Anna GROCHOLEWSKA-CZURYŁO[1], Marek RETINGER[1]

Poznan University of Technology (1)

doi:10.15199/48.2018.03.33

Secure cloud services — extended cryptographic model of data storage

Abstract. *This paper presents secure data storage model based on cloud storage services, which improves basic security aspects such as confidentiality, integrity and availability. Extended model introduces several improvements, where the most important are RAID techniques, "off-line" mode and node distribution mechanism. The document briefly describes additional network node called "proxy node" and contains the system's security considerations.*

Streszczenie. *Niniejszy dokument prezentuje model bezpiecznego składowania danych oparty o magazyny chmurowe, dzięki któremu możliwe jest polepszenie podstawowych aspektów bezpieczeństwa takich jak poufność, integralność oraz dostępność. Rozszerzony model wprowadza wiele usprawnień w tym technikę RAID, tryb „off-line" i mechanizm rozproszenia (redundancji) węzła. Ponadto zdefiniowano dodatkowy węzeł sieci nazwany „pośredniczącym" oraz zawarto rozważania w temacie bezpieczeństwa systemu.* ***(Usługi chmurowe — rozszerzony model bezpiecznego składowania danych)***

Keywords: cloud storage, cryptography, security, RAID, proxy node
Słowa kluczowe: magazyny chmurowe, kryptografia, bezpieczeństwo, RAID, węzeł pośredniczący

Introduction

Cloud storage services are getting more and more popular. Many people use it for storing private files such as photos, videos, text documents or confidential data. What is obvious, the most important parameters for them are capacity, performance and availability. Performance of cloud services is a subject of many researches. We can find documents with information about cloud storage services efficiency and performance in general [16] or in particular context [10]. In 2012 and 2014, the most popular cloud storage services were Microsoft SkyDrive (now OneDrive), Dropbox, iCloud and Google Drive. We have not found any official document describing the services market share after 2014. Unofficial sources — mainly technology portals — give different information depending on target groups. We averaged the data and extended the group of providers above by Amazon S3 service, which nowadays became very popular. We have not included price as "*parameter*", which is also very important, due to the fact that there are many free solutions on the market and they are sufficient for most users. We aggregated basic information about the services in table 1.

Table 1. Common cloud storage services comparison.

	OneDrive	Dropbox	iCloud	Google Drive	Amazon S3
Free plan	Yes	Yes	Yes[1]	Yes	Yes[2]
Free storage	5 GB	2 GB	5 GB	15 GB	5 GB
File size limitation	10 GB[3]	20 GB[4]	50 GB	5 TB	5 TB
File extension limitation	No	No	Yes	Yes	No
Paid plan(s) available	Yes	Yes	Yes	Yes	Yes

In the document, we have not presented the paid plans — each provider offers several different payment models depending on customer needs. The models can be modified without notice.

Cloud storage security is a less popular subject than "*performance*". Providers share just a little bit information about system/data security. We were able to find just a few papers dealing with data protection and involving directly cloud storage [7][20] In our opinion, the best analysis was given in [4]. We very much agree with the conclusions of the paper, that we should all be aware, because the chosen cloud service might not be as secure as we would like it to be and our data might not be properly protected. Our brief analysis of a document concerning cloud security [19] is given later in this paper.

Before presenting the cloud storage encryption models, we describe cloud storage technology in general. Cloud storage is still not formally defined. Sometimes people confuse cloud computing with cloud storage which is incorrect. Formally, definition of cloud computing was presented by NIST [12] and in few words we could summarize it,however not quite accurately, as aggregation of several cloud service models. One of them is a SaaS model (*Software as a Service*). In this model user just connects to the remote application and uses it. He does not care about installation, newer versions and local system resources [8]. The model is also important, because cloud storage is provided as SaaS model. Now we can see that cloud storage is a narrower term than cloud computing. This is one of the main reasons why we should look for some methods to protect our data — typical storage solutions offer disk space, but do not offer any additional (cryptographic) tools.

Currently, the two most popular cloud storage encryption models are a classical server-side encryption model and a mixed user-side/server-side encryption model. In both cases client sends file(s) to the server using a secure connection (e.g. SSL/TLS protocol [14]), however there are some differences. For simplification, we will use the term "*file*" for any kind of transmitted data.

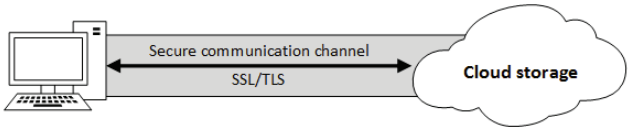

Fig. 1. Server-side encryption model.

In server-side encryption model shown in the figure 1, client sends the original file without using any extra cryptographic techniques. So the way the data is stored in the cloud

[1] Requires at least one Apple device.
[2] 12 months trial.
[3] There are no limits for files send with dedicated software.
[4] See footnote 3.

is very important. The user does not know any keys, and he should be able to download the file any time, from any place in the world. There are two possibilities. First one, the data is not encrypted or it is encrypted by a single, global key. The second possibility is that the provider generates a separate key(s) per file. We can see that, in both cases the cloud administrator can easily access the original data and get to know its content. In general the user can only trust that data was thoroughly secured.

A solution where user is aware of the danger and encrypts data before uploading is much better from the security perspective (figure 2), but it goes against the main idea of the cloud storage — accessibility and mobility.

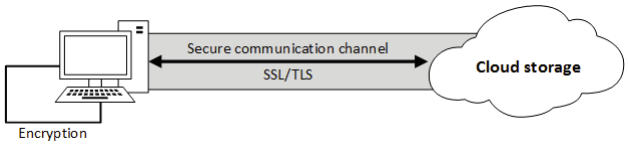

Fig. 2. Integrated user-side and server-side encryption model.

If the user would use good quality keys (pseudo-randomly generated), he would have to store them on some storage media. Lack or loss of the media means that the data will be no longer available or hard to access. Of course, one could store the keys in a different cloud storage system but we should remember, that the companies can collaborate and our secured data can be still obtained by third party.

We found just a few system models, which offer enhanced security of cloud storage and introduce additional network node(s). The first one is RACS (*Redundant Array of Cloud Storage*) described in [1]. The authors presented complete system providing tools for data management, increasing availability and reducing service cost in general. The system was well described. The researchers have shown redundant solution (multiple RACS proxies model), which makes the system fault-tolerant but the model does not increase data security in any area which is important from our point of view. Frauhofer Institute for Secure Information Technology [15] has proposed another solution improving cloud storage security. Authors have designed an advanced system offering secure data storage in clouds. The main idea of the gate is *„to connect any kind of application and backup software with any cloud storage service"*. The system offers built-in key management component, which greatly improves the value of the solution. The system could also be transparent to the users and software. However, this transparency can potentially compromise stored keys and in turn it can enable an adversary to obtain user's data. The last document [17] describes the system based on RAID technique which improves the level of data security. The strength of the solution is working implementation, which has shown real performance benefits. The encryption process was presented briefly and it is different than conceptual model developed by us.

System overview

Our system is based on the classical encryption model. It is impossible to change protocol or cryptographic technique provided by publicly available cloud systems. Every "*new*" system has to be compatible with it. Our solution introduces one, single node (called *proxy node*), which is a semi-transparent gate connected to client's cloud storage account (figure 3). From user's point of view, system looks like a normal web application. In contrast to solutions described in section Introduction, we combined advantages of the RAID technique with the secure data storage. In solution presented in this paper keys are not stored in the node database as shown in [15]. The user should "*remember*" the passwords or store them securely. Because of this the system is not transparent but the keys cannot be compromised, even if attacker obtains access to local database content.

Our proposed node can be used in two different ways:

- as a publicly available system — users can use it from any place in the world, however it is less secure solution;
- as a private system — users can use it from chosen networks which improves system's security; external communication with the node requires a VPN [11] (*Virtual Private Network*) service.

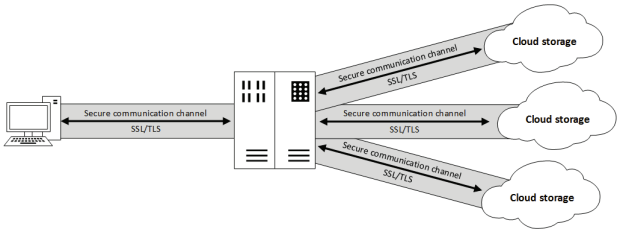

Fig. 3. Proxy node-based encryption model.

One of the most important (not cryptographic) system's features is improved availability aspect. The system uses cloud services as multiple local hard disks working in a chosen mode. It allows the system to use the techniques known from RAID [18] (*Redundant Array of Independent Disks*) mechanism. For example, if data will be stored using RAID level 3 and one of independent provider's systems will not be available, we are still able to obtain our data. "*Splitting*" could also increase the data security level — if user stores split data on clouds in Russia, China and United States of America, the providers have to collaborate, which is highly unlikely.

Proxy node

Proxy node is a semi-transparent gate, which allows user to manage their resources in cloud storages. User uses dedicated software, similar to those offered by cloud providers with one difference — our node supports multiple services and is able to use them as virtual (local) disks. Node offers on-line and off-line modes. The first mode means that data are fully stored in chosen proxy node database. If the database is corrupted or node is inaccessible, user will lose their data. The solution is the second mode called "*off-line*". It allows user to use any proxy node in the Internet but it requires storing small configuration file on the client side. This mode is described in section Off-line mode.

Before user can upload or download a file, he needs access to an account on the proxy node. There are two ways:

- account will be created by system administrator — the best solution for private systems such as companies or a single user, who cares about security;
- account will be created by user himself — option for publicly available systems.

The registration process in public systems is described below.

1. User connects to a proxy node by secure communication channel (SSL/TLS protocol).
2. User fills in required fields with data such as user name, password, e-mail address.
3. System registers the user's account. Client and server generate individual asymmetric key pair. Server sends his public key to the client; client checks once more server certificate and sends his own public key.
4. The process of account registration is finished.

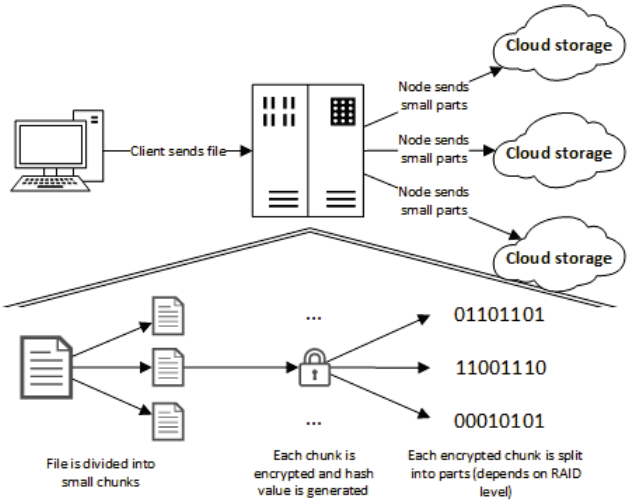

Fig. 4. Simplified upload phase.

In the following part of the document we describe upload (figure 4) and download (figure 5) phases. Both are very important and show how the system works. Each phase is divided into two sub-phases: from user's and server's point of view.

Upload phase — communication between user and proxy node

1. User logs into an account.
2. User chooses upload parameters:
   (a) which cloud services will be used,
   (b) which RAID level will be used,
   (c) enters the protection key,
   (d) which mode will be used (*on-line* or *off-line*),
   (e) optional:
   - sets size of single file chunk[5];
   - chooses hash algorithm for digital signature purposes.
3. User confirms file transfer by entering account's password.
4. User waits for the end of the process. He follows the instructions that appear on the screen and finalizes the whole process.

Upload phase — communication between proxy node and cloud storage systems

1. User takes all required actions, described above.
2. System divides file into N parts, depending on single chunk size. If the size of the last chunk is smaller than single chunk size, it is padded with required bytes[6].
3. System generates random salt value, which will also be used for the chunk name.
4. System determines hash value for file chunk concatenated with salt value.
5. System encrypts the file chunk. The key is an output from key generator described in section Key scheduler.
6. System divides file chunk according to chosen RAID level and uploads its parts to the external cloud storage services.
7. Points 2-6 are repeated until all chunks have been uploaded.
8. System calculates main hash value based on concatenated chunks hashes.
9. System signs the hash with its own private key, stores signature in a local database and sends signed and unsigned values to the user.
10. User stores server-signed value. Further, he signs unsigned value received from server with his private key and then he sends it back to the server.
11. System stores user's hash and generates log. The log allows downloading the file in future. The log is stored in the local database or is sent to the user (in *off-line* mode).
12. System ends the upload process.

In upload phase we can highlight steps that provide:

- integrity — steps: 4, 8;
- confidentiality — steps: 5;
- availability — steps: 6.

Download phase — communication between user and proxy node

1. User logs into an account.
2. User chooses the file, which he wants to download or uploads *off-line* mode configuration file.
3. User and server authenticate themselves [3] and an e-mail message with a temporary confirmation code is sent (optional).
4. User enters the code and waits for the end of the process.
5. User receives report containing information about merging process.
6. User follows the instructions that appear on the screen and he finalizes the whole process.

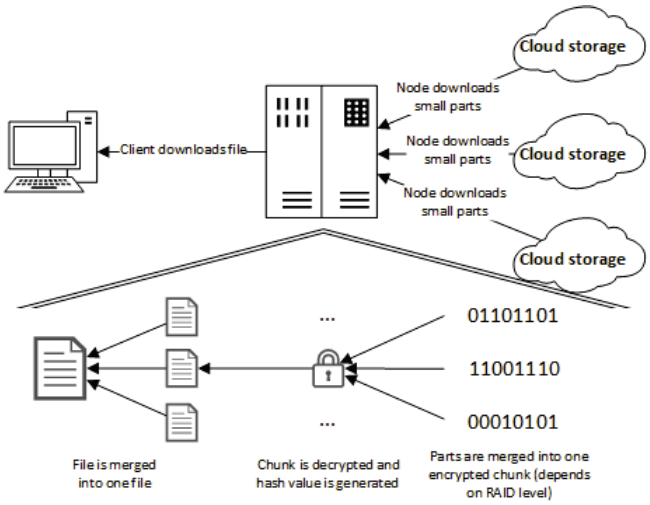

Fig. 5. Simplified download phase.

Download phase — communication between proxy node and cloud storage systems

1. User takes all required actions, described above.
2. System determines number of file chunks.
3. System determines number of chunk's parts.
4. System downloads all parts of one chunk and checks chunk's checksum (if chosen RAID level supports it).
5. System decrypts chunk and determines hash value (salt value is included).
   - (*Off-line mode only*) System compares determined hash value with hash value stored in configuration file. If values are different the system downloads the chunk once again or reports service error.
6. Points 2-5 are repeated until all have been downloaded.
7. System removes padding from last file's chunk.
8. System merges all chunks and generates final file.
9. System calculates main hash value based on concate-

[5]The default chunk size is 1024 kB.

[6]First byte of padding is 11111110, further bytes are pseudo-randomly generated (the same value as first byte is forbidden).

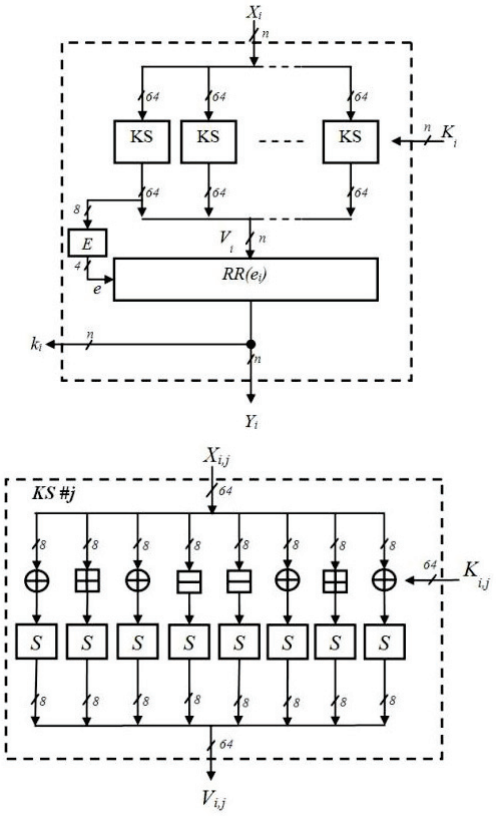

Fig. 6. PP-1 key scheduler.

nated chunks hashes.

10. System compares values (stored and generated hashes). If they are the same, system sends generated hash to the user — if obtained hash is correct, the main download process is finished.
11. System generates final report, which contains information about downloaded file (size, number of chunks, checksums etc.).
12. System ends the download process.

Key scheduler

The system uses key generator, which is required to ensure strong cryptographic protection. Each data chunk (before splitting) is encrypted by AES cipher. We used *PP-1* [6] key scheduler, which produces good quality, pseudo-random round keys [2].

As we can see in the figure 6, there are 2 inputs in the left-side schema — X_i and K_i. Iterator i indicates, which key is produced. The X_i value is equal to:

- for first iteration: n-bit constant $B = B_1||B_2||...||B_t$, where $B_1 = 0x91B4769B2496E7C$ and $B_j = Prm(B_{j-1})$ (Prm is a permutation described in [5]);
- for further iterations: Y_{i-1}.

The K_i value is equal to:

- for first and second iterations: $K_0 = k$ and $K_1 = 0n$ for master key length equals to n or $K_0 = k_0$ and $K_1 = k_1$ for master key length equals to $2n$;
- for third iteration: $RL(B \oplus (A \wedge (K_0 \oplus K_1)))$, where RL is left rotation by 1-bit and $\wedge$ is a Boolean AND operation. Value A depends on master key length (for n: $A = 0n$; for $2n$: $A = 1n$);
- for further iterations: $K_i = RL(K_i - 1)$.

The key scheduler generates keys based on user's key. We keep in mind that users often use too short or dictionary passwords (in general: too weak and easy to crack) [9]. Our system requires user's key, which contains small and big letters, digits and special characters. It's length cannot be shorter than 8 characters. Also from some security reasons, we skip first 3 generated keys.

Off-line mode

Off-line mode was introduced in order to increases system fault-tolerance. In one of possible scenarios, user who uses only one system can lose access to his account (or in general to the system). In this situation it is not possible to restore the data, because we do not have enough information about the whole process. If the *off-line* mode was chosen, the user is in possession of required data. The data allows downloading the file from any the instance of the discussed system. User just needs to register an account and link cloud storage provider services with it. What is more, if there are any problems, user can simply run an instance of the system on his machine and download a file. The structure of the configuration file is shown in listing 1.

Listing 1. Configuration file structure.

```
File version
Protection key salt
RAID level
Flag of availability
List of the cloud storage services

Information about single file
  (I) when logical disks are used only
  chunks localisation in the cloud storages
  chunks hashes

  (II) otherwise
  chunks localisation in the cloud storages with
      information about role (parity disk,
      logical disk)
  chunks hashes

File checksum
```

The mode can also be used to share resources without granting permissions to an account if the access to the cloud resources are publicly available. The user can send the configuration file to other users and give them protection key. The *flag of availability* informs the server if credentials are required. If files are not accessible, the system interrupts downloading process. Changing this value does not affect system's security.

Node redundancy

The system increases the data availability, but single proxy node can be a bottleneck. It can be easily attacked. We have investigated some possibilities for increasing redundancy and chosen two of them - replication and PaaS model (*Platform as a Service*). In the replication case (figure 7), the system consists of several proxy nodes, where each node has it's own, local database. The database systems communicate with each other and update the local information. We can use *near real-time replication* mode. Its performance is quiet good. Nevertheless, it generates a lot of network traffic. This solution has at least two disadvantages:

- it requires some kind of a load-balancer, which will choose less stressed nodes,
- it requires a dedicated database software such as Microsoft SQL Server or Oracle Database, which are expensive and consume a lot of system resources.

The alternative to the above can be cloud computing (figure 8). The problems with the load balancing and distribution

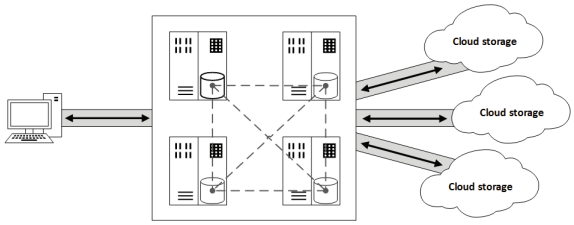

Fig. 7. Replication based model.

do not apply to cloud systems, because of their nature. The system requires only a platform, where it can be run. This solution compared to replication is much easier to configure. However, it requires an external service, where costs can be dependent on traffic. The customer can access the file as long as subscription is active. The user has no direct database control. Database loss means loss of all data.

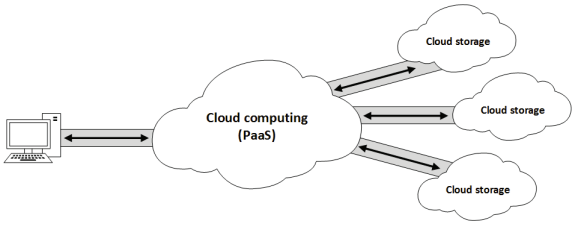

Fig. 8. PaaS based model.

In our opinion, the PaaS platform is better choice (performance/price), but it is important to remember about regular database backup. Both solutions can be used by small-/medium companies, with simple network infrastructure used mainly for non-confidential data. The private cloud is still too expensive and often it is provided as a hybrid cloud (public and private clouds share the same hardware resources).

Simple security analysis

In this section, we briefly describe security of the system. Below, we list the types of stored and secured data.

Stored data

We can highlight two types of stored data — general data and strictly cryptographic data. The first type contains information about user's accounts such as credentials, password's salt, active cloud services, uploaded files and profile configuration. The second type are the asymmetric keys for digital signature purposes. Each user individually negotiates two pairs of keys with the node. Additionally, some data is stored in protected memory area.

Secured data

The system protects important data with one or more keys (except hash functions). It uses:

1. a global symmetric key and a private asymmetric key, stored in a protected memory area, for user's sensitive data protection and authentication purposes based on public key infrastructure (PKI);
2. a private asymmetric key (node-to-user pair), stored in a local database, for digital signature purposes. The key is encrypted by the global symmetric key;
3. a hash function with pseudo-random salt, for transforming user's password into unique hash value.

What is more, all confidential data associated with cloud's accounts (e.g. credentials) are encrypted twice: by global symmetric key and by user password before hashing. Every password's change forces re-encryption of protected data with a new key.

System limitations

Presented system has its limitations. In this section we describe possible problems that can occur.

Lost password

Significant number of systems allow a user to reset password. It can be done in two ways - the system generates new, temporary password and sends it to user's e-mail address or generates a special time-limited web address. Of course there exists others solution based on cell phones, tokens etc. As shown in section Simple security analysis, the system stores encrypted cloud storage services data. If user changes the password, the data is encrypted again. The problem occurs when the user has lost the password. Because of the security level, it is not possible to obtain raw credentials and user is obligated to re-enter all protected information related to cloud services. The solution could be rebuilding encryption system and preparing recovery module. The similar model of data protection is used by Microsoft BitLocker. We did some research and found the document [13], which describes an effective attack on the Windows encryption system[7]. What is more, the data can be recovered using Microsoft account and assigned recovery key(s). It means, that any person who can access user's account, is able to unlock the device.

We decided to stay with a better node's data protection instead of implementing any data recovery solution, which can be used to take over the control of the user's accounts.

File protection key

The second problem is storing file protection keys. The system does not support this feature. Each user has to store them by himself. It is important, because files keys are not associated with user's password and loss of the account's password does not effect the data. Here comes the question: *How should we store the protection keys?*. We suggest to use any publicly available password manager (e.g. KeePass), which allows managing most kinds of text data and offers synchronization mechanisms between our devices.

Future work

Recently we have been working on the system implementation. We have chosen several free cloud services and in the future documents we are going to present the performance results. What is more important, we are going to conduct several security tests, which will help us to determine the real value of proposed model.

Conclusion

This paper presents conceptual model of a system, which helps to increase security level related to confidentiality, integrity and availability using an additional node. Proposed solution is compatible with cloud storage services and could be used by both companies and private users. Additional node can be built as a web-service application so that the client needs just a modern web browser to use it. That is important for users, who are not familiar with new technologies in particular cryptography, but they care about data security.

The document describes several features, which improve the system fault-tolerance level. The redundancy can increase the system availability and can be introduced in a classical way (replication) or based on cloud computing model. The *off-line* mode helps users reach high level of

[7]Microsoft fixes BitLocker in November 2015.

data availability even when the main system is unavailable. The paper also presents possible system problems and ways in which they can be solved.

Acknowledgement. This work was supported by the 04/45/DSPB/0163 PUT grant.

***Authors**: Ph.D. Anna Grocholewska-Czuryło, M.Sc. Marek Retinger, Institute of Control, Robotics and Information Engineering, Laboratory of Information Technology Security, Poznan University of Technology, ul. Piotrowo 3a, 60-965 Poznan, Poland, email: anna.grocholewska-czurylo@put.poznan.pl, marek.retinger@put.poznan.pl*

REFERENCES

[1] Abu-Libdeh H., Princehouse L., Weatherspoon H.: RACS: a case for cloud storage diversity, Proceedings of the 1st ACM symposium on Cloud computing, June 10-11, 2010, Indianapolis, Indiana, USA.
[2] Apolinarski M.: Statistical properties analysis of key schedule modification in block cipher PP-1, part III in book Soft Computing in Computer and Information Science Volume 342 of the series Advances in Intelligent Systems and Computing, pp. 257-268, A. Wiliński et al. (eds.). Springer International Publishing Switzerland, 2015.
[3] Bilski T.,Pankowski T., Stoklosa J.: Bezpieczenstwo danych w systemach informatycznych, Wydawnictwo Naukowe PWN, Poznań, 2001.
[4] Borgmann M., Hahn T., Herfert M., Kunz T., Richter M., Viebeg U., Vowe S.: On the Security of Cloud Storage Services, Fraunhofer Institute for Secure, Germany (March 2012).
[5] Bucholc K., Chmiel K., Grocholewska-Czuryło A., Idzikowska E., Janicka-Lipska I., Stokłosa J.: Scalable PP-1 block cipher, International Journal of Applied Mathematics and Computer Science, vol. 20, No. 2, 2010, 401-411.
[6] Bucholc K., Chmiel K., Grocholewska-Czuryło A., Stokłosa J.: PP-1 block cipher, Polish Journal of Environmental Studies, vol. 16, No. 5B, 2007, 315-320.
[7] Chu C.K., Zhu W.T., Han J., Liu J.K., Xu J., Zhou J.: Security Concerns in Popular Cloud Storage Services, IEEE Pervasive Computing, vol. 12, issue 4, Oct.-Dec. 2013, p. 50-57.
[8] Coyne L.: IBM Private, Public, and Hybrid Cloud Storage Solutions, International Technical Support Organization, January 2017.
[9] Dell'Amico M., Michiardi P., Roudier Y.: Password strength: An empirical analysis, Proceedings IEEE INFOCOM, IEEE, pp. 1-9, 2010.
[10] Drago I., Mellia M., Munafo M.M., Sperotto A., Sadre R., Pras A.: Inside dropbox: understanding personal cloud storage services, Proceedings of the 2012 ACM conference on Internet measurement conference, November 14-16, 2012, Boston, Massachusetts, USA.
[11] Frankel S., Kent K., Lewkowski R., Orebaugh A.D., Ritchey R.W., Sharma S.R.: Guide to IPsec VPNs, NIST Special Publication 800-77, December 2005.
[12] Grance T., Mell P.: The NIST Definition of Cloud Computing, NIST Special Publication 800-145, September 2011.
[13] Haken I.: Bypassing Local Windows Authentication to Defeat Full Disk Encryption, Black Hat Europe 2015, November 2015.
[14] Homin L.K., Tal M., Erich N.: Cryptographic Strength of SSL/TLS Servers: Current and Recent Practices, IMC '07 Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, 2007, 83-92.
[15] Kunz T., Wolf R.: OmniCloud – The Secure and Flexible Use of Cloud Storage Services, Fraunhofer Institute for Secure, Germany (2014).
[16] Li Z., Dai Y., Chen G., Liu Y.: Towards Network-level Efficiency for Cloud Storage Services, Content Distribution for Mobile Internet: A Cloud-based Approach, part V, p. 167-196, 2016.
[17] Meinel C., Schnjakin M.: Implementation of Cloud-RAID: A Secure and Reliable Storage above the Clouds, Park J.J..H., Arabnia H.R., Kim C., Shi W., Gil JM. (eds) Grid and Pervasive Computing. GPC 2013. Lecture Notes in Computer Science, vol 7861. Springer, Berlin, Heidelberg.
[18] Patterson D.A., Gibson G., Katz R.H.: A case for redundant arrays of inexpensive disks (RAID), Proceedings of the 1988 ACM SIGMOD international conference on Management of data, p.109-116, June 01-03, 1988.
[19] Vacca J.R.: Cloud Computing Security: Foundations and Challenges, CRC Press, 2016.
[20] Zhou J.: On the security of cloud data storage and sharing, SCC '14 Proceedings of the 2nd international workshop on Security in cloud computing, p. 1-2, 2014.