On-Premises Cloud Storage – Security Aspects for Small and Medium-sized Enterprises

Bernd Gastermann, Markus Stopper, Anja Kossik, and Branko Katalinic

Abstract—**This paper demonstrates and outlines the practical implementation of a secure cloud storage solution for small and medium-sized enterprises (SME) by example of an Austrian company. It first introduces some basic principles of cloud storage and eventually focuses on a cloud storage solution considered to be optimal for SME purposes. The paper additionally demonstrates how such a service could be designed considering the particular SME requirements. This work focuses on security aspects and therefore covers how to minimize the system's attack surface and to harden necessary software components in order to defy network attacks. It also discusses ways of protecting data on client- and server-side as well as during network transfers. The paper eventually presents one conceptual approach to implement a proprietary Windows application used for client-side file encryption.**

Index Terms—**Cloud Storage, Data Security Aspects, OwnCloud, SME**

I. Introduction

When applying the latest information technologies, employees of an enterprise can easily work outside their office space or even from home. Enterprises offering working opportunities outside of a controlled company environment are confronted with the difficult task to efficiently and easily provide their employees with necessary documents. Simultaneously, all data need to be securely stored and external third-parties should have no access to potentially sensitive documents. Especially in small and medium-sized enterprises (SME), the awareness, the resources, or even the knowledge of how to sufficiently protect these data in such situations are frequently not available. Sometimes even the field staff itself is responsible for an adequate administration of company documents. In this case they also willingly resort to the broad range of available public cloud storage solutions. The advantage is self-evident: existing services are characterized by high availability, they are integrated into a lot of popular products and, above all, they are easy to use. If no regulations exist on the part of the company, field staff will most likely use such services for future business applications that have previously been adopted for private purposes.

Recently, a new trend towards cloud solutions became apparent in the IT world, which was also shown in a study on usage and distribution of cloud computing in Germany [8]. Many renowned companies promote their cloud services and integrate them deeply into their products, which causes more and more data to be transferred into the cloud. Eventually, in a worst case scenario one could completely lose control over data that have been stored there. Recent events made it evident that private as well as business data in the internet require special protection as was published in various media [10]. Regarding these privacy breaches by intelligence agencies it is especially critical that most of the providers offering cloud services are US-based and are therefore not subject to the same high standards for data security as European companies.

The leaking of intelligence programs caused an upset regarding data security and continuously increased skepticism towards cloud services [7] [8]. Consequently, also Austrian SMEs display increasing awareness regarding data security [4]. Therefore, it needs to be evaluated, in how far present dependencies of field staff on public or American cloud storage services can be reduced and how an equally convenient but more controllable and secure replacement can be provided. Thus, this paper will focus on said quintessential question as well as related topics.

II. Cloud Storage Basics

When talking about the "cloud", this IT-related term usually refers to the computing operations taking place in a "computer and network cloud". The cloud metaphorically represents computer systems and networks, which are either too spacious to be depicted in detail, or which are of unknown structure, or irrelevant to a diagram. The general cloud concept abstracts a whole range of different typologies and implementations. In principle, "cloud computing" and "cloud storage" can be distinguished, the latter being the focus of this paper. Cloud storage typically provides distributed data storage space, which can either be accessed by standardized network or data transfer protocols, respectively, or by a proprietary application programming

Manuscript received January 22nd, 2015. This work was supported in part by the Automation Research & Development (ARD) Department of MKW® Austria & MKW® Slovakia as well as the Department of Intelligent Manufacturing Systems, Institute of Production Engineering, Vienna University of Technology, Vienna, Austria.

Markus Stopper is Professor h.c. at Far-Eastern National Technical University, Vladivostok, Russia and now with the ARD Department of MKW® Slovakia, Prešov, Slovakia (e-mail: markus.stopper@ieee.org).

Bernd Gastermann is researcher associated with the UoAS FH Campus Wien, Vienna, Austria and now with the ARD Department of MKW® Austria, Weibern, Austria (e-mail: bernd.gastermann@mkw.at).

Anja Kossik is researcher associated with the Institute of Organizational Development and Intervention Research – University of Klagenfurt, Austria and now Managing Partner of company Promise International Business Consulting, Prešov, Slovakia (anja.kossik@promise-ibc.com).

Branko Katalinic is Professor at Vienna University of Technology, Vienna, Austria. He is head of the Department of Intelligent Manufacturing Systems, Institute of Production Engineering, Vienna University of Technology, Vienna, Austria (e-mail: branko.katalinic@daaam.org).

interface (API) [1] [6]. Cloud storage services usually employ a vast logical pool of virtualized hard drives, which are redundantly distributed over various servers and data centers in order to increase their availability [6]. This virtualized and distributed architecture has many advantages for the service operator as it increases the system's stability and cost efficiency as well as the elasticity and scalability of the whole service [2]. These implementation details are typically transparent to the user [6].

Cloud storage services can also be classified by different deployment and service models. Common deployment models are the public and the private cloud. These models basically define the relation between client and service, thereby giving some indication by which customer range a certain service can be used. In turn, the terms off- or on-premises describe, under whose influence the cloud service is run [3] [5]. Apart from that, the cloud service model allows to technically differentiate cloud services by the three levels "infrastructure", "platform", and "software", leading to the terms Infrastructure-, Platform-, and Software-as-a-Service (IaaS, PaaS, SaaS). Cloud services can be allocated to one of these levels according to their technical design.

Even though the aforementioned principles are essential for the understanding of subsequent sections of this paper, their scope makes an in-depth discussion impossible at this point. Consequentially, the paper can only refer to literature like [9], where these topics are covered in much more detail.

III. Implementing Secure Corporate Cloud Storage

SMEs generally have two options for implementing a suitable cloud storage solution in the company environment: either one of the numerous off-premises services that are readily available on the market or an appropriate cloud service platform within the company (that is, an on-premises solution) can be used. Apart from special applications, SMEs can usually refrain from using IaaS or PaaS services, as these are typically associated with major administrative workload as well as the development or installation of higher software levels. Therefore, the choice is mostly limited to the SaaS service model, which is based on predetermined hard- and software provided by its operator.

As the definition of SMEs comprises an extensive range of different companies, their respective requirements for a cloud storage solution can vary considerably. Therefore, it is impossible to give a general recommendation for specific cloud storage solutions that are ideally suited for SME environments at this stage.

In order to gain deeper insight into the utilization of one of the available cloud services, a specific product was singled out in agreement with the company where the cloud storage solution was implemented. Based on this product a possible approach can be demonstrated, how the secure operation of cloud storage in a company environment can be achieved. Considering the specific company requirements and conditions, a cloud software platform marketed under the name "ownCloud" was selected and implemented on existing on-premises infrastructure. Therefore, the following chapter focuses on ownCloud and sheds a light on various security-related aspects. As a software solution is only as secure as the environment in which it is applied, not only ownCloud but the entire subjacent software platform serving as the basis for its operation are subsequently considered.

A. Company Requirements

In order to avoid the problems outlined in the introduction of this paper, and to provide field staff with a secure option for the storage of company data, an appropriate storage system was to be implemented. This section concentrates on the specific company requirements leading to the decision to use ownCloud as the software platform for the cloud service. The following general requirements were identified:

Functionality: The functional requirements were mainly limited to data storage, but the option for automatic synchronization of files on multiple devices for offline availability was considered important. The synchronization was supposed to happen transparently for the user and to be possible within the company network as well as via the internet. Other functionalities like calendar or contact management have not been required. An additional criterion, however, was the support of at least 150 users and a centralized user account administration.

Security: Naturally, the solution has to support basic security features like access management. As security is an important issue, the encryption of data during transfer by a secure protocol is indispensable. Encryption of server-side user data, however, does not represent a crucial criterion.

Costs: For many companies, especially those of smaller size, product or license costs should not be disregarded. In general, the one-time and running expenses are supposed to remain within certain defined boundaries, which will not be subject to further discussion herein. Thus, cloud solutions that are free of charge are preferred.

Provider Independency: Additional prerequisites for the selection of a storage solution are the independency from third-party providers and the free choice of the data storage location, respectively. In order to lower the risk of losing data control, and as the infrastructure as well as available IT resources necessary for operation are already available, the service should intentionally be operated on-premises.

Platform Support: As a major part of the existing software infrastructure is based on Microsoft technology, it is essential that the implemented cloud platform supports Windows as operating system (OS) on the client as well as on the server. Therefore, the server software needs to run at least on a Windows Server 2008 R2, and synchronization programs have to support Windows 7 or higher. This requirement aims to keep the company's current system landscape homogenous and administrative workload to a minimum. Apart from that, data access has to be possible via a current web browser irrespective of the end device. Common smartphones and tablets also need to be supported.

Expandability: Finally, the general expandability of the applied solution is of major concern. This is related to both hardware and software level. Accordingly, storage capacity has to be expandable just like the provided functionality (e.g. via plug-ins). Consequently, open interfaces (i.e. APIs) have to be available that can be used to add further functionality upon request. The support of the open and standardized WebDAV protocol is expected in this regard.

B. *Software Architecture*

In order to establish an on-premises cloud storage solution running on the company's own hardware, a cloud storage software is necessary that fulfills the specified requirements. A market analysis of currently available cloud storage solutions came to the conclusion, that the freely available community edition of the open-source software called ownCloud is able to meet all of the previously exemplified criteria. Therefore, ownCloud was selected as the cloud storage platform used for the implemented on-premises service. The ownCloud software platform consists of a web application that runs in a web server on top of existing hardware and is based on the platform independent PHP scripting language. Its operation requires an OS, a web server, a PHP runtime environment, and a suitable object-relational database management system (DBMS) used to store application settings and user accounts.

As suitable hardware is already available within the company, the aforementioned software components are installed and implemented as a virtual machine (VM). The following Fig. 1 depicts the technical software layers of the cloud VM as well as of the subjacent virtualization server.

C. *Hardware Architecture*

The key component of the used hardware infrastructure are multiple physical servers that are combined to form a logical cluster used for virtualization. This cluster allows for several VMs to run in parallel and independent of the underlying physical hardware, thereby enabling efficient use of the resources. The server on which the ownCloud-based cloud storage solution is implemented is operated as such a VM, simultaneously running with other unrelated VMs on abovementioned server cluster. This cluster is designed redundantly, in order to compensate for malfunction of individual hardware components and to enable dynamic load balancing. The storage system, on which the individual VMs as well as the uploaded user data are stored, displays a similar redundancy that also includes an uninterruptible power supply. However, further technical details are not discussed here, as these specifications are irrelevant for the implementation of the cloud storage service.

In regard to the network environment the ownCloud VM is located in a so called demilitarized zone (DMZ), which, due to security reasons, separates publicly accessible and therefore attackable services from the remaining and more sensitive network areas. A central firewall isolates the DMZ from these underlying company network areas.

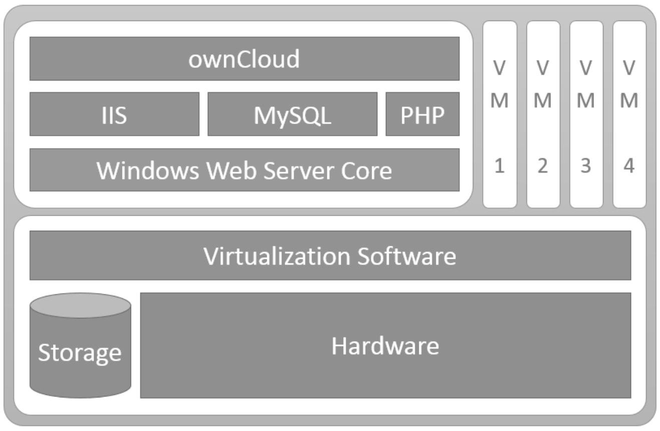

Fig. 1. Composition diagram of the cloud server and its base infrastructure.

D. *Reduction of Attack Surface*

As the use of Windows was a prerequisite, the first step was to install and configure the OS of the cloud system's VM. Already with this first installation step it is recommended, to deliberately consider how the potential attack surface of the system can be minimized. The term "attack surface" denotes the entirety of all services that are exposed to the outside of a system and are therefore potential targets for an attack. The more services are targetable from the outside, the bigger the attack surface of a system and the higher the probability of success for such an attack. Thus, this surface is to be minimized by removing non-essential services, which not only leads to an increase of security, but also reduces the system's overall complexity and resource consumption [4].

Considering this security aspect, a minimal installation of the software components required for ownCloud is reasonable. Consequently the core version of the Windows Web Server 2008 R2 is applied. In Windows Server Core many features that aren't needed for server systems, like end-user applications and services as well as most of the graphical user interface have been removed. Therefore, administration of the server is primarily performed via the internal command-line interface or by various tools that can remotely connect to the server via the internal network, respectively. The reduction of the attack surface is not only crucial for the OS but for all installed software components. All components that are essential for the operation of ownCloud have to be securely configured in this regard. These components comprise a web server, the PHP application library and a DBMS. The Windows-integrated Internet Information Services (IIS) has been installed as a web server and MySQL as a database system. Each of these components must be hardened against network-based attacks by use of several configuration parameters. Due to the complexity and scope of this topic, the installation and configuration steps of the individual components will not be discussed here. Further information can be found in the respective product manuals. As soon as PHP, MySQL and IIS have been carefully secured, ownCloud can be installed.

IV. Protecting Data on Client, Server, and Transport

This main part of the paper provides an in-depth discussion of three security relevant aspects of the presented cloud storage solution. It will give insight into additional technical measures that can improve data security. Three areas can be identified, where such measures can be applied: the encryption of data-at-rest on the server, the protection of data during transfer, and the client-side encryption of data performed directly on the user's end device before they are eventually transferred to and stored on the cloud server.

A. *Server-Side Data Protection*

The first data protection measure solely and directly addresses user data on the cloud server itself. The server represents the central component of the system and is responsible for storing all uploaded files. For this scenario, server-side data protection is mainly supposed to ensure confidentiality. Other aspects of data protection on the server comprise hardware redundancy or the physical protection of the IT infrastructure but these aspects are not taken into consideration in the scope of this paper.

As data are usually filed on the server as plain text, they can – under certain circumstances – theoretically be read out and manipulated by an attacker. Server-side encryption of all stored user data represents a possible countermeasure. If data are only stored in cipher text, this will increase the effort for any entity with malicious intent to remain unnoticed when reading out or modifying these data. Server-side encryption is typically transparent for the user. This means that the user will not notice any difference to unencrypted data storage, when using the service. In case documents are uploaded, the system takes care that they are automatically encrypted and stored as such on the data medium. In the opposite direction, when documents are retrieved, the system reads out the encrypted data from the hard drive, decodes them with the respective key, and then sends the document back to the user. This example also demonstrates a weakness of the system: the key and the encrypted data are both accessible by the server. Users typically do not have any control over the key or the applied cryptographic method and are therefore dependent on the confidentiality and reliability of the service and its administrative environment. From the perspective of the user, this is related to the potential risk that administrators or other persons with respective authorization can retrieve data from the server before they are even encrypted or are able to decode them by use of a system-wide key. Furthermore, insufficiently secure cryptographic methods could be applied without the knowledge of the user. In regard to server-side encryption, the difference has to be made, whether it was performed with an individual user-specific key or a universal key valid for all filed data, respectively. Due to several security related reasons, the former would be preferable, because many individual keys increase the effort for the attacker and could also prevent compromise of the entire system, in case a key is falling into the wrong hands.

Several methods for the implementation of server-side encryption exist. One option could be the use of third-party software or of features provided by the OS, such as “BitLocker” or the built-in encryption mechanism of the NTFS file system. Both procedures address a level close to the hardware so that the protection is directed primarily against the readout by attackers with direct hardware access or other unauthorized user accounts at the level of the OS. However, this protection is turned ineffective in case the attacker is logged in via the administrator account. Thus, a different approach is the encryption on the application level. An optional ownCloud extension is specifically intended for this purpose, allowing server-side encryption on application level. There are no functional limitations related to the activation of this feature, but the encryption is restricted to the file contents only, while file names remain universally readable. Depending on the activation of other features, which could use various caching techniques, it can occur that not all information gets properly encrypted.

The application of server-side encryption inside a controllable company environment often does not provide a clear benefit. In case data (like private data) deserve special protection and should remain inaccessible even to the system administrator, it can be feasible to activate this feature. If the use is restricted to business documents that are potentially freely accessible for the administrator on other network drives, server-side encryption is of no benefit as it increases the overall administration effort and represents an additional source of error. In a worst case scenario, where no documented error recovery method for encrypted data was defined, data can potentially get lost or the import of an earlier backup can be complicated.

B. *Transport Encryption*

In this regard, transport encryption is a completely different issue. The protection of data during transfer is absolutely indispensable in any case. This is caused by the fact that storage services usually have to be accessible not only via the internal company network but also via public networks like the internet. Access via public networks is always associated with the risk that data are intercepted or read out during transfer. This issue is addressed by Hypertext Transfer Protocol Secure (HTTPS), which is very similar to conventional HTTP but uses Secure Socket Layer (SSL) or Transport Layer Security (TLS) for authentication and encryption of communication. Thereby it is of significance, which of the two encryption protocols has been applied. For security reasons the use of TLS in version 1.1 or 1.2 in combination with modern browsers is advisable. Older protocol versions like TLS 1.0 or its predecessor SSL 3.0 should only be used due to compatibility reasons. HTTPS is thought to ensure that the counterpart is really the computer it is supposed to be, and that sensitive data are not easily read along by others.

The configuration of the web server for the application of HTTPS is trivial, but it takes a valid certificate issued by a certificate authority (CA). Provided that such a certificate is available, the web server is switched to use the HTTPS protocol in just a few steps. Even though communication is already basically protected by this measure, further adaptations like prioritizing of the system’s preferred cipher suites are still recommended. Cipher suites describe a standardized set of cryptographic algorithms, which are used for different purposes during communication by SSL/TLS, respectively. A suite comprises algorithms related to four areas: key exchange, digital signature, hash-functions, and data encryption itself.

A multitude of such cipher suites with different characteristics does exist. The broad range of such suites is mainly due to the fact that some of them ensure compatibility with older browser versions, and that others apply procedures, which are more secure, faster, and less prone to certain attacks than the rest. A feature that gets more and more attention from cloud services, is “Perfect Forward Secrecy” (PFS). It mainly describes a quality of key exchange protocols, which are able to guarantee that individual session keys are not affected, even when the private long-term keys, from which individual session keys are derived, have been compromised. In other words: even compromised long-term keys do not provide attackers with enough information to recreate individual session keys. These have to be attacked directly, which will only disclose the respective session, while others remain unaffected.

As previously stated, cipher suites can be distinguished by specific features like speed, resource consumption, and security, but not all of them offer PFS. On the other hand, not all available cipher suites are supported by all OSs or web servers. Therefore, they provide a list of supported cipher suites as well as a priority list that defines, which of them should be applied for client communication. Client and server negotiate the optimum set based on their respective cipher suite preferences, if a match can be found at all. The problem is to find a balance between the support of older browsers and systems on one side, and potentially insecure algorithms on the other. Windows Server 2008 R2 itself defines a set of default cipher suites, which are enabled right after the activation of HTTPS. But this default list is associated with several problems: for instance, cipher suites without PFS support are preferred. Additionally, algorithms are used that are not considered to be sufficiently secure and state-of-the-art by today’s standards. Even cipher suites that refrain from using any encryption at all can be found.

In controlled company environments and in the case of a specific application situation like this, it is easy to judge, which platform and browser need to be supported by the cloud service. The list of cipher suites supported by the ownCloud server can therefore be adjusted accordingly. Knowledge of potential weaknesses or discouraged algorithms can also be considered during this process.

Generally, the application of key exchange protocols based on Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) algorithms are recommended. Both support PFS but in regard to performance, ECDHE should be preferred. Concerning compatibility, it is advisable to use cipher suites based on the fast RSA procedure for key exchange, which does not support PFS, however. RSA can also be used as a signature algorithm. Alternatively, Elliptic Curve Digital Signature Algorithm (ECDSA) can be applied. Considering hash functions, the newer SHA-2 procedures should generally be preferred, as the SHA-1 algorithm, according to current technical knowledge, does no longer provide sufficient protection and cannot be judged resistant enough against collision attacks.

C. Client-Side Encryption

Client-side encryption offers a very high level of protection as user data are encrypted directly on the computer of the user, before they are even uploaded to the server. Ideally, it allows countering the potential security issues of the two aforementioned approaches. The confidentiality of user data is therefore solely related to the confidentiality of the used computer and to the security of the used encryption algorithm, because the actual data remain encrypted even when the transport encryption is circumvented. The same is valid for data-at-rest, as neither the administrator nor other persons usually have the possibility to decode user data, as long as the applied key is securely guarded by the owner. As such, a distinctive feature of client-side encryption can be pointed out: management of the secret key is the exclusive responsibility of the user, who has to ensure that the key is not getting compromised. This approach therefore provides data confidentiality and privacy, which is basically independent of the encryption used during transmission or on the server.

Despite these advantages, client-side encryption is less suited for company environments. This line of argument corresponds to the one followed for server-side encryption. First of all, no sensitive or private documents are stored on the cloud server that have to be shielded from the view of the administrator. This approach also complicates assistance in case of problems or support requests. Apart from that, client-side encryption leads to extensive functional consequences: as the encryption is mostly performed with symmetric encryption algorithms due to several reasons like speed, practicability, etc., the ownCloud feature for document sharing loses its practical function. If encrypted data were shared with others, the private symmetric key would have to be passed down, which in turn would inevitably lead to other security problems. Data access via the web interface of ownCloud would also be complicated, as data would have to be manually decrypted after download and manually encrypted before upload using an appropriate tool. In a worst case scenario, this approach could also lead to data loss, if users lose their private key or forget their password, as the administrator would be unable to restore access. In total, all these reasons make client-side encryption typically unattractive for business environments. Although it is not suitable in the scenario described herein, there are of course situations where it may be more appropriate. The following section describes one such scenario and presents a concept on how to implement a basic file synchronization application for that case.

Concept for a Proprietary File Synchronization Client

Client-side encryption is especially reasonable, when the cloud service is run by a potentially untrustworthy third-party. In such cases, where even server-side encryption may not be sufficiently trustworthy, client-side encryption could prove helpful. The approach could also be desirable in a different scenario for business environments: in order to accelerate the acceptance of a new cloud storage service by the employees, or to make the use of public storage service for company documents increasingly unattractive, a hybrid encryption technique could be pursued on the cloud storage service. Employees could then store business as well as private documents on the provided service. However, both sections would have to be logically separated, which could be easily achieved by means of a predefined subdirectory in the employee’s storage location. This subdirectory would be exclusively dedicated to private use. Consequentially, the company would have to provide a proprietary desktop synchronization application that automatically encrypts all files of the private folder with an individual user-chosen key, whereas all other business documents outside of this folder remain unaffected by the user’s encryption.

How could such a synchronization application for a scenario like this be technically realized? It is difficult to give an answer to this question that is generally valid, as the technical characteristics and limitations of the respective platforms need to be considered. Nevertheless, one common starting point could be the standardized WebDAV protocol, which is supported by ownCloud and which allows to perform fundamental data operations independently from the respective platform. As far as Windows is concerned, such a software could be implemented using the .NET Framework.

The following Fig. 2 gives an overview of the software layout of such a synchronization application based on the .NET Framework. The application basically comprises five components. The foundation of the application is provided by the “FileSystemWatcher” class of the .NET Framework, which allows to receive notifications about changes on the file system within a specified directory. On the other side the WebDAV module is located, which serves as an interface to the cloud service. This module can either be implemented from scratch based on the open WebDAV standards or an available application library can be used alternatively. Additionally, a cryptography module is required, which encrypts and decrypts user files transparently for the user. Similar to many other frameworks, the .NET Framework already provides an appropriate module that can be used for this purpose. In order to detect file system changes even when the application is not active, a database is necessary. Within this database, which could be SQLite for example, all existing data of the monitored directory as well as metadata like modification dates, file sizes, or hash values are stored. If the user changes a file at a later point, it can be easily detected retrospectively by the application. Needless to say, a final central program logic is also required to connect and control all the previously mentioned components.

The basic program sequence should present itself as follows: when the application is started for the first time, it has to search the directory, which is predefined by the user for existing files and save their relevant metadata in the internal database. Simultaneously, the user directory on the server has to be scanned via WebDAV. It is assumed that no data are stored on the cloud server at the first time start. In case the user now creates a file in the monitored directory, the “FileSystemWatcher” immediately reports a change, which then has to be processed. Dependent on whether the file is stored in a private subdirectory, it then needs to be encrypted or not. In the case of a business document that is located in a monitored directory outside of the private subdirectory, the file is directly uploaded to the server. At the same time metadata need to be constantly kept up-to-date in the application’s database. In the case of a private file in

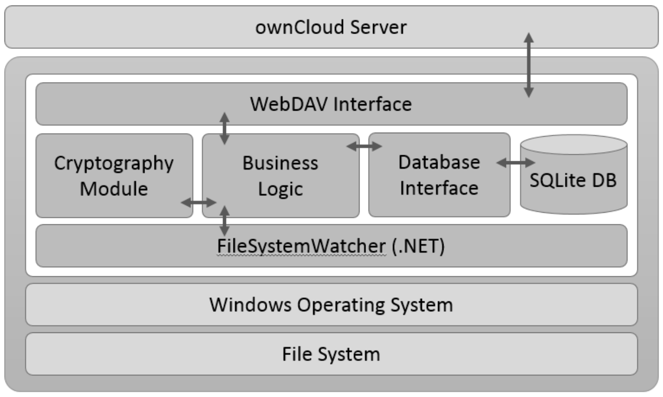

Fig. 2. Conceptual architecture of a proprietary synchronization application.

the respective subdirectory, it is first encrypted by the application and is then transferred to the server. If the computer is now shut down and files are modified on a different device, the application then must be able to recognize modified or newly added files upon the next startup. This happens by comparison of the information stored in the database with the files stored on the server or the local hard disk. Here, file names, hash values, file sizes, or similar information can be compared. Modified or new files need to be downloaded and deleted files need to be removed. Conflict resolution in case of contradictory information is a general problem of data synchronization. At that point, the latest file date could serve as a criterion for the decision of file replacement. However, a generally valid solution for this problem cannot be suggested here. If new or changed files that belong to the private area of the user are found on the server, they can be downloaded, decrypted, and eventually be stored in plaintext in the private subdirectory. This, however, can only take place, if the user entered his personal key upon starting the program. If the key was not entered, data can be downloaded but not decrypted. This can still happen at a later point, though. In this context, the specific approach to key management needs to be clearly defined. Should the key be entered only once and is then stored on the hard drive, or is it necessary to enter the key with every application start? As the answer to this question is dependent on the individual requirements, no specific solution will be suggested in this work.

The approach outlined in this paper represents an option, how client-side encryption can be implemented in a transparent way. Of course, numerous additional details need to be clarified in this regard and no generally valid solution can be presented as such a solution is, among others, dependent on the respective platform, the type of key management, and whether it is feasible at all to store unencrypted data on the user’s local hard drive. In the approach presented here only encrypted data are stored on the server and remain unencrypted on the local PC for better convenience. However, this mode of operation is not always desirable so that alternatively manual decryption would have to be initiated by the user. The answers to this and other questions remain solely in the sphere of influence of the company or the software developers, respectively.

V. Conclusion and Outlook

The purpose of this paper is to provide an overview over some basic but important security aspects that have to be considered when implementing a secure cloud storage service for SMEs. Companies planning to employ such a solution within their business environment first need to determine which type of service fits their respective requirements best. However, no universally applicable recommendation can be given here, as the requirements for SMEs are typically diverse. Thus, the focus of this paper was on a particular Austrian SME. Based on this company’s requirements, it generally seems more appropriate for SMEs to use SaaS-based cloud storage solutions than PaaS or IaaS-based services. This conclusion is drawn from the fact that the latter two usually require increasing administrative and financial efforts when compared to SaaS. Another factor not to be ignored is the type of deployment. The company discussed in this work, for example, explicitly required to establish an on-premises cloud storage service operating on existing infrastructure. Of course, this may not be the case for other companies so a different type of deployment or service model could prove more appropriate here.

For the sake of comprehensibility, the paper thus introduced the respective company requirements that eventually led to the selection of ownCloud as cloud software platform and also briefly explained how it was implemented on readily available hardware infrastructure. However, as the primary focus of the presented solution is on security, the paper subsequently describes why the reduction of the system's attack surface is essential and how this was achieved. The system's overall security ultimately depends on the security of all its incorporated software components. Consequently, the minimalistic and more secure core version of Windows Web Server 2008 R2 was used as the basis for the implemented system. Additionally, it was attempted to harden all other components that are indispensable for the operation of ownCloud against potential network-based attacks.

In the course of this work it was also evaluated, which other measures are capable of improving the overall data security. This comprises client- and server-side encryption of user data but also the protection of data during transfer. Even though the former generally do not seem to be of much benefit for a private on-premises service, the use of HTTPS is an indispensable measure nowadays. Considering transport encryption, a major potential for optimization was detected in regard to PFS and potential attack vectors against employed security protocols. Despite being less useful in this specific business case, client-side encryption still represents a powerful way of protecting cloud data, for example when using a third-party service. The paper thus briefly outlined one possible application for client-side encryption and showed how to develop a basic file synchronization software for this purpose based on Microsoft's .NET Framework.

The cloud storage solution presented herein is not perfect by far and this discussion raises no claims to completeness. In this regard, the whole topic is much too complex, so that only a few important software-related security aspects could be outlined here. This, however, poses a major limitation of this work as it is necessary to consider the whole company environment in order to build a secure system. As such, employees, company guidelines, and other IT systems as well as the physical security of the infrastructure must not be ignored. Some weaknesses of the presented solution also need to be mentioned: first of all, the applied version of the OS is slightly outdated by now. Newer versions are recommended although older software is not insecure as long as it is serviced and supplied with security updates. Unfortunately, the provided virtualization infrastructure did not support higher versions of the OS, so problems could eventually arise in the medium-term when support for this product is discontinued and security flaws remain untended. Furthermore, the intrinsic problems of the selected on-premises operation are obvious: all deficits discussed so far demonstrate that, apart from the investment in required infrastructure, the operation of one's own cloud storage service is related to continuous administration effort.

Overall, the implemented cloud storage service provides important benefit for the discussed Austrian SME. It provides field staff with an appropriate and secure file management tool for internal and external access, while at the same time, the company retains control of all sensitive data as they are stored on server hardware within the company on Austrian territory. Although the implementation of a secure cloud storage system for this company was considered essential and is completed by now, further development of this cloud service has to continue. Routine service is necessary in order to keep system security up to date with the latest attacks. In order to validate the security of the implemented system an extensive penetration test has to be performed by an independent and professional entity. Such a test alone can verify system security on a deeper level and can detect potential vulnerabilities of the software components. Based on the outcome of the penetration test, further actions or research may be required to improve certain security aspects of the implemented cloud storage system and its IT environment.

References

[1] R. Seiger, S. Groß, and A. Schill, SecCSIE: A Secure Cloud Storage Integrator for Enterprises, in *Proceedings of the 13th IEEE Conference on Commerce and Enterprise Computing (CEC)*, pp. 252-255, ISBN 978-1457715426, DOI 10.1109/CEC.2011.45, Institute of Electrical and Electronics Engineers (IEEE), 2011

[2] T. Sasidhar, P. K. Illa, and S. Kodukula, A Generalized Cloud Storage Architecture with Backup Technology for any Cloud Storage Providers, *International Journal of Computer Application (IJCA)*, Volume 2, Issue 2, ISSN 2250-1797, RS Publication, 2012

[3] G. Kulkarni, R. Waghmare, R. Palwe, V. Waykule, H. Bankar, and K. Koli, Cloud Storage Architecture, in *Proceedings of the 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, pp. 76-81, ISBN 978-1467345491, DOI 10.1109/TSSA.2012.6366026, Institute of Electrical and Electronics Engineers (IEEE), 2012

[4] R. L. Krutz and R. D. Vines, *Cloud Security: A Comprehensive Guide to Secure Cloud Computing*, ISBN 978-0470589878, Wiley Publishing, Indianapolis, IN, USA, 2010

[5] T. Erl, Z. Mahmood, and R. Puttini, *Cloud Computing: Concepts, Technology & Architecture*, ISBN 978-0133387520, Prentice Hall, Upper Saddle River, NJ, USA, 2013

[6] M. Borgmann, T. Hahn, M. Herfert, T. Kunz, M. Richter, U. Viebeg, and S. Vowé, *On the Security of Cloud Storage Services*, Fraunhofer Institute for Secure Information Technology (SIT), SIT Technical Reports, SIT-TR-2012-001, ISBN 978-3839603918, ISSN 2192-8169, Fraunhofer Verlag, Stuttgart, Germany, 2012

[7] Microsoft, Press Conference "New Deal for the Digital World": Location Factor Digital Trust – Solutions for Austria to steer clear from the current confidence crisis [Online], Available: http://www.microsoft.com/de-at/news/Press/2014/Apr14/New-Deal-fur-die-digitale-Welt.aspx

[8] KPMG and BITKOM, Cloud-Monitor 2013: Cloud Computing in Germany [Online], Available: http://www.bitkom.org/files/documents/Studie_Cloud_Monitor_sec.pdf

[9] B. Gastermann, M. Stopper, A. Kossik, and B. Katalinic: "Secure Implementation of an On-Premises Cloud Storage Service for Small and Medium-sized Enterprises", *Procedia Engineering*, 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, ISSN 1877-7058, Elsevier Ltd., submitted for publication.

[10] D. Vaile, The Cloud and data sovereignty after Snowden [Online], Australian Journal of Telecommunications and the Digital Economy, Vol. 2, No. 1, pp. 31.1-31.58, ISSN 2203-1693, March 2014, Available: http://search.informit.com.au/documentSummary;dn=187308443693292;res=IELBUS