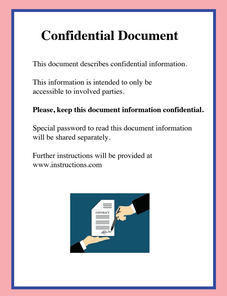
PDF Days Europe 2025

Integrity Protection via Message Authentification Code

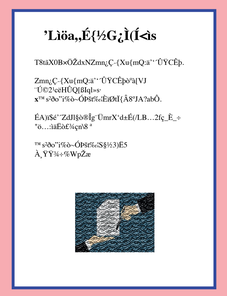
itext by apryse

Problem

Usual document

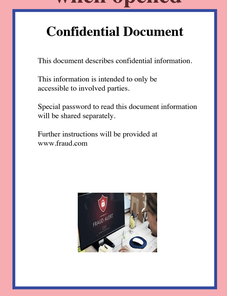


Encrypted document



Tampered document

Tampered document when opened



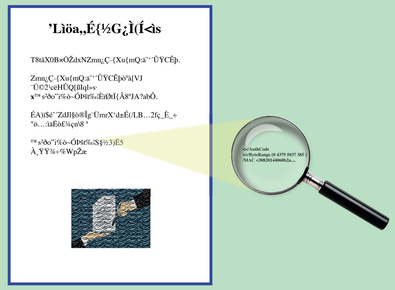
What can sufficiently intelligent attacker do, to and encrypted document, without knowing the password:

- Delete any encrypted content.
- Insert new objects, by marking them as not to be encrypted through the use of *Identity* crypt filter.
- Modify cross-reference table in any imaginable way.
- See and modify any content, besides *string* and *stream* objects.

What happens from a user perspective:

1. User gets tampered document.
2. Enters user password to see the content.
3. Veiwer shows the document without any notifications, there is no way to identify possibly commited fraud.
4. User sees severly compromised document, while being completely sure, that everything should be fine.

Solution



- MAC Protects the integrity of an encrypted PDF document by embedding signed MAC token into the document's trailer.
- The token is generated by signing complete document content.
- Encrpytion key is used to perform token generation. No other secret is required.
- MAC is backwards compatible, processors which don't support this feature can still read the content
- MAC can be integrated together with digital signatures

How does it work?

The workflow is pretty much the same, as with digital signatures. Document content is hashed and signed using the encryption key (usually password). Signed token is placed inside of a document, in a place prepared in advance. Whenever someone opens the document, the token generation process is repeated by the viewer and the token is compared with the one, stored in the document. If the values are different, document was tampered and shouldn't be rusted If the content is modified legitimately, MAC token is regenerated and replaced, but this is only possible if encryption key or password is provided.

Why can't it just be removed?

What stops sufficiently intelligent attacker from removing MAC token from the document, so that it looks like an encrypted document without MAC support? When MAC feature is enabled, responsible processor adds special permission bit to the encryption dictionary. Those permission bits are encrypted, so there is no way for an attacker to modify them. If the permission bit is present, but MAC is stripped from a document, a viewer is expected to notify a user, that document might be tampered.

Potential difficulties

What happens, if document revision doesn't contain MAC?

If document contains a revision, in which content is already encrypted, but MAC is not yet there, the security may be compromised.
The problem here is that an attacker can replace permission bits in a MAC protected revision by those,
which can be found in the revision, which is not MAC protected.

What if user password is not secure enough?

Encryption security strongly depends on password security. If user password is empty or not secure enough,
the encryption can easilly be broken. MAC protection is not an exception here.
It also strongly depends on the secureness of user password.
So, what should a responsible processor do, if user password is left empty?

Enabling MAC by default, yes or no?

Obviously, MAC feature provides extra level of protection for encrypted PDF documents, so why not to always enable it?
Well, there are some downsides, first of all, MAC is not yet widely supported, not many viewers are capable of validating MAC.
Secondly, MAC generation requires complete document to be stored in memory, before MAC token is generated.
This increases memory usage significantly, which might be a problem for ginormous documents.

What should happen, if byte range doesn't cover complete document?

Similarly to digital signatures, MAC feature uses byte range object, which determines the content to be covered with MAC token.
When MAC token is generated, complete document is expected to be covered with byte range.
However, if this document was modified by a processor, which doesn't support MAC, MAC may became "stale".
Byte range won't cover whole document anymore Even though this sounds completely valid,
there is no way to distinguish this scenario with the one, in which an attacker maliciously modifies the document.