forgeries as follows.

**Theorem B.2.** Let \( \tau \) and \( \delta \) be parameters for the Algorithm 1. Suppose that we have a paper whose warps are determined by the positive function \( b : \mathbb{R} \rightarrow \mathbb{R} \), where \( b \) is piecewise differentiable and continuous. Let \( R = [x_0, x_1] \times [y_0, y_1] \subseteq \mathbb{R}^2 \) be a rectangle. Let \( \alpha_w \overset{\text{def}}{=} \{(x, y) \in R \} \) and \( \alpha \) be the projection of \( \alpha_w \) in the virtual plane. Define \( \alpha_f \) to be the same region as \( \alpha_w \) but considering the flat paper sheet. Suppose that the projection of \( \alpha_f \) in the virtual plane is a \((\delta, \tau)\)-detectable forgery. Let \( \rho \) be as explained in Equation (12). If \( \tau' \overset{\text{def}}{=} \rho \cdot \tau \), then \( \alpha \) is a \((\delta, \tau')\)-detectable forgery.

Proof. According to the definition of \( \alpha \), \( |\alpha| = A(R'_w) \), and the area of the projection of \( \alpha_f \) on the virtual plane is \( A(R'_i) \). Considering the hypothesis, \( A(R'_i) > \tau \). So,
\[
|\alpha| \overset{\text{def}}{=} A(R'_w) = \frac{A(R'_w)}{A(R'_i)} \cdot A(R'_i) = \rho \cdot A(R'_i) > \rho \cdot \tau = \tau'.
\]
Hence, \( \alpha \) is a \((\delta, \tau')\)-detectable forgery.

**B.1 Proof of the Theorem 4.2**

We proceed to prove the Theorem 4.2 stated in Section 4. This theorem can be considered as a generalization of the Theorem B.2.

First, according to the Appendix B, we have that
\[
|\alpha| \overset{\text{def}}{=} A(R'_w) = \sum_{j=1}^{k} A(R'_{i_j}).
\]
On the other hand, the area of the projection of \( \alpha_f \) is
\[
A(R'_i) = \sum_{j=1}^{k} A(R'_{i_j}).
\]
By definition,
\[
\rho_i \overset{\text{def}}{=} \frac{A(R'_w)}{A(R'_i)}.
\]
Then,
\[
|\alpha| = \sum_{j=1}^{k} A(R'_{i_j}) = \sum_{j=1}^{k} \rho_i A(R'_{i_j}) \geq \sum_{j=1}^{k} \rho A(R'_{i_j})
= \rho \sum_{j=1}^{k} A(R'_{i_j}) = \rho \cdot A(R'_i) > \rho \cdot \tau = \tau'.
\]
CHOOSING AN APPROPRIATE FUNCTION $b$

In the above computations, the function $b$ can be instantiated with any function piecewise differentiable and continuous. We can choose a suitable function to give concrete results. A good candidate for $b$ would be

$$b(x) \overset{\text{def}}{=} \omega x \cdot e^{-\lambda x},$$

(13)

for $\omega, \lambda > 0$. According to empirical experience, this $b$ models a warp in the paper sheet when it is made in the half of the page. This would be a good starting point due to the real-world application of Theorem B.2 and Theorem 4.2 in official document authentication tasks, where the letters are commonly folded with warps of this type or similar.

By choosing this function $b$ we can restate the ratio in Equation (12) as

$$\rho \overset{\text{def}}{=} \frac{A(R_I')}{A(R_I)} = \frac{\left(\frac{SO}{SO - \lambda x \cdot b(x)}\right)^2}{\int_{x_0}^{x_1} \left(\frac{SO - \lambda x \cdot b(x)}{SO - b(x)}\right)^3 \ dx}. \tag{14}$$

Moreover, by setting the function $b$ to be of such form, we can find an upper bound for $A(R_I')$. We know that $b'(x) = \omega e^{-\lambda x}(1 - \lambda x)$, and replacing this value in Equation (10), we have that

$$\int_{x_0}^{x_1} \sqrt{1 + \left[ b'(x) \right]^2} \ dx = \int_{x_0}^{x_1} \sqrt{1 + \left[ \omega e^{-\lambda x}(1 - \lambda x) \right]^2} \ dx \leq \int_{x_0}^{x_1} \sqrt{1 + \left[ \omega(1 - \lambda x) \right]^2} \ dx. \tag{15}$$

Then, substituting $u(x) = \omega(1 - \lambda x)$, we obtain

$$\int_{x_0}^{x_1} \sqrt{1 + \left[ \omega(1 - \lambda x) \right]^2} \ dx = \frac{-1}{\omega^2} \int_{\omega(1 - \alpha x_0)}^{\omega(1 - \alpha x_1)} \sqrt{1 + u^2} \ du. \tag{16}$$

The solution of this last integral allows us to conclude that

$$A(R_I') \leq \frac{(y_1 - y_0)}{2\omega \lambda} \left( x \cdot \sqrt{x^2 + 1} + 1 + \log \left( \sqrt{x^2 + 1} + x \right) \right)_{\omega(1 - \lambda x_0)}^{\omega(1 - \lambda x_1)}. \tag{17}$$

With this upper bound, we can find a lower bound for the ratio which in practical scenarios will also allow us to compute an even more conservative $\tau$.

Even if we want to find more precise bounds, we can compute the integrals with numerical methods for this particular instantiation of the function $b$. After some empirical experiments, we found that solving the integrals by using numerical algorithms gives us very precise results where the upper bound of the integration error has values less than $10^{-10}$.

PROOF OF THE LOOP INVARIANT

Here, we show a detailed proof of the loop invariant that appears in the proof Theorem 4.3, Case (b).

**Initialization:** the initialization is trivial because if $\alpha \in \Delta'$, then prior the first iteration it still holds that $\alpha \in \Delta'$.

**Maintenance:** suppose that the previous iteration was such that $\alpha \in \Delta'$ and $\Delta \neq \Delta'$. In Lines 10 and 11, $\Delta \leftarrow \Delta'$, and $\Delta' \leftarrow \emptyset$. This means that $\alpha \in \Delta$. In the for-loop in Line 12, $\alpha$ is reached at some point. There, Line 13 computes a neighborhood $\sigma_\alpha$ of $\alpha$ and tries to find an homography between $d(\sigma_\alpha)$ and $d'(\sigma_\alpha)$. There are two cases:

1. Suppose that the homography is found. According to our assumption about local homographies, $d'(\sigma_\alpha)$ will look like a perfectly flat version of $d(\sigma_\alpha)$, then $|\alpha| > \tau$, and in Line 18 $\alpha$ will be detected as a difference. Hence, $\alpha$ is added to $\Delta'$ (i.e. $\alpha \in \Delta'$). Notice that the for-loop only adds elements to $\Delta'$, which means that at the end of the for-loop it still holds that $\alpha \in \Delta'$. Then the while loop returns to Line 9 to evaluate the condition, so that the invariant holds.

2. Suppose that the homography is not found. In this case, the execution reaches Line 16, where $\alpha \leftarrow d'(\sigma_\alpha)$. Then, the algorithm tries to find a difference between $d'(\sigma_\alpha)$ and $d(\sigma_\alpha)$. The hypothesis of the invariant is that $\alpha$ was found in Line 8, so the difference found in Line 18 should show $\alpha$ as a forgery again, because the algorithm does not modify the paper, but only crops it to a particular region. Then as in the previous case, $\alpha$ will be added to $\Delta'$ and it remains there until the end of the for-loop. Finally, the execution will reach Line 9 again and the invariant holds for this case.

**Finalization:** In the final loop, $\Delta = \Delta'$, and because of the invariant, $\alpha \in \Delta'$ once the exit condition is tested. Then the algorithm will return $\alpha$ as a forgery.