Your Cloud in my Company: Modern Rights Management Services Revisited

Martin Grothe, Paul Rösler, Johanna Jupke, Jan Kaiser, Christian Mainka and Jörg Schwenk
Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
{martin.grothe, paul.roesler, johanna.jupke, jan.kaiser-e5q, christian.mainka, joerg.schwenk}@rub.de

***Abstract*—We provide a security analysis of modern Enterprise Rights Management (ERM) solutions and reveal security threats. We first take a look on Microsoft Azure, and discuss severe attack surfaces that companies enabling Azure in their own trusted infrastructure have to take care of. In addition, we analyze Tresorit, one of the most frequently used End-to-End encrypted cloud storage systems. Tresorit can use Azure and its Rights Management Services (RMS) module as an additional security layer: a user should be able to either trust Tresorit or Azure. Our systematic evaluation reveals a serious breach to their security architecture: we show that the whole security of Tresorit RMS relies on Tresorit being trusted, independent of trusting Azure.**[1]

***Keywords*-ERM, AD, RMS, Cloud, Azure, Tresorit**

I. Introduction

Information security is one of the key challenges when it comes to protecting a company's assets [2], [12], [4], [6]. Rights management is one widely used technique to address this challenge [3], [38], [29]. Currently there are different approaches realizing rights management in software, depending on the area of application. For example, a publishing house can protect their ebooks with Digital Rights Management (DRM). The same principle is valid for music and video files [5], [11], [37]. Apart from the consumer area, there is Enterprise Rights Management (ERM) used by companies in business environments [7], [28]. For example, banks, law firms and automotive make use of Microsoft Rights Management Services (RMS). RMS can be used to restrict reading, writing, or printing permissions of office documents, texts or binary files. In recent years, companies also tended to outsource their locally hosted systems into modern cloud environments. This has obvious performance, scalability and availability advantages by reducing costs for the outsourcing company.

In this paper, we give an answer to the following research question: *which security threats can occur, when combining cloud systems with ERM?*

We therefore take a look at Microsoft Azure RMS [39], one of the most prominent and wide-used RMS cloud systems. We describe the structure of Azure RMS and how it technically protects a file. By this means, we identify an obvious, but serious security threat: since Microsoft controls the Azure servers, and RMS uses a certificate chain to protect files, whereat the certificates are generated by Azure, Microsoft is able to decrypt all RMS protected files, stored on Azure storage space.

Due to this threat, we go one step further and investigate Tresorit, a widely used cloud storage provider [42]. Besides end-to-end encryption, Tresorit is the only available solution in the Google Play Store supporting RMS. To conduct this, Tresorit uses a two layer approach: while the file is protected with Azure RMS, the file is additionally encrypted with Tresorit's end-to-end encryption.

The analysis of Tresorit is a complicated task, since Tresorit does neither provide its source code nor gives detailed protocol descriptions [36]. We therefore created a test setup and extracted the Tresorit protocol step-by-step via traffic analysis and process monitoring. Since the Tresorit servers communicate with Azure servers in the background (Server-to-Server communication), our test setup cannot cover this traffic, which makes the analysis more difficult. We estimated this communication by observing the traffic that we could see and communicated our results with the Tresorit developers, who confirmed our results. After reconstructing the protocol in detail, we found a major security breach in Tresorit: We show that the Tresorit servers can get access to content keys held by Microsoft. This was explicitly excluded by the authors of Tresorit whitepaper [41]. We contacted the developers and they acknowledged our work.

Our Contribution. We make the following contributions.

- We give an overview on modern rights management systems and their requirements.
- We analyze Microsoft Azure and show how it uses RMS to protect files.
- We identify security threats that appear when a company decides to use modern Rights Management Services in a cloud.
- We analyze Tresorit and their RMS implementation.
- Our analysis reveals a serious security threat in Tresorit (VI-C) that breaks the concept described in Tresorit's whitepaper [41].

Our results were responsibly disclosed with the developers. They updated their whitepaper and removed the statement that Tresorit cannot read RMS protected files.

II. Modern Rights Management Systems

Currently there are different solutions available to manage rights on digital contents. They are categorized by

[1]This is the full version of our paper published at the *2016 11th International Conference on Availability, Reliability and Security (ARES 2016)*.

their area of application: DRM and ERM.

Digital Rights Management. Control over digital content in the consumer area is commonly known as DRM and is pushed along by the music and movie industry. The idea behind DRM is to sell and enforce usage rights for digital contents. A practical example for this is Amazon Kindle with its e-book reader with an integrated DRM system. Kindle readers have their own e-book format, which is only readable with Kindle software and only authorized users are permitted to read a book.

Enterprise Rights Management. In the business area, ERM is used to manage the access of company's employees to files that must be kept internal to protect intellectual property. Therefore these access rights are set by the author. For example, a certain user is maybe allowed to view a file, but has no rights to copy the content or to print it. Once set, the access rights must be independent from the physical and logical location of the file, so that no unauthorized user can access the content. Thus the goal of using ERM is that companies can protect confidential files and control their usage in many details.

In this paper, we concentrate on ERM and Microsoft's cloud implementation of it, called Azure RMS.

III. ERM Characteristics

ERM systems have to fulfill special requirements. We therefore divide ERM characteristics into the following categories. They are mostly equivalent to the features of DRM systems described in previous works [3], [27], [30], but adopted to the requirements of ERM and more recent technologies.

Availability. The availability of ERM protection consists of basically two aspects: (1.) protected data can be accessed for a specific amount of time, while the user is offline. This usually means, that on a first use, the user must receive some cryptographic keys from a trusted keyserver, which is either on the Internet or part of the company's network. Later on, the user can work with a file independent of that server. (2.) Protected files must be accessible from different devices, for example, laptops, tablets and smartphones. Recent ERM implementations offer browser-based access.

Full Rights Control. The author of a protected file must have full rights control. This includes: (1.) control of access and usage of it – the author can grant and revoke rights. (2.) Changing access rights must be possible after the first distribution. (3.) In addition to the author's control, it must be possible to transfer rights directly or through a Trusted Third Party (TTP), for example, by a company authority.

Confidentiality and Integrity. ERM protected files must be confidential and unmodifiable. (1.) This must be independent of the files distribution. For example, files transferred via email must be protected as well as files exchanged via mass storages. (2.) The protection of files is persistent. This especially means, that a protected file leaving a company (e.g. via email) must still be protected and only authorized users must be able to use it.

IV. Microsoft Azure

Microsoft Azure is a modular cloud platform system and can be separated into three categories: Cloud Services, Virtual Machines, and App-Services. Cloud Services provide special purpose virtual machines, which can be used to build scalable web services. In contrast to normal virtual machines, these special VMs are maintained by Microsoft. The category Virtual Machines is comparable with standard services. A user rents storage, virtual machines, installs custom software on them and links these virtual machines via virtual networks. The App-Services are used to export application logic, resources and instances into the cloud. Besides these three categories, Microsoft Azure has a large pool of services it provides [15]. For this paper, the most relevant service is Azure RMS.

A. Azure RMS

Azure Rights Management Services is part of Azure Active Directory and enables its users to share files, for example, Office documents, with other users and preserve the control over the content and its distribution. In comparison to the classical Active Directory Rights Management Services (ADRMS), Azure RMS client software is available on all modern platforms (Android, iOS, Linux, Mac OS, Windows) [16]. Therefore, protected content can be consumed on mobile devices, laptops and computers.
These devices need Azure RMS client software, for example, RMS plugin for MS Office, to consume protected files. In short, the user needs 3 clicks to set permissions (e.g., *view*, *edit*, *print*, etc) for a *.docx file. Subsequently the file can be distributed to other users, via email, cloud storage or USB stick. Before another user can open the file, he has to authenticate to Azure. This is automatically done if the user has already supplied Office (e.g., Microsoft Word) with his account credentials. Otherwise the user can create an Azure account or use the Single Sign On protocol with an identity provider of his choice (e.g., Google, Facebook, Yahoo, etc.) [20]. Afterwards Azure RMS performs the authorization process. In case the user is allowed to access the document, the RMS plugin gets the necessary information from the Azure RMS and the document is opened by the Office application. [17]

B. Azure RMS in detail

The protection mechanism of Azure RMS relies on a public key infrastructure (PKI), symmetric cryptography and application level rights enforcement. To get a valid client certificate for the PKI, a user has to authenticate to the Azure AD. There are different ways, how a user can authenticate to an Azure AD, depending on the Single Sign-On protocol used and the identity provider. In case the authentication procedure finishes successfully, the Azure AD server forwards the user to the Azure RMS instance. This instance issues a client certificate (Rights Account Certificate (RAC)) for the users client software, for example, Office RMS plugin. The certificate

consists of a private-, a public key and a signature from the Azure RMS instance. Once issued, the client software uses the client certificate to authenticate to the Azure RMS instance. When this initial procedure is finished, a document can be protected via Azure RMS. [17]

Creating protected documents. We describe this process, using the example of Microsoft Word with an installed RMS plugin. The structure of a protected Office Word document is shown in Figure 1.

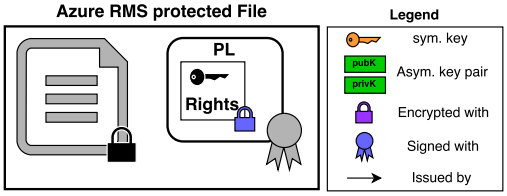

Figure 1. Azure RMS protected Word file.

(1.) The RMS plugin generates a random, symmetric AES content key. This key is used to encrypt the original (unprotected) content, for instance, the Word document. (2.) Afterwards two licenses are created by the RMS plugin: the first one is for the author of the file. The second one is the so-called Publishing License (PL), that is used by all other users. The PL stores a list of specific users, groups, their corresponding rights (policy) and the symmetric content key. The PL is intended for the Azure RMS instance, which is responsible for creating licenses for every other user or group specified by the author. To ensure that the policy in the PL is authentic, a signature with the private key of the Rights Account Certificate (client certificate) is created over the policy. In addition, the policy and the symmetric content key in the PL are both encrypted with the public key of the Azure RMS instance. (3.) The encrypted original document (Step 1), the encrypted content key and the signed and encrypted policy (Step 2) are stored in one file: the protected document.

In general, access rights for documents can be assigned by using templates, which are predefined policies or by creating a custom policy with individual access rights for groups and users. Groups can be created via the Azure AD to assign specific access rights to all its users at once.

Accessing protected documents. Once a protected document is received by a user, the client software starts the consuming process. This process is shown in Figure 2. We assume, that the user already authenticated to the Azure AD and the RAC is already stored on the client device.

(1.) The client software sends the PL, that is contained in the protected file, as well as the RAC, of the user, to the Azure RMS instance. (2.) Azure RMS decrypts those elements with its private key of the Server Licensor Certificate (SLC). (3.) A list of access rights for the requesting user is created by the Azure RMS instance, according to the policy. (4.) The content key is extracted from the decrypt policy. (5.) A Use License (UL) is created from the content key and the access right list. The UL

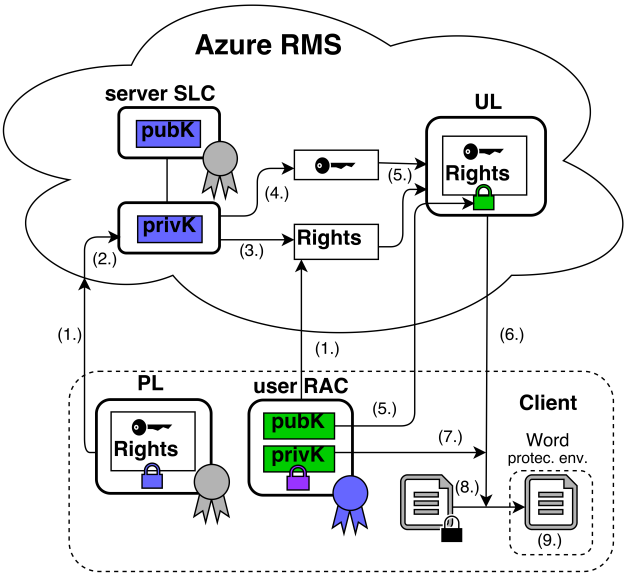

Figure 2. Accessing a protected Word file.

is encrypt with RAC public key of the requesting user. (6.) Afterwards the UL is sent to the Azure RMS client. (7.) The client software decrypts the UL with the private key of the user's RAC. (8.) The protected document is decrypted with the symmetric content key from the UL. (9.) The opening software (e.g., Microsoft Word) gets the decrypted document with a list of rights from the RMS plugin. The listed rights are enforced by Word [40].

C. Security Threats

Despite the fact, that Microsoft uses open standards and techniques in Azure, there is currently one strong limitation: Azure runs only on servers of Microsofts data centers. All Services, like Azure RMS, run on Microsoft servers, resulting in trust threats we need to address. With Azure Pack, Microsoft provides a collection of Azure technologies that can be installed on-premise, but they include just a few services and not Azure RMS. [23]

1) Trust Threats: We define trust threats as possible scenarios, in which authentication processes are manipulated and customer data are misused. For that we clarify a trust model.

Suppose a company that is using Azure. In this model, Microsoft is considered the attacker, independently of the aspect, whether the company using Azure trusts or mistrusts Microsoft. The attacked company uses a local Active Directory (AD), which is integrated in an existing Azure instance. We will answer the following questions:

(1.) How is the access to cloud data defined by Microsoft?
(2.) In which way has Microsoft access to stored data?
(3.) Does Microsoft has access to certificates, licenses or keys of Azure RMS?
(4.) Can Microsoft restore deleted data from the cloud?

(1.) How is the access to cloud data defined by Microsoft? Microsoft defines a special privacy and security statement for online data. The stored data is split

in customer data, administrator data, payment data, and support data. Customer data includes every type of data (sound, text, images, or software) which is stored, used, or processed in the cloud or within cloud applications. Administrator data, payment data, and support data includes contact and financial data for the correspondence between Microsoft and the customer.
In general, Microsoft verbalizes the statement "You own your data" to indicate that they do not access that data. Also, they do not mine customer data or use it for advertising. In the former case of a government or law enforcement request, Microsoft had to disclose data. [19]
(2.) In which way has Microsoft access to stored data? Microsoft itself asserts that customer data is only used for providing chosen services as well as troubleshooting, service improvement and personalized customer experiences. A Microsoft support member is only able to access the stored customer-related data for a defined time and task. In case the company mistrusts Microsoft, the data should only be stored encrypted in the cloud [19]. Due to the fact that Microsoft provides the data for services and performs troubleshooting, Microsoft must have access to customer data. In these cases Microsoft refers to the *EU Model Clauses* and the *ISO/IEC 27018* standard [18] Assuming that Microsoft is the attacker and ignores their privacy statements, all data, which is stored or processed in the cloud, as well as application data can be accessed by Microsoft.
(3.) Does Microsoft has access to certificates, licenses or keys of Azure RMS? Microsoft uses different servers to store certificates and licenses [13], [14]. These servers are controlled by Microsoft. Further Microsoft issues, by default, all certificates and licenses used by Azure RMS. Thus, Microsoft controls the access to protected files and is able to grant itself rights for any file. In general, certificates, licenses, and private keys are defined as access control data and as this, stored encrypted in the cloud [19], [24]. Microsoft offers a protection mechanism named *Azure Key Vault*, which is an advancement of *Bring your own key* technology, used for the first time in Azure RMS. This technique helps against key stealing by intruders, due to the fact, that the keys are stored in Hardware Security Modules (HSMs). However, HSMs do not protect against using the keys for decryption, signing or encryption as insider [22], [21]. A customer cannot distinguish, whether the access to a HSM is triggered by a legitimate Azure RMS request or a Microsoft employee.
(4.) Can Microsoft restore deleted data from the cloud? According to Microsoft, every data which is deleted by the cloud option "delete" cannot be retrieved. As a result of this delete procedure, the pointer to the stored document in the data storage is cleared. Thus, no real delete operation is triggered and a recovery of the data is possible. Further an Azure administrator is able to read all "free" blocks of Azure storage. This includes previously stored and deleted data as well. Through this way a user-data association is not possible, but deleted data can be retrieved. If a Microsoft employee gets access to a fitting task, he is also able to retrieve data. [43]

2) Active Directory modification attack: With the collected information, we try to design a possible attack scenario. In case a company uses Azure RMS with an own on-premise AD server, the Azure AD and the on-premise AD get synced side-to-side. Thus, Microsoft can add, modify or delete user information in the Azure AD instance and wait for the synchronization process. Resulting in a modification of the on-premise AD and in a compromise of company infrastructure.

3) Impersonification attack: Since Microsoft has access to customer and authorization data, this information can be used for further attacks, such as impersonification of an employee of a company, which uses Azure services. As a result, an attacker would get detailed information and could try to infect the infrastructure with malware. It just needs one fraudulent Microsoft administrator and a company using Azure RMS.

V. Tresorit

Tresorit is a cloud storage alternative to Dropbox that focuses on security. It implements client side encryption for all stored files and claims that the company behind Tresorit has no possibility to read the plaintext of protected files.

A. Tresorit End-to-End Encryption

With the Tresorit client software a registered user of Tresorit has to create an account with the help of the client software. He afterwards can create a so-called *tresor*, which can be seen as a protected folder: all files within this folder are end-to-end encrypted with AES256 in Cipher-Feedback Mode. The encrypted files are sent to the storage server and afterwards synchronized across all Tresorit devices (Desktop and/or mobile) owned by the user.

Tresorit additionally offers *sharing* of a tresor with other Tresorit users. This feature allows collaborative working with Tresorit users. Due to the nature of end-to-end encryption, all users must be able to decrypt the files within the tresor in order to read or edit them. For this reason, Tresorit implements a key management that we have analyzed and describe in the following.

All file keys within a tresor are encrypted with one AES-256 bit tresor key. The tresor key is encrypted with the public key (RSA-2048 bit) of the user. Since the private key of the user is distributed to all of his devices, the files can be decrypted on these devices.

Sharing is a Threat. As previously described [44], Tresorit is the only Certification Authority (CA) of its system, due to this fact Tresorit can obtain the key of a shared *tresor* by performing a Man in the Middle (MitM) attack during the sharing process. Since the whole sharing process is untransparently conducted by the client software by requesting the user certificate of the receiving user and requesting an Access Control List (ACL) update, this process requires trust in: (1.) the client software, (2.) the Tresorit CA and (3.) the server which enforces the ACL permissions. The process of sharing a tresor between user

A and B is divided into 8 steps: (1.) The owner of the tresor (user A) requests the certificate of user B. All certificates are issued by Tresorit and contain the public key of the corresponding user. (2.) The client sends an invitation to the Tresorit storage server. It includes the asymmetric encrypted key for the shared tresor, whereat the public key of the user B is used. This asymmetric encrypted tresor key is stored in a file, the so called *group key* file. For every invited Tresorit user, an additional encrypted tresor key is stored in this file. Thus, no group key agreement for the encryption of the tresor key is required. (3.) Finally, the Tresorit storage server adjusts the server-side controlled access control permissions, so that the user B can download the shared files from the server (and later decrypt them). (4.) Afterwards user B is notified on the invitation. (5.) User B can download the group key file, (6.) decrypt the tresor key with his private key, (7.) download the tresor from the storage server and (8.) decrypt the files in the tresor with the tresor key.

B. Security Threat

The previously described security design has a major pitfall [44]: As soon as a tresor is shared with another user, Tresorit is able to get in possession of the respective decrypted files.

Therefore, Tresorit has to perform the following 3 steps: (1.) when a user A wants to share access to a tresor with user B, Tresorit has to respond with a certificate. Instead of responding with the certificate belonging to the user B, Tresorit generates a new asymmetric key pair, embeds the public key into a certificate, and uses this certificate for responding instead of the user B's one. (2.) Once the user A uploads the group key file, Tresorit can decrypt the tresor key with the private key, generated together with the public key for the certificate. Consequently Tresorit can decrypt all files in the tresor. (3.) To conceal the attack, Tresorit can replace the encrypted tresor key, in the group key file, with the one that is correctly encrypted with the user B's public key.

Neither the owner (user A) nor the invited user (B) could detect the attack because Tresorit does not provide any key verification (Proof-of-Possession). Please note, the certificate structure of Tresorit is more complex than described here, but the main problem behind it is that all certificates are issued by one of the Tresorit CAs.[2]

To address the problem of a monolithic CA, Tresorit RMS can be used.

VI. Tresorit RMS

Tresorit RMS or Tresorit DRM, how the developer call it, protects files with two layers: (1.) The inner layer is the RMS protection, which is directly applied to the file. (2.) The outer layer is the Tresorit end-to-end encryption which is wrapped around the RMS protected file.

The inner RMS protection is provided by Microsoft Azure RMS. As such, Tresorit RMS makes use of the same licenses/certificate system, AD groups, and servers as described before. Thus, it is independent of Tresorit's CA architecture. The outer layer is the previously described Tresorit end-to-end encryption and therefore has the same security threat. The combination of both layers should – according to the Tresorit whitepaper [41] – result in a higher security level, because files are protected by two independent systems.[3] This means, that a user must either trust Tresorit or Azure.

In the following, we show that this concept has a security breach and a user *must* trust Tresorit independently of trusting Azure or not.

A. Test Setup

Since Tresorit does neither provides a detailed protocol flow of its RMS module nor the source code, we reconstructed the protocol scheme, by analyzing the Tresorit client software in a test environment. We chose the Windows client software, since it provides the most functionality in comparison to other client implementations. Three types of traffic were monitored: Network traffic, access to the Windows Registry and access to the local file system. The communication to the Tresorit servers is protected by Transport Layer Security (TLS) and the respective CA certificates are pinned in the software, to prevent MitM attacks on the network layer. For our analysis, the Tresorit developers thankfully provided a modified client version, which logs the whole plain network traffic, so that we could eavesdrop and analyze each message between the Tresorit client and the Tresorit servers. Despite this logging, the provided software client behaves identically to the original software. We collected the access data with Process Monitor [25], Wireshark [1] and the modified client version of Tresorit. We tested the functionality of Tresorit (e.g., registration, uploading, sharing, revoking, ...) in a fix test sequence with and without integrated Tresorit RMS. The collected data from the mentioned sources were merged and filtered to obtain the protocol related actions.

B. Tresorit RMS in detail

To understand the reason for the security breach three workflows need to be understood: (1.) Initialization of Tresorit RMS (2.) Creation of a protected tresor (3.) Modification of the access rights for a protected tresor. Due to space limitations they are summarized in the following and can be found with a detailed protocol graphic in the full version of the paper.

During the registration at Tresorit, a new Azure AD user is registered by the Tresorit servers. The respective Azure credentials are stored on Tresorit servers and are distributed to all client devices of this user. As soon as a user protects a *tresor* and its including files with RMS, these credentials are used to authenticate at the Azure RMS servers. The client software of the author

[2]Contrary to the description of Wilson and Ateniese all Tresorit CA certificates are currently issued by one Tresorit root CA.

[3]After our responsible disclosure to the developers of Tresorit, they removed the statement that Tresorit is unable to decrypt Tresorit RMS protected files.

only requests the creation of a PL for three AD groups (manager, editor, reader) with the respective permissions while the Tresorit servers create these groups with the users, the owner permits to have access. As soon as the author or a manager wants to update the permissions, only the AD groups are adapted by the Tresorit servers.

Initialization of Tresorit RMS. In order to authenticate at the Microsoft servers (1.) the Tresorit client software requests the Azure credentials for the user during the registration or login phase from the Tresorit RMS API. (2.) These credentials are register at the AD server by the Tresorit DRM server. (3.) Once the user agree to use the Tresorit RMS in the client software, the Azure credentials are stored in the Windows Registry where (4.) a Microsoft RMS compatible software (e.g., Microsoft Office) can fetch them.

Creation of a protected tresor. During the creation of a Tresorit RMS protected tresor (1.) three new Azure AD groups (see below) are created by the Tresorit DRM server at the Azure AD server and (2.) their names are forwarded to the client. (3.) As soon as a new file is stored in the Tresorit RMS protected tresor, the client authenticates at the Azure AD server. This server grants a token for the communication with the Azure RMS server. (4.) Next the client requests a PL for the three Azure AD groups with the following permissions: *Readers* can read the files, *editors* can read and edit the files and *managers* can read, edit, extract[4] and print the files. Furthermore managers can share and revoke the permissions of other users. (5.) The Azure RMS server responses the PL together with the RMS file key. (6.) The Tresorit RMS module uses this file key to encrypt the stored tresor and appends the PL to it. (7.) Finally the tresor is Tresorit end-to-end encrypted and uploaded to the Tresorit storage server.

Modification of the access rights for a protected tresor. To modify the permissions an owner or manager conducts a permission update as it is proceeded without the Tresorit RMS protection, except the last step: (1.) The client sends an invitation or removal to the Tresorit storage server including the updated group key file, in case access to the file is either initially granted or fully revoked. (2.) The Tresorit storage server applies the permissions regarding the access control and forwards them to the Tresorit RMS server. (3.) The Tresorit RMS server adds or removes the invited or removed user at the Azure AD server to or from the respective AD group.

The receiving Tresorit client requests the permissions from the Tresorit storage server. When a file is opened with an RMS compatible software, the Azure AD server is requested for the group membership and the RMS software enforce the permissions.

Protected File. As illustrated in Figure 3 and described previously the protected file is encrypted twice. The symmetric file key of the RMS encryption is stored in the PL,

[4]Managers can create a local copy of the file while editors can only modify the respective file.

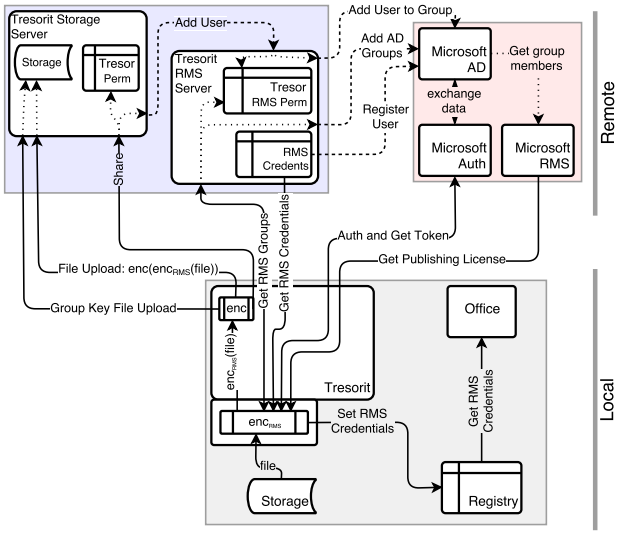

Figure 3. Included parties of the Tresorit RMS workflow and their relations. The protocol messages are extracted from our traffic analysis. Dashed lines indicate traffic that was not seen in our test setup (e.g., server to server communication). Thus, we guessed it and contacted the Tresorit developers who confirmed our guesses.

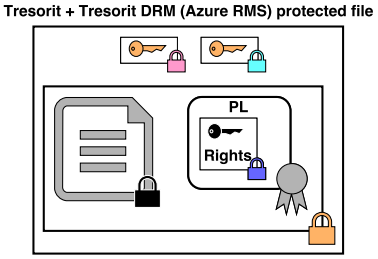

Figure 4. Protected file with applied Tresorit end-to-end encryption and Tresorit RMS protection: the protected RMS file (see Figure 1) is encrypted with a Tresorit file key, which is encrypted for the private Tresorit keys of the permitted users.

as outlined in section IV-B. The file key of the Tresorit end-to-end encryption is encrypted with the tresor key, which is encrypted with the public keys of all permitted Tresorit users. The encrypted file keys are attached to the respective encrypted files, which are stored on the server with the group key file. The Azure credentials of the users are stored on the Tresorit RMS server, on the Azure authentication server and locally as it can be seen in **??**.

C. Tresorit RMS Security Breach

As Tresorit RMS provides a two layer protection, which encryption procedures are independent, we consider the providers of Tresorit and Azure RMS servers to be "honest but curious". Thus, all requested actions are performed correctly, but they can use all remotely stored data to gain protected information [10]. Therefore we assume, that either Tresorit or Azure RMS needs to be trusted to reach confidentiality. Further, we mistrust the CA of Tresorit, as in the recent approach of Wilson et. al. [44]. This seems to result in a weak attack, but for a client-side encrypted software it is legit to assume, that some parts of the infrastructure provider cannot be trusted. It is also confirmed in the Tresorit whitepaper: "even if

Stored Information	Tresorit			Azure	
	RMS Server	Storage Server	Certificate Authority	Active Directory Server	RMS Instance
Azure authentication information	X			X	
Azure RMS permission information	X	X		X	X
Azure RMS file encryption key					X
End-to-End encrypted RMS protected tresor		X			
Public Keys and trusted Certificates			X		

Figure 5. Stored information for Tresorit RMS and their location.

Tresorit has access to Tresorit RMS protected files" it would be "unable to decrypt the (protected) content"[5]. All information regarding the permission management of Tresorit RMS is received from and stored on the Tresorit (storage and RMS) server. The group encryption key file is stored on the Tresorit storage server and the access permission information is stored on the Tresorit RMS server. Partial permission information in the form of AD group membership and the corresponding rights for a tresor are stored on the Azure AD server and the Azure RMS server. The Azure credentials are stored on the Azure AD server and on the Tresorit RMS server.

Tresorit RMS attack. In short, the attack needs a privileged Tresorit employee *M* and a user *A*, who shares his files with another user *B*.

The following steps are needed in order to perform a successful attack: (1.) A privileged employee *M* creates a public/private key pair and issues an additional certificate, containing the created public key, for an existing user *B*. As result, *M* can impersonate *B*. (2.) When user *A* wants to share a tresor with *B*, he requests *B's* certificate. (3.) The CA will send the new by *M* issued certificate to *A*. (4.) As part of the sharing process (section VI-B) *A* will add the asymmetric encrypted content key for the tresor to the group key file. The asymmetric encryption key *A* used for the encryption of the content key, is the public key from the certificate he got from the CA. (5.) Now *M* can use his previously created private key and decrypt the content key from the group key file. (6.) Afterwards, the content key can be used to remove the end-to-end encryption. (7.) The resulting document is still protected by Azure RMS. *M* now has two opportunities: a) He copies the Azure credentials of user *A*, stored in plaintext on Tresorit RMS server. Then he uses them in the corresponding Azure capable RMS software, like MS Word and removes the Azure RMS protection. b) He uses his privileges to add himself to the Azure RMS reader, editor or manager group for the end-to-end decrypted tresor. Regardless which way is used by *M*, both protection layers are removed by the attack and the whole content of the previously protected tresor is readable. *M* could further encrypt and forward the file to *B* in order to cover his tracks.

[5]From the Tresorit RMS white paper, it was updated after we disclosed our findings: https://tresorit.com/files/tresorit-drm-whitepaper.pdf

Responsible Disclosure. The Tresorit developers acknowledged and confirmed the results of our analysis, which we responsibly disclosed. They announced to adapt the design, so that customers can use their own AD instance. Depending on the implementation this could lead to the expected split of trust.

Recommendations. In general, if a system pursues to reach confidentiality with limited need of trust in the provider, all permission management, including cryptographic operations, should be conducted on the client machine. For Tresorit we propose the following: by shifting the registration of new Azure AD accounts and the update of permission information for tresors to the client software, the trust relation to Tresorit and Azure could be split effectively. Since the Azure instance is currently administered by Tresorit it would not enhance the security level, but this shift would enable the usage of separated Azure AD and Azure RMS instances. Thereby the providers of the Tresorit and Azure servers could be fully parted.

Tresorit could also implement the import of own user keys or a feature to verify contact keys to decrease the requirement of trust in the software.

VII. Related Work

During our research, we saw different names used for the topic Access Rights Management, such as Rights Management Services, Enterprise Rights Management, Information Rights Management (IRM) and Enterprise Digital Rights Management (EDRM). Due to their close relationship we see them as synonyms. This also applies for the related work paper discussed in the following. For the paper we conducted different analyses related to the topic. We categorized them as follow: (1.) theoretic and practical research on AD and ERM, (2.) risk assessment of cloud computing in general and specifically of Azure and (3.) analyses of secure cloud storage implementations.

Yu and Chiueh proposed a Display-Only File Server [45]. Therefore, they describe requirements to ERM systems and summarized three specific ERM implementa-

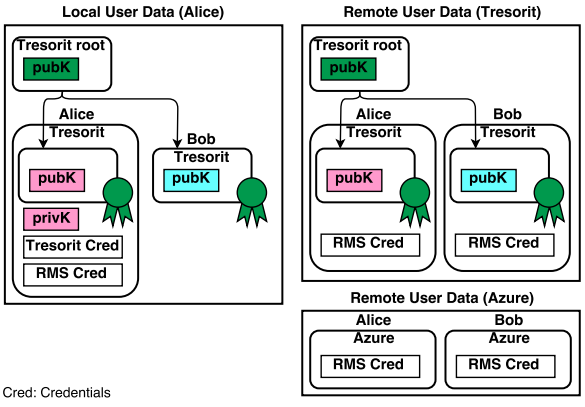

Figure 6. Simplified distribution of user certificates and credentials of Tresorit and Tresorit RMS.

tions. (1.) Microsoft RMS, (2.) Liquid Machines which was afterwards bought by Check Point [31] and (3.) Authentica's PageRecall which was afterwards bought by EMC [32]. They argue that the effectiveness of an ERM system bases on the security of the client software, which enforces the permissions. Further, they assumed that as soon as a protected document is opened by a permitted attacker the content can be extracted, since the protection is processed on the client. Concluding they proposed an ERM, which shifts all processes to the trusted server and only sends the current document state in the form a picture to the client. The trusted server processes document modification requests and responses with the "display image" of the information. Thus plaintext data is not revealed at the client.

Schrittwieser et. al. [35] describe techniques and guidelines how Enterprise Rights Management Services, such as RMS and Adobe LiveCycle Rights Management, can be analyzed from the viewpoint of a digital forensic scientist. They state out that the protection of RMS on a client computer is vulnerable, in case an attacker has access to the memory of the computer.

More advanced attacks on Microsoft RMS in Azure and AD have been shown by Grothe et al [9].

Borgmann et al. examined the security mechanisms of several cloud storage services [26]. Therefore, they gave an overview on requirements of users, legal regulations regarding security and state of the art technical security measurements. Among others also end-to-end encrypted cloud storage services like Wuala [33] which bases on Cryptree [8] and TeamDrive [34] are analyzed. As it is conducted for this paper, they observed the network traffic to verify the documented security measurements.

Wuala [8] and Tresorit were also analyzed concerning a MitM attack caused by sharing of files [44]. Wilson and Ateniese observed network traffic and disassembled the client software to review the certification of client keys. As mentioned before, they determined that the central CA of the respective cloud storage provider enables the provider to forge client certificates to obtain protected information during the sharing process.

VIII. Conclusion

The protection of data in terms of information security is a major challenge for modern companies. In case their information, for instance, their files, are accessible from outside, they have to be protected. This is true for consumable files, for example, ebooks and music files, but also for the business area, in case files are accidentally sent to unauthorized people via email or if files are lost on portable storage devices.

In this paper, we shed a light on the use-case, if a company uses a professional RMS system and integrates this services into their workflow. Besides highlighting the threat that comes by integrating another company's cloud solution, we analyzed Tresorit and its DRM module in detail. Tresorit and Tresorit RMS provide confidentiality under the assumption that Tresorit acts "honest but curious" and Tresorit originally claims that they cannot read the content of Tresorit RMS protected files. Due to our systematically in-depth analysis of the Tresorit RMS protocol, we showed, that this is not true.

In summary, our work shows that security in the context of data migration to the cloud is a non-trivial task. Even the strong combination of end-to-end protection with Microsoft's RMS protection can result in unauthorized access. *Trust* is an important aspect for modern Rights Management Services and must be considered wisely. In case a cloud provided does not reveal its protocols or their client software source code, users are bound to their claims. Thus, an independent investigation of pentesters or researchers is necessary to introspect them.

Acknowledgements

The research was supported by the *German Ministry of research and Education (BMBF)* as part of the VERTRAG research project.

References

[1] Ahmad, A., Maynard, S.B., Shanks, G.: A case analysis of information systems and security incident responses. International Journal of Information Management 35(6), 717–723 (2015)

[2] Arnab, A., Hutchison, A.: Digital rights management-an overview of current challenges and solutions. In: Proceedings of Information Security South Africa (ISSA) Conference 2004. Citeseer (2004)

[3] Gasmi, Y., Sadeghi, A.R., Stewin, P., Unger, M., Winandy, M., Husseiki, R., Stüble, C.: Flexible and secure enterprise rights management based on trusted virtual domains. In: Proceedings of the 3rd ACM workshop on Scalable trusted computing. pp. 71–80. ACM (2008)

[4] Grolimund, D., Meisser, L., Schmid, S., Wattenhofer, R.: Cryptree: A folder tree structure for cryptographic file systems. In: Reliable Distributed Systems, 2006. SRDS'06. 25th IEEE Symposium on. pp. 189–198. IEEE (2006)

[5] Grothe, M., Mainka, C., Rösler, P., Schwenk, J.: How to break microsoft rights management services. In: 10th USENIX Workshop on Offensive Technologies (WOOT 16) (2016)

[6] István Lám, Szilveszter Szebeni, Levente Buttyán.: Tresorium: cryptographic file system for dynamic groups over untrusted cloud storage. In: 41st International Conference on Parallel Processing Workshops. pp. 296–303 (2012)

[7] Labuschagne, L., Eloff, J.H.: Electronic commerce: The information-security challenge. Information Management & Computer Security 8(3), 154–157 (2000)

[8] Microsoft: How does Azure RMS work? (Online: 05. 03. 2016), https://technet.microsoft.com/en-us/library/jj585026.aspx#BKMK_HowRMSworks

[9] Microsoft: Microsoft Trust Center // Privacy // You own your data (Online: 15. 02. 2016), https://www.microsoft.com/en-us/TrustCenter/Privacy/You-own-your-data

[10] Microsoft: Windows Azure BYOK (Online: 05. 03. 2016), https://technet.microsoft.com/en-us/library/dn440580.aspx

[11] Microsoft: Windows Azure Key Vault (Online: 05. 03. 2016), https://azure.microsoft.com/en-us/documentation/articles/key-vault-whatis/

[12] Microsoft: Protecting data in microsoft azure (Online: 05 03 2016 August 2014), http://download.microsoft.com/download/0/D/D/0DD8FB12-6343-4A50-80B2-545F2951D7AE/MicrosoftAzureDataProtection_Aug2014.pdf

[13] Moritz Borgmann, Tobias Hahn, M.H.T.K.M.R.U.V.S.V.: On the security of cloud storage services. Fraunhofer Institute for Secure Information Technology SIT (2012)

[14] Mulligan, D.K., Han, J., Burstein, A.J.: How drm-based content delivery systems disrupt expectations of personal use. In: Proceedings of the 3rd ACM workshop on Digital rights management. pp. 77–89. ACM (2003)

[15] Päivärinta, T., Munkvold, B.E.: Enterprise content management: An integrated perspective on information management. In: System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on. pp. 96–96. IEEE (2005)

[16] Park, J., Sandhu, R., Schifalacqua, J.: Security architectures for controlled digital information dissemination. In: Computer Security Applications, 2000. ACSAC'00. 16th Annual Conference. pp. 224–233. IEEE (2000)

[17] Schrittwieser, S., Kieseberg, P., Weippl, E.: Digital forensics for enterprise rights management systems. In: Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services. pp. 111–120. ACM (2012)

[18] Schryen, G., Kadura, R.: Open source vs. closed source software: towards measuring security. In: Proceedings of the 2009 ACM symposium on Applied Computing. pp. 2016–2023. ACM (2009)

[19] Shin, D., Kim, J., Shin, D.: A study on the digital right management of MPEG-4 streams for digital video library. Springer (2003)

[20] Singh, K., Panda, I., Gratia, R., Maharana, B., Nayak, D.K.: Digital right management and its application to library and information science. IJAR 1(12), 878–881 (2015)

[21] Talaat, S.: Azure rights management services. In: Pro PowerShell for Microsoft Azure, pp. 163–177. Springer (2015)

[22] TechNet: What is Azure Rights Management? (Online: 15. 02. 2016), https://technet.microsoft.com/en-us/library/jj585026.aspx

[23] Tresorit: Tresorit DRM White Paper (2015), https://tresorit.com/files/tresorit-drm-whitepaper.pdf

[24] Tresorit AG: End-to-End Encrypted Cloud Storage for Businesses | Tresorit (Mar 2016), https://tresorit.com/

[25] Walter Myers III: Microsoft Azure Data Security (Data Cleansing and Leakage) (Online: 15. 02. 2016), blogs.msdn.com/b/walterm/archive/2012/02/01/windows-azure-data-cleansing-and-leakage.aspx

[26] Wilson, D.C., Ateniese, G.: "to share or not to share" in client-side encrypted clouds. In: Lecture Notes in Computer Science Volume 8783. pp. 401–412 (2014)

[27] Yang Yu, T.c.C.: Enterprise digital rights management: Solutions against information theft by insiders. In: Information Management & Computer Security (2007)