PDF Days Online 2021

Cryptography in PDF: future perspectives

Matthias Valvekens

2021–09–28

Setting up

About me

- Background:
  - Mathematician by trade
  - FOSS developer/tinkerer
  - Long-time cryptography enthusiast
- Research engineer at iText since 2020
- Active in various working groups, both within ISO TC171 SC2 and the PDFa

Purpose of this talk

- Cryptography best practices evolve over time
- PDF has made use of cryptography for over two decades:
  - Encrypting documents (confidentiality)
  - Digital signing (establish authenticity / assent / existence / ...)
- Need to keep an eye on advances in cryptography engineering, look for ways to innovate

Purpose of this talk (cont.)

Goals for today:

- look at innovation that’s already underway;
- take a step back and think outside the box;
- plant some seeds for later discussion.

Disclaimer

Disclaimer: This talk contains *opinions*

Structure of the talk

- Roughly two parts:
  - Encryption in PDF
  - Digital signatures in PDF
- For each of those, we'll
  - have a quick look at current standardisation work;
  - take a step back and brainstorm about what else we can improve on.

Encryption in PDF

PDF encryption has come a long way…

Improvements in ISO 32000-2 (PDF 2.0):

- all RC4 usage deprecated
- file encryption key generated independently from passwords
- promote use of AES-256

⇝ Confidentiality of the data at rest is up to snuff.

…but there are still issues

AES–CBC is malleable:

- CBC = Cipher Block Chaining
- Only provides confidentiality, no authentication
- Exploitability research: Ruhr University Bochum, 2019
  - Known plaintext in **Perms** provides content injection vector
  - Arbitrary content can be injected without knowing any passwords
  - Can also be used to mount exfiltration attacks in some viewers

Consensus in cryptography field:

$\rightsquigarrow$ confidentiality without authentication is seldom meaningful.

…but there are still issues (cont.)

Lots of homegrown algorithms and wheel-reinvention:

- Key derivation/retrieval:
  - Convoluted and hard to analyse rigorously
  - New key retrieval procedure in PDF 2.0 is still very baroque
  - Should use battle-tested, industry-standard techniques in future revisions

What about solutions?

- Current efforts through PDF extensions in the pipeline: ISO 32003, ISO 32004
- Some ideas for the future

ISO 32003: new crypt filter for AES–GCM in PDF

AES–GCM has several advantages over AES–CBC:

- AES–GCM ciphertext is authenticated
- Supports fast random access
- The diversity in options is probably a good thing

But it’s not a panacea:

- Nonce/IV reuse is catastrophic in GCM, less so with CBC
- Authentication applies only to strings and stream content
  - ⇝ separate integrity protection is still needed

ISO 32004: MAC-based integrity protection

Goal of ISO 32004 is to address the authentication issue:

- MAC = Message Authentication Code
- Support adding a MAC covering the entire document
- Generating/validating MAC requires knowing the file encryption key
- Implementation similar to digital signature:
  - Compute digest over **ByteRange**
  - Can also be combined with digital signatures

Ideas for a new security handler...

- Password-based encryption: encrypt file encryption key with key derived from password via PBKDF2/Argon2/`scrypt`.
- Phase out **Perms** entry in encryption dictionary, authenticate permissions through other means (insofar as that is meaningful)
- Unify and modernise public-key and standard security handlers.

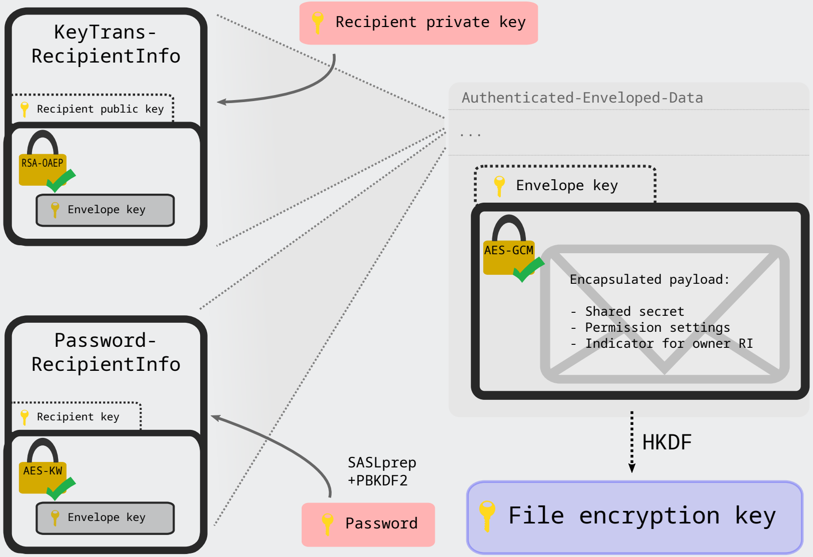
⇝ Standard data containers for these purposes already exist

Unification strategy

Leverage `Authenticated-Enveloped-Data` (RFC 5083)

Rough idea: mimic **Recipients** structure of PubSec handler:

- Authenticated/encrypted envelope containing
  - Shared secret
  - Permission settings
- Encrypt envelope key separately for each “recipient”
  - PKI-based: encrypt with recipient’s public key
  - Password-based: encrypt using key derived from password
  - One recipient is designated as the owner
- Note: still no technical barrier to changing permissions with “user-level” access


Recipient private key
KeyTrans-RecipientInfo
Recipient public key
RSA-OAEP
Envelope key
Authenticated-Enveloped-Data
...
Envelope key
AES-GCM
Encapsulated payload:
- Shared secret
- Permission settings
- Indicator for owner RI
Password-RecipientInfo
Recipient key
AES-KW
Envelope key
SASLprep
+PBKDF2
Password
HKDF
File encryption key

Digital signatures in PDF

Digital signing in PDF: a success story

- Over 2 decades of history in PDF, but more relevant now than ever!
- Updated multiple times over the years
- Widely known among the general public by now
- Part of government & industry workflows alike
- Integration with other standards: PDF + CAdES ⇝ PAdES

Shiny new things in the ISO pipeline

Some upcoming PDF extensions:

- ISO 32001: add support for SHA-3 family of digests
- ISO 32002: significant improvements to ECC support:
  - Spell out supported ECDSA curve-hash combinations
  - Add support for EdDSA (!!): Ed25519 and Ed448
  - This is a big deal: EdDSA is much harder to get wrong
- Updating our cryptographic primitives is a good thing, and we should keep that up.

Go forth and implement these (once published)!

Another look at the fundamentals

- PDF signing is an oddball in the DigSig landscape
  - “signature-in-data”: the signature is **embedded into** the data being signed
  - much more common: signature **envelops** signed data, or is entirely **detached** from it

- For modern CMS-based digital signatures, this is suboptimal:
  - Signature container size is hard to predict exactly
  - Requires special file writing logic to produce
  - Multi-sig workflows become complex
  - ...

Increases in complexity

These days, a signature almost never comes alone:

- Revocation information
- Signature/content/document timestamps
- Multiple signers?
- DocMDP / FieldMDP / ...
- ...

Increases in complexity: consequences

Modern PDF signatures are difficult to handle robustly

- Multi-signer workflows require specialised processing
- Signatures must be serial.
- Validation is hard
  - Biggest PDF-specific issue: incremental update analysis
  - Interoperability?
  - Especially with multi-signer / DocTimeStamp / PAdES-LT(A)

What other options do we have?

Should we keep relying on the “signature-in-data” model exclusively?

Technical debt:

- Even assuming airtight standards, the validation complexity is still massive
- Handling these signatures correctly will only get harder

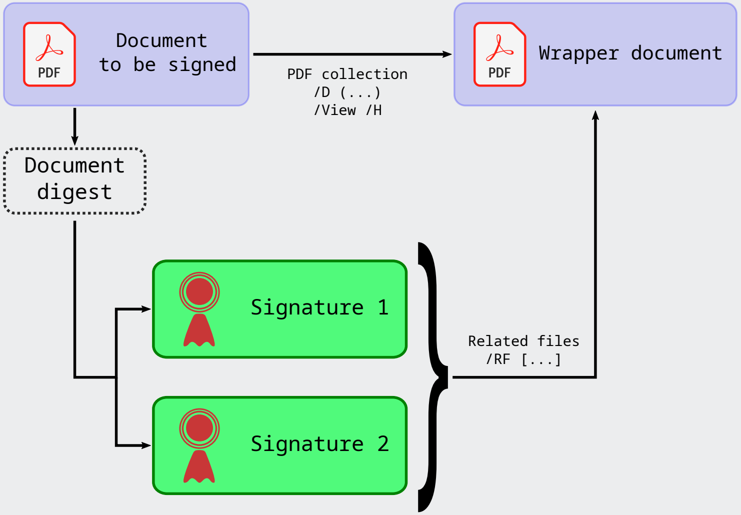
But, the genie is out of the bottle:

- We can’t just scrap & forget the current way of doing things
- What *are* the alternatives?

Tools we have in ISO 32000-2

- Wrapper document pattern
  - Main document content in embedded file using the Collections mechanism
  - Precedent: unencrypted document wrapper pattern
  - Also came up in the DigSig TWG to solve a related problem (notarization signatures)
- Cryptographic Message Syntax (AKA new-style PKCS #7): still going strong!
  - Virtually everyone uses CMS-based signatures in PDF nowadays
  - We can get much more mileage out of that than we currently do.
  - Can we make more direct use of CAdES- and CAdES-X-based standards (+implementations)?
- Things like related files (`/RF ...`) to link signatures to files

⇝ We can rearrange those puzzle pieces to build something that works


PDF
Document to be signed
PDF collection
/D (...)
/View /H
PDF
Wrapper document
Document digest
Signature 1
Signature 2
Related files
/RF [...]

Advantages of such a scheme

Core benefit: maximal decoupling of document data & signature data

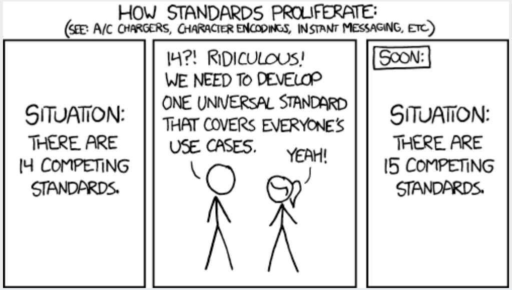
- Can lean on existing cryptographic libraries & standards even more
- Lower barrier to entry
  - Original document is digested completely (i.e. no **ByteRange**)
  - Existing signers & validators can adapt fairly easily
  - New players have an easier time getting off the ground.
- Parallel signing becomes trivial
- Signature container can expand in size as necessary
- Can do LTV signatures without incremental updates
- …

…but of course there's a price to pay

TANSTAAFL:

- Burden for non-DigSig PDF consumers: not everyone supports Collections
- What about the connection between signatures and forms?
- Signature appearances are also tricky
- …and there are undoubtedly many more pitfalls that I didn’t think of.

In closing


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Source: XKCD

- Coming up with new ideas is the easy part…
- …actually developing them requires industry-wide effort by many, many people.