Package ‘pdfsigner’

June 26, 2026

Title Digitally Sign and Verify PDF Documents

Type Package

Version 0.2.4

Description Digitally sign PDF documents with a 'PKCS#12' keystore and verify their signatures. Signing produces a detached 'PKCS#7' / 'CMS' signature ('adbe.pkcs7.detached') over the document and is applied as an incremental update, so existing signatures remain valid. The cryptography and PDF manipulation are performed by a bundled, pure-'Rust' backend (the 'pdf_signer' crate); no Java runtime, 'OpenSSL', or external command-line tools are required. Visible signature appearances with custom text are supported.

Depends R (>= 4.2)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

License GPL-3

Copyright The pdfsigner package and its bundled 'pdf_signer' Rust backend are released under GPL-3. The build vendors the source of third-party Rust crates (src/rust/vendor.tar.xz) that remain under the copyright of their respective authors and are licensed under permissive terms (MIT, Apache-2.0, BSD, ISC, Zlib, Unicode-3.0, 0BSD, Unlicense or CDLA-Permissive-2.0); see the inst/AUTHORS file for the full list.

Encoding UTF-8

URL https://github.com/StrategicProjects/pdfsigner

BugReports https://github.com/StrategicProjects/pdfsigner/issues

SystemRequirements Cargo (Rust's package manager), rustc

Config/rextendr/version 0.5.0

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Andre Leite [aut, cre],
Hugo Vasconcelos [aut],
Diogo Bezerra [aut],
Authors of the vendored Rust crates [ctb, cph] (see inst/AUTHORS for the bundled crates and their licences)

Maintainer Andre Leite <leite@castlab.org>

Repository CRAN

Date/Publication 2026-06-26 15:30:02 UTC

Contents

`sign_pdf` *Digitally sign a PDF document*

Description

Signs `pdf_file` using an RSA key + certificate stored in a PKCS#12 (`.p12`/`.pfx`) keystore, writing the signed document to `output_file`. The signature is a detached `adbe.pkcs7.detached` CMS over the whole document and is added as an incremental update, so any pre-existing signatures stay valid.

Usage

```
sign_pdf(
  pdf_file,
  output_file,
  keystore_path = Sys.getenv("KEYSTORE_PATH"),
  keystore_password = Sys.getenv("KEY_PASSWORD"),
  signtext = NULL,
  validate_link = NULL,
  reason = NULL,
  signer_name = NULL,
  page = 1,
  x = 36,
  y = 36,
  width = 320,
  height = 64,
  font_size = 8,
  font = NULL,
  image = NULL,
  border = TRUE,
```

```
  translate = FALSE,
  tsa_url = NULL,
  pades_level = c("bb", "bt", "blt", "blta")
)
```

Arguments

pdf_file	Path to the input PDF.
output_file	Path where the signed PDF is written.
keystore_path	Path to the .p12/.pfx keystore. Defaults to the KEYSTORE_PATH environment variable.
keystore_password	Password for the keystore. Defaults to the KEY_PASSWORD environment variable.
signtext	Optional text for a *visible* signature box. When NULL or empty the signature is invisible.
validate_link	Optional validation URL appended to the visible box.
reason, signer_name	Optional /Reason and /Name for the signature dictionary.
page	1-based page number for the visible box.
x, y, width, height	Visible box geometry, in PDF points (origin at the page's bottom-left).
font_size	Font size for the visible box, in points.
font	Optional path to a TrueType/OpenType font file (.ttf/.otf) to embed in the visible box. When NULL, the standard Helvetica is used. Only the WinAnsi (Latin-1) glyph range is embedded. Ignored for invisible signatures.
image	Optional path to a PNG or JPEG logo drawn in the visible box. Ignored for invisible signatures.
border	Draw a border around the visible box.
translate	If TRUE, the date label in the visible box is in Portuguese; otherwise English.
tsa_url	Optional RFC 3161 Time-Stamping Authority http:// URL. Required for pades_level "bt" and above. Requires network access.
pades_level	PAdES conformance level: "bb" (baseline, default), "bt" (+ signature timestamp), "blt" (+ DSS with certificates and CRLs), or "blta" (+ a document timestamp over the whole file). Levels "bt" and above need tsa_url.

Value

Invisibly, the path to the signed PDF. Raises an error on failure.

Examples

```
## Not run:
sign_pdf(
  pdf_file = "input.pdf",
  output_file = "signed.pdf",
```

```
  keystore_path = "keystore.p12",
  keystore_password = "password",
  signtext = "Document digitally signed by CastLab",
  validate_link = "https://castlab.org/validate",
  translate = TRUE
)

## End(Not run)
```

`verify_pdf_signature` *Verify the digital signatures of a PDF*

Description

Cryptographically verifies every signature in `pdf_file` using the bundled Rust backend. Each signature is checked by re-deriving its signed byte range, confirming the `messageDigest` against `SHA-256` of the content and validating the signer's RSA signature over the signed attributes.

Usage

```
verify_pdf_signature(pdf_file, roots = NULL)
```

Arguments

`pdf_file`	Path to the PDF to verify.
`roots`	Optional path to a PEM file of trusted root certificates (e.g. the ICP-Brasil AC Raiz set). When supplied, each signer certificate chain is validated against these roots and reported in `chain_trusted`.

Value

A list with one entry per signature. Each entry is a named list with `valid` (logical), `signer` (subject DN), `chain_trusted` (logical or `NA` when no `roots` given), `covers_whole_document` (logical), `signed_len` (bytes), `byte_range` (numeric length-4) and `detail`. A length-zero list means no signatures were found.

Examples

```
## Not run:
result <- verify_pdf_signature("signed.pdf", roots = "icp-brasil-roots.pem")
vapply(result, function(s) s$valid, logical(1))

## End(Not run)
```

Index