IRIQ AI RESEARCH

# Privacy-First Document Intelligence

## Designing an On-Device Capture, OCR, Enhancement, and Export Pipeline

**Kranthi Chaitanya**

ORCID: 0009-0009-3105-4603

July 22, 2026

*Academic-style architectural research paper*

**Kranthi Chaitanya**

**ORCID:** [0009-0009-3105-4603](https://orcid.org/0009-0009-3105-4603)

**July 22, 2026**

## Abstract

Mobile document applications routinely process identity records, financial statements, contracts, medical forms, and other material whose disclosure can create lasting harm. Conventional cloud-first architectures upload page images before enhancement or recognition, expanding the number of systems, operators, and network paths exposed to document content. This paper develops a privacy-first alternative: an on-device pipeline that performs capture, geometric correction, image enhancement, optical character recognition (OCR), review, and PDF export locally by default. IRIQ Capture is used as an implementation case study, but the contribution is architectural rather than product-specific. The design combines data minimization, explicit user control, least-privilege acquisition, bounded intermediate storage, deterministic transformations, and streaming export. A threat model identifies exposure at acquisition, temporary storage, inference, logging, export, and sharing boundaries. The paper also distinguishes privacy properties from security claims: local computation reduces disclosure opportunities but does not by itself protect a compromised or unlocked device. Because controlled benchmark data are not yet reported, no accuracy, latency, or energy superiority is claimed. Instead, a reproducible evaluation protocol is proposed for measuring character error rate, page-processing latency, peak memory, energy consumption, export fidelity, and unintended network transmission. The result is a practical reference architecture for building useful document intelligence while treating external disclosure as an exception rather than a prerequisite.

**Keywords:** document intelligence; on-device OCR; privacy by design; mobile capture; image enhancement; PDF export; Android

## 1. Introduction

The smartphone has become a general-purpose document scanner. A single capture session may contain tax identifiers, signatures, bank details, addresses, employment information, or health data. The central engineering question is therefore not only whether an application can recognize and export a document, but where each transformation occurs and which parties can observe the underlying pixels and extracted text.

Cloud processing offers elastic compute and rapidly updated models, but it also requires document content to cross a device boundary. Encryption in transit protects a network channel; it does not eliminate collection, server-side processing, retention, operational access, jurisdictional exposure, or configuration risk. A privacy-first architecture starts from a different default: if a task can be completed acceptably on the user's device, the original document should not need to leave that device.

This paper presents a local-first pipeline for mobile capture, enhancement, OCR, review, and PDF export. Its three research questions are:

1. Which architectural controls materially reduce disclosure across the document-processing lifecycle?
2. How can a mobile pipeline remain responsive under constrained memory, compute, storage, and battery budgets?
3. How should the privacy, quality, and efficiency properties of such a system be evaluated without overstating unmeasured results?

The principal contribution is an implementable decomposition of the pipeline and its trust boundaries. The paper does not claim that on-device execution is universally more accurate or secure. Instead, it argues that local processing can reduce the number of disclosure paths when paired with disciplined storage, logging, permission, export, and sharing practices.

# 2. Background and Design Principles

## 2.1 Document intelligence on mobile devices

Document intelligence extends beyond taking a photograph. A usable system must acquire a legible image, detect or accept page boundaries, correct perspective and orientation, improve visual quality, recognize text, retain page order, and create an interoperable output. Each stage can create a new representation of the same sensitive source: a camera frame, cropped bitmap, enhanced bitmap, OCR text graph, thumbnail, temporary file, database record, and exported PDF.

On-device OCR is now technically practical for common scripts. Google's ML Kit Text Recognition v2, for example, performs recognition on the device and returns a hierarchy of blocks, lines, elements, and symbols [1]. Its documentation also emphasizes that focus and sufficient pixel density materially affect recognition quality [2]. These characteristics motivate a pipeline in which capture feedback and deterministic preprocessing occur before recognition rather than relying on post-upload repair.

## 2.2 Privacy by architecture

NIST describes privacy risk as arising from data processing and organizes its Privacy Framework around Identify-P, Govern-P, Control-P, Communicate-P, and Protect-P functions [3]. For a document application, these functions translate into concrete engineering questions: What data representations exist? Why are they needed? Who can access them? How long do they persist? When does content cross a trust boundary? How can a user inspect, export, or delete it?

Android guidance similarly recommends data minimization, permission minimization, transparency, and reduced visibility across applications [4], [5]. These are architectural properties, not statements in a privacy policy. A camera-only workflow should not request broad library access; a photo-import workflow should prefer the system photo picker; and OCR that runs locally should not silently transmit images or recognized text.

This paper adopts six principles:

- **Local by default:** core capture, enhancement, OCR, and export do not require document upload.
- **Purpose limitation:** each representation exists only for a defined stage or user-visible feature.

- **Least privilege:** the app requests the narrowest available input and output access.
- **Bounded persistence:** temporary full-resolution artifacts have explicit ownership and deletion rules.
- **Observable boundaries:** network-assisted or sharing actions are explicit and distinguishable from local processing.
- **Resource proportionality:** processing quality is balanced against memory, latency, energy, and device capability.

# 3. Method

This work uses an architectural case-study method. The pipeline is decomposed into stages, trust boundaries, data representations, and failure modes. Privacy controls are mapped to each stage. Resource behavior is considered at the bitmap lifecycle level, because full-resolution page images typically dominate mobile memory. The analysis is informed by the IRIQ Capture implementation, Android platform guidance, on-device OCR documentation, the NIST Privacy Framework, and archival PDF standards.

No human-subject data or private document corpus is used in this paper. No empirical performance results are presented. Section 8 defines an evaluation protocol intended to make subsequent measurements comparable and falsifiable.

# 4. Threat Model

## 4.1 Assets

Protected assets include original page pixels, enhanced images, OCR text and coordinates, thumbnails, document metadata, page order, exported PDFs, and residual temporary files. Metadata can itself be sensitive: timestamps, filenames, document titles, and page counts may reveal behavior even when content is encrypted.

## 4.2 Adversaries and failure conditions

The design considers:

- passive observation over a network path;
- unintended collection by analytics, crash-reporting, advertising, or third-party SDKs;
- excessive operating-system permissions;
- accidental disclosure through shared storage, previews, logs, backups, or exported files;
- stale temporary data after cancellation, failure, or process termination;
- resource exhaustion that causes crashes, partial exports, or corrupted output;
- application-level defects such as duplicate transformations or orientation errors.

A fully compromised operating system, malicious keyboard, physical access to an unlocked device, and deliberate user sharing are outside the protection boundary. Local execution reduces remote disclosure opportunities but cannot guarantee confidentiality on a compromised endpoint.

### 4.3 Trust boundaries

The primary boundary encloses the application sandbox and on-device inference. Four crossings require special treatment: acquisition from camera or system picker, optional model delivery, export into user-selected storage, and explicit sharing to another application or service. Any telemetry channel is a fifth boundary and must exclude document pixels, OCR text, filenames, and content-derived identifiers by design.

## 5. Reference Architecture

**ACQUIRE → CORRECT → ENHANCE → OCR → REVIEW → EXPORT**

**Figure 1.** Local-first document-processing pipeline. The network is not a required stage of the core path.

### 5.1 Acquisition

The camera path should bind access to the user-initiated capture session. Import should use a system picker where available so the user grants access to selected items rather than the entire media library. Android's permission model and privacy guidance support minimizing requested access and requesting it in context [4], [6].

Acquisition also performs early validation: image decoding, dimension and format checks, orientation normalization, and rejection of inputs that exceed safe resource limits. Thumbnails should be derived from the selected source, not treated as authoritative export pixels. The full-resolution source remains immutable until the user commits an edit or the pipeline stores a new canonical representation.

### 5.2 Geometry and orientation

Perspective correction maps an observed quadrilateral to a rectangular page. Let source points be \(p_i=(x_i,y_i)\) and destination points be \(p'_i=(u_i,v_i)\). A planar homography \(H\) satisfies

s\begin{bmatrix}u_i\\v_i\\1\end{bmatrix}=H\begin{bmatrix}x_i\\y_i\\1\end{bmatrix},

where \(s\) is a scale factor. The transform should be computed from validated, non-self-intersecting corner points, with output dimensions capped before allocation.

Rotation requires one source of truth. If pixels are physically rotated, downstream PDF code must not reapply a stored rotation as a second transform. This invariant prevents a class of double-rotation defects and ensures that displayed dimensions, thumbnail aspect ratio, OCR coordinates, and export geometry refer to the same canonical orientation.

### 5.3 Deterministic enhancement

Enhancement aims to improve legibility without altering semantic content. A conservative chain can include luminance normalization, contrast adjustment, grayscale conversion, mild sharpening, and optional thresholding. For an input intensity \(I\), a simple contrast transform is

I' = \operatorname{clip}(\alpha(I-\mu)+\mu+\beta),

where \(\alpha\) controls contrast, \(\beta\) controls brightness, and \(\mu\) is a reference luminance. Sharpening may use unsharp masking,

I_{sharp}=I+k(I-G_\sigma * I),

with conservative gain \(k\) and Gaussian blur \(G_\sigma\).

Enhancement should be non-destructive or reversible until committed. A before/after preview, bounded controls, and explicit reset allow the user to detect clipped highlights, amplified noise, or lost faint text. “Auto enhance” should be understood as a deterministic preset unless a learned model is actually used and disclosed.

## 5.4 On-device OCR

OCR consumes the canonical working image and returns text with structural coordinates. The output should be associated with the exact image revision used for recognition. If crop, perspective, rotation, or enhancement changes afterward, OCR is stale and must be invalidated or recomputed.

The OCR boundary must also be explicit at the dependency level. Some SDK configurations bundle models; others download them dynamically. ML Kit documents both bundled and unbundled installation choices, which trade application size against first-use availability [2]. Dynamic model delivery is not equivalent to uploading the document, but it is still network activity that should be tested, documented, and separated from content transmission.

Recognition failure must not be reported as proof that an image contains no text. Blur, low resolution, handwriting, unsupported scripts, glare, skew, and model availability can all produce empty or incomplete output. The interface should distinguish “no text detected” from capture-quality problems, unavailable recognition, and processing errors.

## 5.5 Page model and review

The document model should track stable page identifiers, canonical image locations, pixel dimensions, revision state, OCR state, and ordering. Derived thumbnails are caches and may be deleted and regenerated. Full-resolution page order and edits should be committed transactionally so that a crash cannot silently reorder pages or point metadata at a missing image.

Review is a privacy control as well as a usability feature. It allows the user to remove unintended pages, inspect crop boundaries, verify orientation, and detect sensitive material before export or sharing.

## 5.6 Streaming PDF export

A naïve exporter decodes every page before writing the PDF. If each bitmap occupies approximately \(w \times h \times b\) bytes, then retaining \(n\) pages yields an image-memory term near

M_{naive}\approx nwhb+M_{encoder}+M_{runtime}.

A streaming exporter processes one page at a time:

M_{stream}\approx \max_i(w_i h_i b)+M_{encoder}+M_{runtime}.

The practical sequence is decode one page, apply the committed transformation, write the page, release or recycle the bitmap, and then continue. This bounds the number of resident full-resolution images and reduces the chance of memory pressure increasing with document length.

Export must preserve aspect ratio, orientation, page order, and predictable margins. Long-term archival requirements may justify a PDF/A profile. ISO 19005 defines constrained uses of PDF intended to preserve the static visual representation of page-based documents over time [7], [8]. Ordinary PDF output should not be labeled PDF/A unless it is actually produced and validated against the selected conformance level.

### 5.7 Save, share, and deletion

Saving should use an operating-system document destination or scoped media interface that gives the user control over location. Sharing is a deliberate boundary crossing and should occur only after export. Temporary share URIs should be time-bounded where the platform supports it.

Deletion semantics must cover the page source, committed edit, thumbnail, OCR record, export cache, and abandoned temporary files. A cleanup routine should run on successful completion, cancellation, recoverable startup, and explicit deletion. Backup behavior should be reviewed separately; an application sandbox can still be copied by platform backup unless configuration and user expectations align.

## 6. Privacy and Security Control Matrix

| Stage | Principal risk | Design control | Verification evidence |
|---|---|---|---|
| Acquisition | Excessive media access | Camera session or system picker; contextual permission | Manifest review and permission test |
| Preprocessing | Residual full-resolution copies | Canonical ownership and bounded temporary files | Storage inspection after success/failure |
| OCR | Silent content transmission | On-device recognizer; SDK/network inventory | Offline functional test and traffic capture |
| Logging | Pixels or recognized text in telemetry | Content-free structured errors | Log and crash-report audit |
| Review | Export of unintended pages | Page-level preview, delete, reorder, orientation check | Interaction and accessibility test |
| Export | Memory exhaustion or incomplete file | One-page-at-a-time decode/write/release | Peak-memory trace and failure injection |
| Share | Unintended third-party disclosure | Explicit user action and scoped URI | Intent/URI permission test |
| Deletion | Stale derivatives | Cascading deletion and startup cleanup | File/database reconciliation test |

The matrix demonstrates why “OCR runs locally” is necessary but insufficient. Privacy depends on the entire lifecycle, including dependencies and failure paths.

# 7. Engineering Invariants

The following invariants make the architecture testable:

4. Core document processing succeeds with network connectivity disabled after any required model is available.
5. No document pixel, OCR string, filename, or content-derived fingerprint enters analytics, advertising, or crash telemetry.
6. At most one full-resolution page bitmap is owned by the streaming export caller at a time, excluding bounded encoder internals.
7. Canonical pixel orientation, stored dimensions, OCR coordinates, preview aspect ratio, and PDF orientation agree.
8. Every temporary artifact has a named owner and a cleanup path for success, cancellation, exception, and process restart.
9. A stale OCR result is never represented as current after an image revision.
10. External save and share operations require explicit user intent.

These invariants can be enforced through unit tests, instrumentation, dependency inspection, static checks, and controlled fault injection.

# 8. Evaluation Protocol

## 8.1 Dataset

A future evaluation should use a consented, de-identified corpus stratified by document type, script, lighting, resolution, skew, perspective, glare, background clutter, font size, and page count. Synthetic pages may supplement but should not replace real capture conditions. Ground-truth text must be independently verified.

## 8.2 Quality metrics

OCR quality should report character error rate (CER) and word error rate (WER):

CER=\frac{S+D+I}{N}, \qquad WER=\frac{S_w+D_w+I_w}{N_w},

where substitutions, deletions, and insertions are measured against ground truth. Geometry can be evaluated using corner error or intersection-over-union between corrected page masks and annotations. Export fidelity should compare page count, order, aspect ratio, orientation, and render similarity.

## 8.3 Systems metrics

Measurements should include median, 95th percentile, and worst-observed page latency; peak proportional set size or equivalent process memory; temporary and final storage; energy per page; export completion rate; and time-to-first-usable OCR. Results should be stratified by device class and thermal state rather than aggregated into a single headline number.

### 8.4 Privacy validation

Privacy tests should combine:

- offline completion of every core workflow;
- packet capture under first run, OCR, export, failure, and background/foreground transitions;
- dependency and manifest review;
- inspection of logs, crash payloads, backups, caches, and shared storage;
- deletion verification after normal completion, cancellation, forced termination, and restart.

The strongest claim supported by these tests is bounded: no unintended transmission was observed under specified test conditions. It is not proof of the absence of every possible disclosure.

### 8.5 Comparative baseline

An optional cloud baseline should use the same source images and report both task quality and exposure-relevant properties: transmitted bytes, retention configuration, external processors, failure behavior, and connectivity dependence. Comparison should not treat privacy as a single numerical score without publishing the weighting model.

## 9. Discussion

### 9.1 Privacy-performance trade-offs

On-device computation removes a mandatory content-upload step, supports offline operation, and can reduce network latency. It also inherits heterogeneous device performance, thermal limits, model-size constraints, and slower improvement cycles when models are bundled. Cloud processing may outperform local models for difficult handwriting, complex layouts, or very low-quality images. A privacy-first product can therefore offer optional network-assisted features, but they should be separately named, consented, scoped, and disabled by default for the core workflow.

### 9.2 Accuracy and enhancement

More enhancement is not always better. Aggressive thresholding can erase faint strokes; sharpening can amplify compression artifacts; and automatic crops can remove marginal notes. OCR accuracy depends on capture quality and script support, not merely preprocessing intensity. The pipeline should preserve a recoverable source and make semantic changes visible to the user.

### 9.3 Privacy is not synonymous with local storage

Local-only marketing can obscure risks from screenshots, backups, logs, notification previews, shared exports, rooted devices, and third-party SDK behavior. Privacy claims should be attached to testable data flows and versioned implementations. Security controls such as encryption, application sandboxing, secure update practices, and dependency management remain necessary.

## 10. Limitations

This is a design paper and implementation case study, not a controlled performance study. It does not report measured OCR accuracy, latency, memory, energy, or comparative cloud results. The architecture is Android-oriented, although its principles generalize to other mobile platforms. Hardware-backed encryption, compromised-device resistance, legal compliance across jurisdictions, accessibility outcomes, multilingual benchmark coverage, and formal PDF/A validation require separate study. The specific behavior of third-party components may change across versions and must be re-audited during releases.

## 11. Conclusion

A document scanner's privacy posture is determined by its complete data lifecycle, not by a single OCR choice. The architecture presented here makes local execution the default for capture, correction, enhancement, recognition, review, and export; minimizes permissions and persistent derivatives; bounds full-resolution memory through streaming export; and treats saving, sharing, telemetry, and optional network assistance as explicit trust-boundary crossings.

The approach does not eliminate endpoint risk or guarantee better recognition than cloud systems. Its value is more precise: it reduces the number of parties and systems that must receive document content for the core workflow to function. By pairing that architecture with falsifiable invariants and a reproducible evaluation protocol, teams can make privacy claims that are narrower, clearer, and more defensible.

## References

[1] Google, "Text recognition v2," *ML Kit*. https://developers.google.com/ml-kit/vision/text-recognition/v2 (accessed July 22, 2026).

[2] Google, "Recognize text in images with ML Kit on Android," *ML Kit*. https://developers.google.com/ml-kit/vision/text-recognition/v2/android (accessed July 22, 2026).

[3] National Institute of Standards and Technology, *NIST Privacy Framework: A Tool for Improving Privacy through Enterprise Risk Management, Version 1.0*, Jan. 2020. https://www.nist.gov/system/files/documents/2020/01/16/NIST%20Privacy%20Framework_V1.0.pdf

[4] Android Developers, "Privacy," Google. https://developer.android.com/privacy (accessed July 22, 2026).

[5] Android Developers, "Design for Safety," Google. https://developer.android.com/quality/privacy-and-security (accessed July 22, 2026).

[6] Android Developers, "Permissions on Android," Google. https://developer.android.com/guide/topics/permissions/overview (accessed July 22, 2026).

[7] International Organization for Standardization, *ISO 19005-1:2005: Document management—Electronic document file format for long-term preservation—Part 1: Use of PDF 1.4 (PDF/A-1)*, 2005. https://www.iso.org/standard/38920.html

[8] International Organization for Standardization, *ISO 19005-4:2020: Document management—Electronic document file format for long-term preservation—Part 4: Use of ISO 32000-2 (PDF/A-4)*, 2020. https://www.iso.org/standard/71832.html