Document Encryption and DRM Security

An overview of document encryption and why DRM is needed for document control

What is document encryption?

Document encryption is the process by which documents are protected with cryptographic keys (a password, public key, token, etc.) so that only individuals with the corresponding decryption keys (the same password, private key, token, etc.) can open them. It is used to protect documents in transit (i.e. sent via email) and at rest (i.e. stored on a disk or in the cloud) from being accessed by unauthorized users.

Document encryption programs can either be standalone applications (like PGP, which encrypts any files and not just documents) or plugins to certain applications like MS-Word or Adobe Acrobat which enable documents to be encrypted when they are saved to disk.

Not all document encryption is equal. Whilst most document encryption programs provide NIST approved AES 256 bit encryption they vary with their protection methods – some provide password protection, others use hardware tokens, whereas others provide public key technology (PKI). Each scheme has its own pros and cons – passwords can be given away (and are often weak and therefore easily broken), tokens have to be distributed and maintained, and public key technology has to have a management hierarchy.

Why encrypt documents?

According to some politicians, if you have nothing to be worried about then you don’t need encryption! Strange. Historically there were five groups of people who wanted encryption:

- Purveyors (traders)
- Princes (the military)
- Politicians (diplomats)
- Paramours (lovers)
- Popes

Unlikely though it may seem, purveyors come out top of the list. The earliest preserved writings were sales manifests on baked clay tablets, and early cryptographic documents included a recipe for pottery glaze. So maybe trade is the most important thing of all? Consider – the famous
Enigma machine was first used in commerce before being taken up by the military. So it is important to preserve commercial activity, and there is a lot more of that than all the others. It’s just so much less exciting.

So maybe you don’t need a lot of imagination to figure out that each group must have had information they want to share, but only with carefully selected people, and not always on the same basis for each recipient. And you can also figure out that they would want to be sure that the information did not get in the wrong hands, was not changed or misrepresented, and maybe could be denied if necessary. And that is just what document encryption provides.

Encryption is the most valuable tool supporting our commonest security requirements for Internet based transactions. SSL encryption is ubiquitous for protecting information as it races round the networks from tablet to bank to merchant, or to anywhere else for that matter. Because of that Internet trade is underpinned by something that makes it all tick.

Document encryption isn’t just useful – it’s the essential technology for securing documents in today’s society on the Internet.

But here’s the rub.

**Document encryption – the downside**

Encryption works just fine if you can maintain an encryption key register – just like you do an email address book. That’s easy to say but evidently not so easy to do, otherwise the email suppliers would have cracked this one years ago.

There are two big problems with encryption that are difficult to solve. The first is that you have to have all the keys of all the people you might want to send secret information to. And the second is that when they get the secret information from you they can do just what they want with it.

So document encryption works splendidly protecting documents in transit and at rest (stored on a device). The issue however is that the recipient can give the Crown Jewels to the enemy (either the encrypted document with the corresponding decryption key, or the decrypted document) and you can’t do anything about it. So document encryption is what you have to start from, but it needs more functionality if it’s going to give you full document control.

As I mentioned earlier, there can be a big management problem looking after and protecting all the cryptographic keys involved, unless you have a management system that is based upon both users and documents, and that manages all the keys secretly – actively preventing the users
from getting involved in (and messing up?) a complicated system that can all too easily go wrong.

So although we need document encryption to protect a document while it travels around the network or sits on a hard drive somewhere, we also need ways of controlling what can be done with the content once the document has been opened – and that can be achieved with Digital Rights Management (DRM).

**Document DRM – the next step for document control**

DRM has to be both available to the application using the document, and implemented in it by design so that weaknesses are not exposed.

Computer applications (apart from those doing encryption) were usually not designed with document security control in mind – mostly they were designed to get to market as rapidly as possible and making copying documents quick and easy. Another consideration was speed of operation. Complex application packages are large and use a lot of processing, usually rendering images onto the screen or manipulating large data in memory or on disk (where they would make uncontrolled copies of the documents to speed up performance).

This does not help document DRM providers because they are obliged to either find some way of bootstrapping themselves into the applications, or using Systems Development Kits (SDKs) and handling the business functionality themselves.

Naturally, both methodologies have been used.

**Bootstrapping yourself into someone’s application works in one of two ways.** You can either reverse engineer into the application (difficult and may cause litigation in some jurisdictions) or, if the manufacturer allows, plug-in to their functionality (usually requires a licensing deal that can be quite complex).

Both of these approaches present problems. If you have reverse engineered into an application and the applications provider changes anything, what you are doing will probably cease working immediately. The same applies to plug-ins, where the locations will likely change, again meaning that everything stops working.

There are also problems for these types of implementations if document locations within the data storage of the application change, and it can be difficult to maintain security/integrity if
new unencrypted copies of the document are created by the application, even as temporary files.

A more secure approach to implementing DRM controls into applications is to develop a custom DRM application using appropriate Software Development Kits (SDK). There are many reasons why this is a more secure and reliable approach.

If you plug in to another architecture you face the problem of trying to prevent users accessing commands. For instance, in Microsoft Word, after highlighting some text you can copy it using Alt e c or Ctrl C or using the appropriate mouse clicks, and you can SaveAs a document using Alt F A as well as using the toolbar. Using an SDK allows the DRM provider to determine what functionality they are going to support and how it is being implemented. So the DRM provider can be sure that there are no unencrypted copies of files created or left on hard drives, and that memory used for decryption is not transferred to a swap file, and how text and images are rendered to the screen (some systems send text as text and others as graphics – which is more difficult to copy back to text).

Another advantage of using an SDK is improved stability. Product manufacturers are continuously making changes to their products, and according to Gilb’s hypothesis (on unreliability), every time a bug is fixed the fix creates one new bug that can be found and one that likely cannot. From a security standpoint, the more changes that are made, the more new bugs are introduced. So unless there is a security reason to make a change (say it stops working) it is more reliable for the users not to have changes, and if you must have changes alter the smallest system where you will do the least damage. This may sound negative, but consider: the PGP encryption system is well known and widely implemented, and very stable and reliable. It is also very small, with limited functionality. And that is why it is so successful.

A summary of document encryption and DRM

So document encryption (preferably using a strong and reliable method such as that used by PGP) is a critical tool for protecting document contents. But it needs to be allied to a DRM system that can enforce a limited set of controls to allow document owners to specify what is allowed once the content is decrypted – note that document decryption should only ever occur in memory so users cannot grab content from temporary files on disk. Document DRM applications should also lock decryption keys to devices so they cannot be readily shared – note that this cannot be achieved if the system uses passwords that are known to the user (since these can be shared with other users).
A document DRM system must itself be compact and reliable if it is going to deliver the expected results and not be affected every time a major applications manufacturer makes changes. So document encryption is the essential building block for document security, but successfully implemented DRM technologies are required to integrate with document encryption and application functionality.

About LockLizard

LockLizard is a DRM (digital rights management) company that specializes in document security and copy protection for PDF files, ebooks, and elearning content. We protect information with US Government strength encryption and rights management controls to ensure complete protection against copyright piracy. Use our DRM software to control document use - stop copying, prevent printing, disable print screen, expire content, instantly revoke access to information, and track document usage (views and prints).

http://www.locklizard.com