Sanford Bingham
Founder & CEO
FileOpen Systems Inc.
October 6, 2016

Microsoft RMS: Standard or Stranglehold?

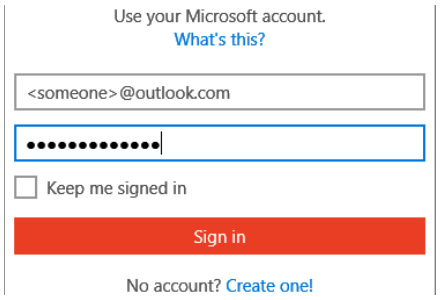
Cloud security, identity management, and the future of the open Internet

The World Wide Web is built upon standards, at every level. Because of Web standards, users can access information from multiple different browsers on a variety of platforms and operating systems, and organizations can choose from a diverse set of servers and databases to host content. This arrangement has worked well for more than twenty years, encouraging innovation and competition—as long as the information being served is intended to flow freely and without restriction. However if the contents' owners want to control access to their information through encryption and authentication, the interoperability of the Web breaks down. The means of decrypting content for secure viewing has always required a separate, proprietary viewer or plug-in, because there is no Web standard for rights management at the file level.

Opinions differ on the wisdom and feasibility of implementing DRM for consumer content on the Web (although the channeling of rights-managed content into closed platforms such as Amazon Kindle and Apple iBooks has rendered opposition somewhat futile), but the absence of a security standard in the enterprise has created a vacuum into which a private corporation (namely Microsoft) could come to dominate the creation, distribution and consumption of information from end-to-end. Imagine a scenario in which the entire document workflow of the Fortune 1000 is beholden to a single vendor for its authoring applications, user identity management, decryption keys, usage data and other private and proprietary information.

"Azure Stack is completely unique to Microsoft. No one else who is in the public cloud business at any scale, has that kind of capability."

- Microsoft CEO Satya Nadella


Use your Microsoft account.
What's this?

<someone>@outlook.com

•••••••••••••

Keep me signed in

Sign in

No account? Create one!

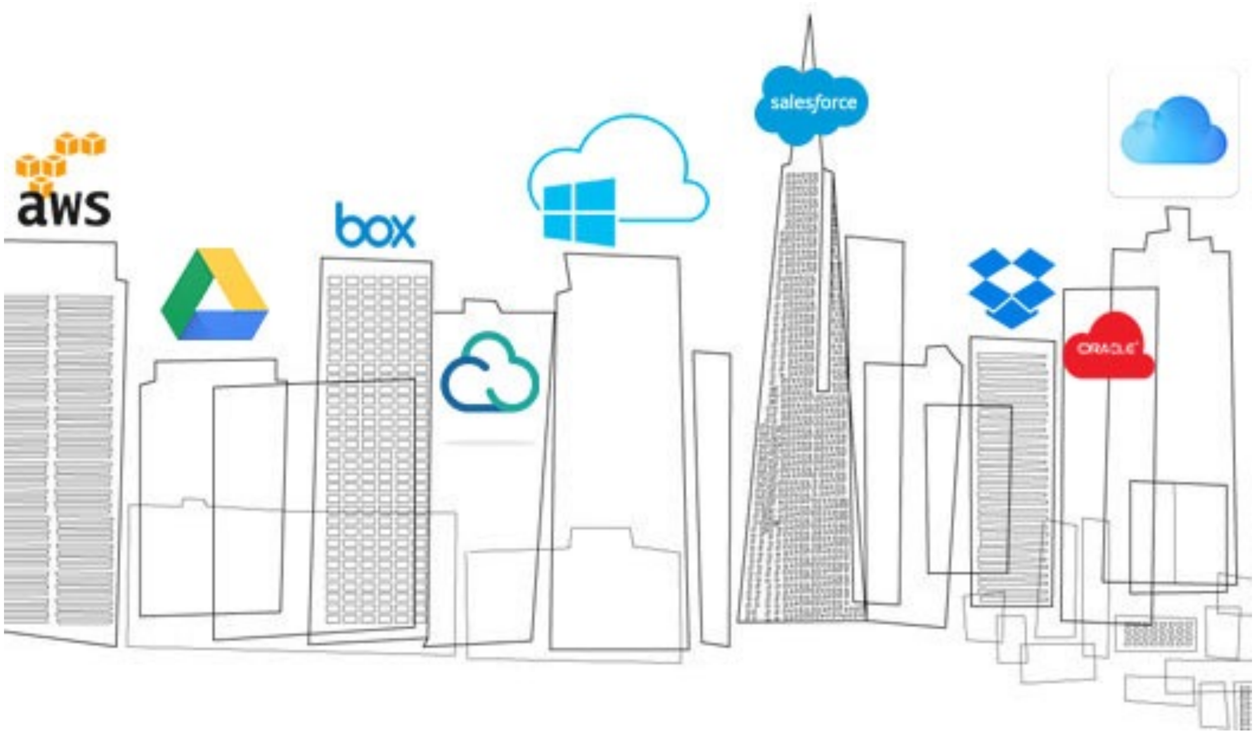
Aside from wiping out entire categories of cloud storage platforms and security vendors, a private monopoly in rights management technology would prevent its customers from choosing among workflow and server technologies, and put their data at risk of a potentially catastrophic security breach if any vulnerabilities were exploited. The vast majority of encrypted documents would be secured with a single proprietary method. Further, such a monopoly would threaten the fundamental framework of the Web, which was originally designed to enable servers and clients on different platforms, from different vendors, to speak to each other in a common language. Without a published standard for handling encrypted documents on the Web, it becomes possible, even necessary, to limit usage to a single viewer or editor—the one sold by the same vendor of the productivity suite, identity framework and encryption tools.

The Mirage of Cloud Security

How could we be at risk of a rights-management monopoly, when the cloud era has disrupted incumbent vendors and the dominant players now encourage integration via open APIs? In his blog post from 2013 "The Rise of The Cloud Stack,"[1] Aaron Levie of Box proclaimed "From this diversity of systems, customers will get choice, and with this choice we'll see better applications and solutions emerge. At a lower cost, and at a

[1] https://blog.box.com/blog/the-rise-of-the-cloud-stack/

higher quality." The cloud stack was supposed to free us from single-vendor "ecosystems" so we could migrate between increasingly sophisticated platforms. But as we will see, the lack of a standard document security protocol has instead resulted in a Babel-like city of secure towers that don't speak the same language, and therefore can't interoperate.



Each cloud service provider is an island, with every encrypted document trapped on that island

First, let's look at how cloud services handle user identity management, because any workable rights management scheme depends on user identity to determine whether or not to decrypt a file. One of the keys to the evolution of the cloud stack is the availability of what Gartner Group has named Identity and Access Management as a Service (IDaaS).[2] Using an IDaaS service, a company can create a Single Sign-On (SSO) system for a variety of different, and often interconnected, cloud applications. With a single login, a user might now update a record in Salesforce, triggering an alert in Slack, updating a spreadsheet in Google Docs and a report in Box.

[2] https://www.gartner.com/doc/3337824/magic-quadrant-identity-access-management

From the perspective of a network administrator, it is necessary that these new workflows also be auditable and their use controlled. So the "Access Management" part of IDaaS has become an important new element of the information security toolkit. In the example above, it is also critical that the administrator be able to revoke access to all the applications in that workflow all at once. In the current environment the granularity of IDaaS is normally at the application level; i.e. the IDaaS vendor controls access to the cloud provider itself, with that cloud provider managing access internally based on the identity provided.

In some cases, the cloud vendor may implement access control at the document-level, but the identification and permissioning of documents is specific to that environment. That is, there is no mechanism exposed by IDaaS vendors by which a document exported from Google Docs and imported into Box or Dropbox can be tracked or controlled as it moves from one environment to the next. Every cloud provider that offers some form of DRM is an island, and every protected document is trapped on that island.[3]

All DRM interactions involving the Azure RMS client/server exchange require a token from the Azure RMS server

RMS: Bundling Security and Identity

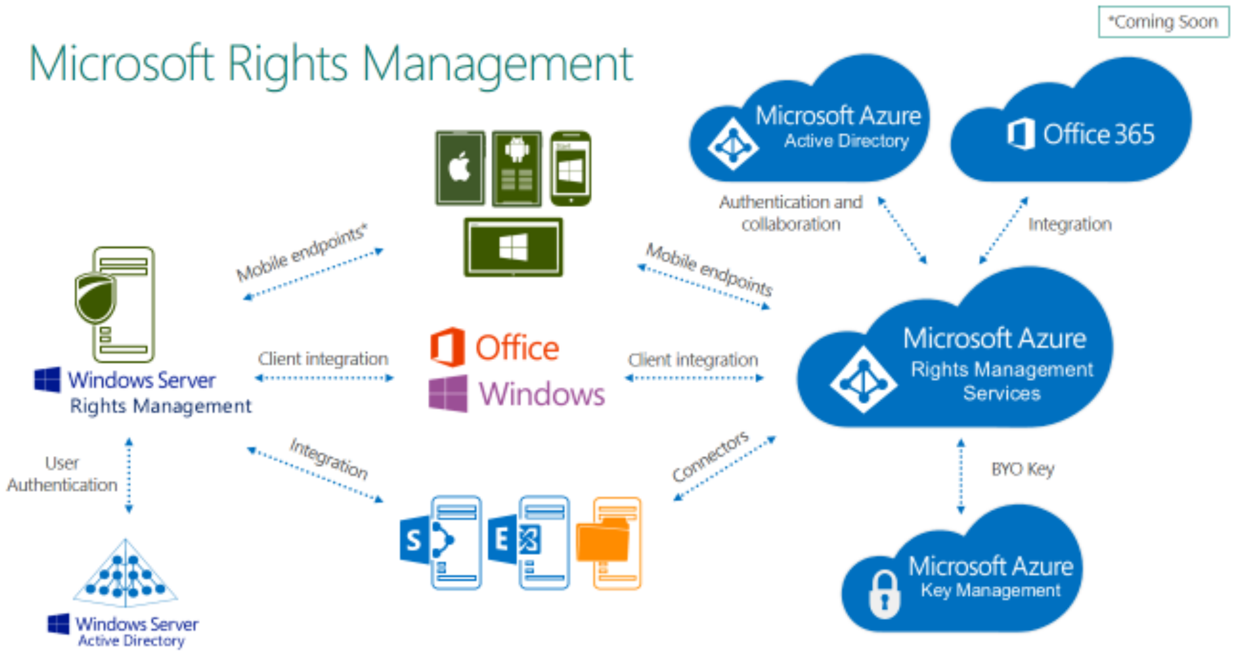
If there is no standard for document-level security in the cloud, what about behind the firewall, in the enterprise? There are multiple competing approaches to Enterprise Document Rights Management (EDRM), including the one developed by my company, FileOpen Systems. Each is functionally a silo; there is no standard for DRM interoperation. Among the EDRM solutions, the largest (by installed base or market share) and best-known framework is Microsoft Rights Management Services (RMS). It can certainly be argued that RMS is the *de facto* standard for controlled distribution of MS Office documents; the question is whether it can or should be the basis of a more formal web standard.

Microsoft RMS is a framework for DRM implemented natively in the Microsoft Office suite of products and also by third parties for some PDF viewers. The RMS software was introduced in 2003, evolving from and ultimately replacing an eBook DRM system,

[3] Documents can, of course, be moved from one cloud environment to another, but only in unencrypted form. The DRM implemented by, say, Box cannot be used outside of Box so documents must be decrypted for export.

Microsoft Reader, which was introduced in 2000 and discontinued around 2007.[4] Today RMS is predominantly, perhaps exclusively, focused on enterprise use-cases and workflows. The functionality was originally bundled into Windows Server 2003 and remains a part of the core Windows Server product. The RMS system is also tightly integrated with Microsoft's Active Directory product, indeed is formally named "Active Directory Rights Management Services" (AD RMS).

The tight integration of RMS with Windows Server has impeded adoption by entities not primarily using the Microsoft stack; RMS cannot be implemented directly in environments using Linux, or other Unix derivatives, at the server.[5] Further, the reliance upon Active Directory has reduced the utility of the system for external document distribution, insofar as the Active Directory environment of the document creator and


*Coming Soon
Microsoft Rights Management
Microsoft Azure Active Directory
Office 365
Authentication and collaboration
Integration
Mobile endpoints*
Mobile endpoints
Windows Server Rights Management
Client integration
Office
Windows
Client integration
Microsoft Azure Rights Management Services
User Authentication
Integration
Connectors
BYO Key
Windows Server Active Directory
Microsoft Azure Key Management

that of the document consumer are rarely integrated.[6] Microsoft is now addressing this concern via the federation of Active Directory networks using Azure Active Directory

[4] The original *Microsoft Reader* is not the same as the current *Microsoft Reader*, which is a PDF/Tiff/XPS viewer shipped natively with Windows 8.1 and later.

[5] There are almost exactly twice as many Linux/Unix servers on the web as Windows servers, see https://w3techs.com/technologies/overview/operating_system/all

[6] In such cases it is normally a requirement that the system administrators of the two networks exchange and install digital certificates, etc.

(Azure AD) and the cloud version of RMS (Azure RMS). These cloud-based systems for identity management and DRM are intended, among other things, to simplify the process of distributing encrypted content outside the firewall.

Microsoft has also integrated the RMS functionality into a number of clients on desktop and mobile platforms, under the banner "Office everywhere, encryption everywhere." This initiative[7] addresses both document security in the MS Office suites and message security in the Microsoft mail clients, via S/MIME. Viewing of RMS-protected documents on iOS can now be done in third-party viewer applications or with Azure RMS in the Microsoft Office applications for Mac, iOS and Android.

Microsoft's approach is fundamentally at odds with the core design of the World Wide Web, which is premised on the loose coupling of client and server functionality.

Support for PDF files in the RMS environment is implemented via partners, including Foxit, Nitro and Nuance, and in Adobe Acrobat/Reader via a plug-in from GigaTrust. RMS-encrypted PDF cannot be opened in the Microsoft Reader application, which is the default handler for PDF on Windows 8.1 and later, nor by the embedded PDF viewer in Microsoft Edge (or any other browser).

Standard or Stranglehold?

The presence of the RMS client in the primary Microsoft Office applications enables workflows in which effectively any user can distribute encrypted content to any other user, without a requirement that either install new software. No other DRM system can make that claim.[8] And while this situation is reminiscent of Microsoft's bundling of Internet Explorer with Windows, which was ultimately sanctioned by the US courts[9], the integration of Microsoft's DRM into Microsoft's Office applications has not been the subject of any antitrust actions. So it is likely that the bundling will continue, and

[7] See https://blogs.office.com/2015/02/18/office-everywhere-encryption-everywhere/ The document references IRM, or Information Rights Management, which is effectively synonymous with RMS in this context.

[8] We believe that the DRM system with the second largest number of native clients is FIleOpen, which is integrated into the pdf viewers of Foxit, Nuance, Nitro, Bluebeam, and Tracker.

[9] https://www.justice.gov/atr/us-v-microsoft-courts-findings-fact#vf

"Standards allow different layers to evolve independently and therefore faster and better."

- Sir Tim Berners-Lee, Founder & Director, W3C

continue to be an advantage for Microsoft in the enterprise, as long as Office remains the primary vehicle for document creation and collaboration.

Similarly, the bundling of RMS with Windows Server and Active Directory is not optional, as the RMS functionality depends upon proprietary logic in the Microsoft Server stack. Use of RMS with any alternative Web server is not supported. We assume that Microsoft tightly integrated the RMS client and server functionality in order to increase the security[10] and reliability of the system rather than as a competitive strategy, but the result is indistinguishable. There is no set of APIs and no licensing framework that would permit an independent vendor to create an alternative to the Microsoft implementation of RMS in Windows Server, or a cloud service that could be used as a replacement for Azure RMS.

While the original on-premises version of RMS required use of Active Directory for identity provision, the Azure RMS product offers the option of a federated identity mechanism, meaning that other IDaaS vendors can be integrated into an RMS workflow. However, by design, all DRM interactions involving the Azure RMS client/server exchange require a token from the Azure RMS server. So while an alternative identity provider may be provisioned, there is no mechanism in either RMS or Azure RMS for that identity provider to also manage the permissions on the document. In short, the RMS client and server cannot be decoupled.

As a result of these Microsoft design choices, it is not possible for an entity that chooses not to use Microsoft Server to implement RMS inside the firewall, nor is it possible for an entity using Azure RMS to operate with full independence from Microsoft. An IDaaS vendor can provide primary identity management, but cannot independently provide access control at the document level, as this capability necessarily involves the sharing of user and client data with Microsoft.

[10] Whether the design of RMS is, in fact, secure can be doubted, see https://www.nds.rub.de/research/publications/how-break-microsoft-rights-management-services/

An Alternative: Standards-Based DRM

Microsoft's approach is fundamentally at odds with the core design of the World Wide Web, which is premised on the loose coupling of client and server functionality. It is, for example, inconceivable today that a browser vendor would attempt to require use of a particular, and proprietary, server in order to implement SSL.[11] An entity publishing data on the World Wide Web today can exert complete control over the code used at the Web server, to the point of having the option to write that code anew in any language and run it on any hardware, provided that the result properly implements the required protocols, all of which are public.

It follows that any attempt to create a standard system for DRM on the World Wide Web would, at a minimum, need to support this degree of client/server independence. However, DRM systems are rarely designed around open interfaces, at least partly due to a tendency to think that the most secure system is one in which the developer has control over both ends of the client/server interaction.[12] Moreover, DRM differs from standard cryptographic implementations because one side of the interaction, the client, is not necessarily a willing participant in the secrecy of the conversation, and indeed may be an adversary.

So a completely open source DRM system is likely to be impossible. What can work, as demonstrated by the implementation chosen for the W3C Encrypted Media Extensions (EME)[13], is the combination of a compiled binary client library and an open interface to an arbitrary key management system. EME is now embedded in multiple applications and platforms, and allows content distributors to manage protected video files using one or a combination of key management and delivery systems (Google Widevine, Apple FairPlay, Microsoft PlayReady, Adobe Primetime, etc.)

---

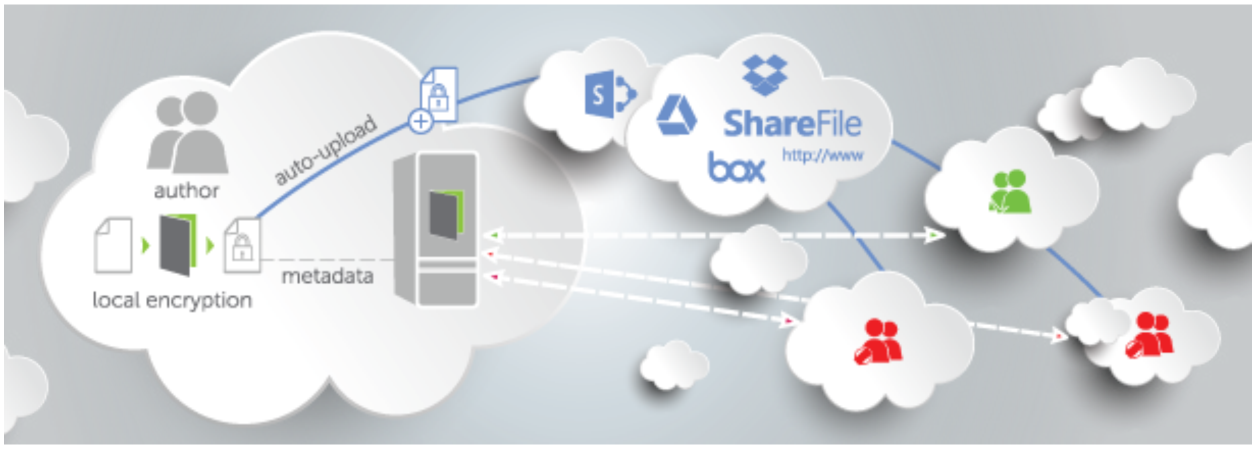
[11] This was not always so. Netscape Communications owns the patent on SSL (United States Patent No. 5,657,390 : Secure socket layer application program apparatus and method) and for a time did require use of the Netscape server. However Netscape granted a royalty-free license to any party attempting to build an implementation covered by the patent claims or the IETF (Internet Engineering Task Force) TLS specification.

[12] But this is not necessarily the case, as demonstrated by the continued effectiveness of open standards like SSL, S/MIME, OpenPGP (and now Bitcoin).

[13] https://w3c.github.io/encrypted-media/ For background see also http://www.html5rocks.com/en/tutorials/eme/basics/https://w3c.github.io/encrypted-media/ For background see also http://www.html5rocks.com/en/tutorials/eme/basics/

The FileOpen Rights Management Layer

The original design of the FileOpen DRM software, first articulated in 1997, describes the same general architecture.[14] FileOpen rights management was explicitly designed to work with any external identity provider, and to permit the entity encrypting documents to specify all relevant metadata (e.g. document identifiers, encryption keys). In the company's early years, the only way to implement FileOpen software was to develop a custom server to manage customer and document identities and to implement business logic governing usage. Today, commercial "PermissionServer" implementations are available in .NET and PHP, and multiple cloud services are built around the FileOpen Client[15] and protocol, which is published.



FileOpen DRM is typically configured to encrypt documents on a local device, so unencrypted files are never distributed. The encryption process stores document metadata on a "PermissionServer," which may be on-premises or hosted in the cloud. Once files are encrypted and registered they may be distributed by any method, and stored by any cloud hosting provider.

The architecture of FileOpen rights management is such that is not tied to any particular document format or viewing application.[16] This differs markedly from Microsoft's approach, which developed RMS for its own Office applications and has

[14] http://www.fileopen.com/hs-fs/hub/74620/file-15465899-pdf/docs/fileopenSPA.pdf

[15] The FileOpen Client is distributed as a plug-in for the Adobe Acrobat/Reader platform, an Add-in for Microsoft Office, and as a multi-platform library that can be implemented into any application.

[16] The original and still most used implementation of FileOpen DRM is for the Portable Document Format (PDF), the most open and standard format for distribution of complex or formatted information. Similarly, one of the first and largest sectors to standardize on the FileOpen software is the community of Standards Developing Organizations (SDOs), e.g. ANSI, ASME, Afnor, DIN, etc.

demonstrated a clear bias toward those applications with respect to competing tools (e.g. LibreOffice) and formats (e.g. PDF). We believe that decoupling the rights management protocol from the document format and viewer vendor enables organizations to share documents securely with the broadest set of users, inside and outside the firewall. It also protects publishers and document creators from a potentially costly "lock-in" to the format vendor, who would control the keys to encrypted documents and store private data about their end-users.

Moreover, because the FileOpen software does not rely upon any particular identity provider, it can be implemented natively by any IDaaS vendor. All such implementations are independent and private, in the sense that information about users/documents/permissions is fully controlled by and visible only to that vendor. Interoperation between IDaaS implementations, or between IDaaS vendors and cloud application providers, or between different cloud application providers, is likewise possible without direct involvement of FileOpen Systems, using the published interfaces of the system.

While FileOpen rights management is not yet a standard, it was designed and continues to be developed around the principles that underlie Web standards. In particular, the FileOpen software was designed not to be an end-to-end secure publishing system, or even a complete system for DRM, but a flexible component that can be integrated to extend the functionality of such systems--one thread in a larger, interconnected, Web.

For more information on FileOpen rights management architecture and implementation options, visit https://www.fileopen.com/products/office-rights-management