DOI: 10.5604/01.3001.0012.8013

FEATURES OF THE MANAGEMENT OF DATA ENCRYPTION KEYS IN THE CLOUD STORAGE MS SQL AZURE

Olexander Beley
Lviv Polytechnic National University, Institute of Computer Science and Information Technologies / Department of Computer-Aided Design

Abstract. *The main principles of data security and access organization in the Microsoft Azure cloud storage are considered. A role of hierarchy and access keys are presented. We describe the setup and the use of their keys (BYOK) for transparent data encryption (TDE) using Azure Key Vault keyring.*

Keywords: communication equipment, data communication, cyberspace, data transfer

CECHY ZARZĄDZANIA KLUCZAMI SZYFROWANIA DANYCH PRZECHOWYWANYCH W CHMURZE MS SQL AZURE

Streszczenie. *Uwzględniono główne zasady bezpieczeństwa danych i organizacji dostępu w chmurze Microsoft Azure. Przedstawiono zagadnienia hierarchii ról i kluczy dostępu. Zostały opisane dostosowywanie i używanie własnych kluczy (BYOK) do przezroczystego szyfrowania danych (TDE) przy użyciu magazynu kluczy platformy Azure.*

Słowa kluczowe: sprzęt komunikacyjny, komunikacja danych, cyberprzestrzeń, transfer danych

Introduction

With the spread of the Internet, the technology of data processing has undergone significant changes. Previously, a computer without any software was just a pile of scrap metal. With the appearance of cloud technologies, even a simple mobile phone with the access to the Internet can help solve complex tasks. Cloud technology is a browser-based mailbox interface with the ability to create and edit online office documents, the solution to complex mathematical calculations, for which the power of one personal computer is not enough.

Currently the cloud computing technology is one big concept that includes many different concepts: software, infrastructure, platform, data, and workplace. The most important feature of cloud technologies is to meet the needs of users who need remote data processing.

Of course, for an average computer user, cloud-based technology is not something you can do nothing without. However, cloud computing is essential for business. Its main advantage is the ability to save on expensive software. After all, you do not have to install expensive office packages and specialized data processing software on each computer. In addition, cloud computing can allow all employees of the enterprise to use, in general, only one operating system, with the access to their workplaces through much cheaper terminals.

However, the concept of cloud technology is a subject to considerable criticism. The main drawback is the security, because not everyone considers it to be reliable to store personal data on a remote server. However, cloud computing has significant prospects, since Microsoft, Apple and Google almost simultaneously began to implement cloud-based technologies in their designs and are not going to abandon them before too long.

Flexibility, scalability and cost-effectiveness contribute to cloud-computing enterprises. This is the answer to the constantly changing economic, financial and technical conditions in which modern enterprises have to work. Constant changes require new ways of thinking, working and doing things. In this new reality, the development of a hybrid cloud model is based. Successful businesses, from small businesses to multinational corporations, recognize the importance of an information system that provides secure data access and effective administration. Cloud-based systems need to be quickly rebuilt to provide cost-effective efficiency with a positive return on investment. This combination of requirements is best served by the various IT services offered in the hybrid cloud.

The cloud storage solution for modern applications that provides stability, availability and scalability to meet customer needs is the MS Azure repository. The Azure repository provides many different security features, such as: the storage account can be protected by RBAC and Azure Active Directory; the data transmitted between the application and Azure can be protected by encrypting the client, HTTPS or SMB; automatic data encryption can be configured when writing to the Azure repository using the "Encrypting the repository" function; for OS drives and data disks used in virtual machines, the Azure disk encryption can be configured; delegated access to data objects in the Azure repository service can be provided with signed URLs; also you can use the analytics to track the authentication method used when accessing the repository.

The security features of Microsoft Azure repository include storage keys, data encryption during data transfer, inaccessible data encryption, and repository analytics. We can protect your storage account with role-based access control. Restricting access according to security principles (the principle of providing access only in those cases and to the extent that knowledge of such information will be necessary, as well as the principle of minimum rights) is extremely important for organizations that need to apply security policies for access to data. These rights are granted by assigning an appropriate role to RBAC groups and applications for a specific area. We can assign users the rights, for example, the rights to a member of the storage account, with the built-in RBAC roles.

The public signature provides delegated access to resources in your storage profile. This means that the client can get a limited right to work with objects in your storage profile for a certain period of time and with a certain set of permissions. We may grant these limited rights without notifying access keys to your account. To access the repository resources with SAS, client just has to pass SAS to the appropriate constructor or method, which contains all the information needed to access the authenticated storage repository in its query parameters.

The Azure data repository for data protection includes the use of: transport layer encryption for data transfer to or from the Azure storage service; SMB 3.0 encryption for Azure file resources; client-side encryption, which encrypts the data before being transferred to the repository and decrypts it after the data is received from the repository service.

1. Basic principles of security management in the Microsoft Azure Data Warehouse

The main recommendations for protecting and encrypting data in Azure include: Multi-Factor Authentication; role-based access control (RBAC); encryption of Azure virtual machines; use of hardware security models; safe management of workstations; SQL data encryption; data protection during transmission; application of file-level encryption.

Verifying the authenticity of the user is the first step in providing access to and management of data in Microsoft Azure. To do this, the Multi-Factor Authentication (MFA) Azure method is used with additional tools other than the user name and password. This method helps to protect access to data and applications without compromising the user's login process. By incorporating Azure MFA for our users, we add a second level of security for signing in and transactions. A transaction may refer to a document hosted on a file server or in SharePoint Online. Azure MFA also minimizes the risk that third parties will be able to access data using compromised credentials. Enterprises that do not use this additional level of protection are more vulnerable to attacks through stealing credentials, which may lead to compromising data.

The main key store solution for Azure services is the Azure Key Vault, which provides general key management capabilities. Keys are stored by users and services. Azure Key Vault supports the creation of user keys or the import of custom keys using the scripts used by our encryption keys.

When implementing encryption of inactive data, several encryption keys are used. Asymmetric encryption can be used to provide the credentials and authentication necessary for key management and access to them. Symmetric encryption is more efficient for mass encryption and decryption, which ensures much more reliable encryption and better performance. Restricting the use of the encryption key reduces the risk of its damage, as well as the cost of re-encrypting if it is necessary to replace the key.

The AES256 (DEK) symmetric key is used to encrypt a partition or data block, which may include many sections and many data encryption keys. Encrypting each data block with another key creates additional complexity for executing attacks on encrypted data. When creating a new DEK key, re-encrypting this key requires only data in its associated block.

The Asymmetric Encryption Key (KEK) is used to encrypt data keys, the use of which allows you to encrypt the data encryption keys directly and manage them. Since KEK keys are needed to decrypt the DEK keys, it can actually be considered as the single point to control the DEK keys. Data encryption keys encrypted using key encryption keys are stored separately, and only the entity that has access to key encryption keys may receive any encryption key encrypted with the KEK key.

The client's encryption model is the encryption process that we perform with the service program in Azure or an application that works in the user's data processing center. When using this encryption model, we will receive encrypted data in the form of a large binary code without the possibility of decrypting it and also without the access to encryption keys. In this model, the keys are managed by the appropriate application and this process is opaque for the Azure service.

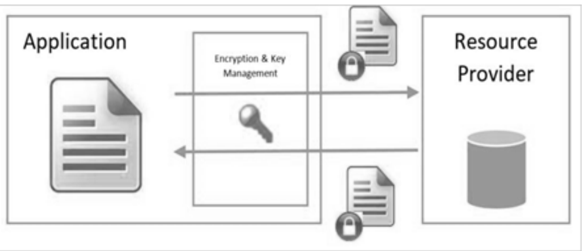


Fig. 1 Model encryption client

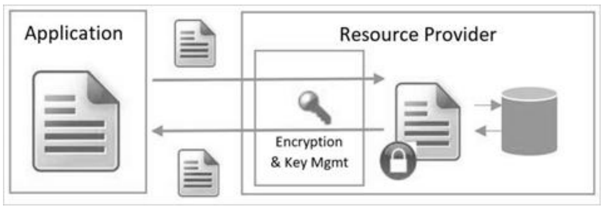


Fig. 2. Models for managing encryption keys on the server side

Server-side encryption models are Azure encryption. In this model, we perform encryption and decryption operations. In the Azure repository, you can retrieve data using plain text operations, and then perform internal encryption and decryption. We may use Microsoft encryption keys or our own.

If you want to encrypt inactive data and manage encryption keys, you can use server-side encryption with the help of user-managed keys in the repository. Some services can store root key encryption keys in Azure Key Vault and store the encryption key of encrypted data in the internal repository along with the data. In this scenario, we can transfer your own keys (BYOK) to the key store or create keys to use them to encrypt the desired resources. In doing so, we perform encryption and decryption operations using the custom key as the root key for all encryption operations.

When Azure disks are encrypted, the Azure key store is used. Thanks to this, we can manage the keys and secrets of disk encryption as a part of the key store subscription. Encryption is performed for all inactive data on virtual drives in the Azure storage service. The Azure key store should be used for key audit and policy use. There are many risks associated with the lack of suitable secret key protection tools used to encrypt data. If intruders have the access to secret keys, they will be able to decrypt data and gain access to confidential information.

To encrypt data stored on disk using a secret key, secure key creation, storage, access control and encryption keys are required. Although some points may differ, the implementation of inactive data encryption can be described using the concepts illustrated in the following scheme (see Fig. 1).

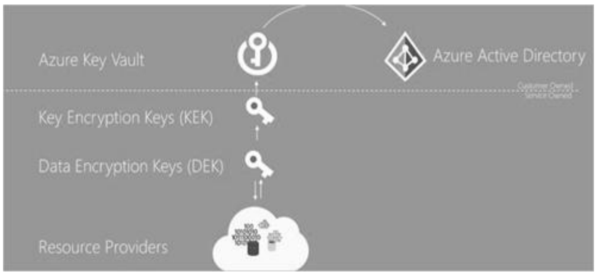


Fig. 3. Azure key storage

Since most attacks are aimed at the user, one of the main targets of the attackers is the end point. If the attacker damages the end point, they will be able to use user credentials to access the organization's data. Most end-to-end attacks occur because users are administrators on their local workstations.

All Azure storage services (BLOB storage, queue storage, spreadsheet and Azure file service) support server-side inaccessible encryption, and some services support customer-managed client-side encryption keys. The Azure BLOB storage and the Azure file service also support the 2048-bit RSA-controlled RSA keys in Azure Key Vault. When using client-side encryption, data is encrypted and transmitted as an encrypted large binary object.

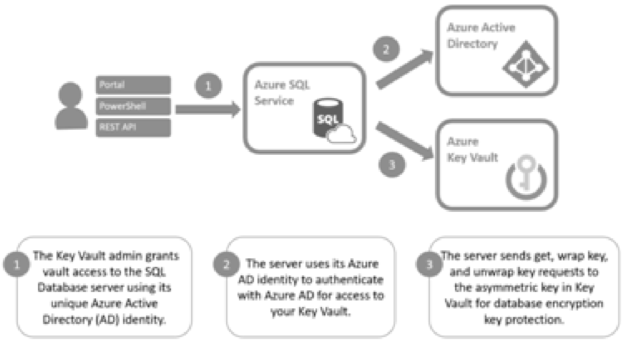
Currently, the SQL Azure database supports inaccessible data encryption in client-side and on Microsoft managed services. Server-side encryption support is currently provided using SQL functions that implement transparent data encryption. Inactive data encryption can be enabled at the database level and at the server level. The SQL Azure database supports the client-managed 2048-bit RSA key in Azure Key Vault.

Azure role-based access control (RBAC) is used to assign users, groups, and access rights within a particular area. A role can be a subscription, a group of resources or a separate resource. We can assign users rights using the built-in RBAC roles in Azure: the role of "Member of the storage account", the role of "Member of the classical storage account", the role of "Participant of the virtual machine".

2. Transparent encryption of data using its own key

Using Your Own Keys (BYOK) for Transparent Data Encryption (TDE) allows you to encrypt the database encryption key (DEK) using an asymmetric key called protector TDE. With TDE keys, we can manage and store data in the Azure Key Vault cloud system for managing external keys. The TDE encryption key stored on the bootstrap database page is encrypted and decrypted with the TDE fuse. The protector TDE is stored in the key store of Azure Key Vault. If the server's access to the key store is cancelled, it will lose the ability to decrypt and read the encrypted database. Protector TDE is configured at the logical server level with the inheritance of all databases that are associated with this server.

TDE with BYOK support allows us to provide: higher transparency and precision with the ability to independently control the TDE fuse; centralized protector's TDE control (with other keys and permissions for all other Azure services) due to their location in Key Vault; division of responsibilities for managing keys and data in an organization; increasing customer confidence, as the Key Vault principle does not allow Microsoft employees to see or receive encryption keys; key change support.



1 The Key Vault admin grants vault access to the SQL Database server using its unique Azure Active Directory (AD) identity.

2 The server uses its Azure AD identity to authenticate with Azure AD for access to your Key Vault.

3 The server sends get, wrap key, and unwrap key requests to the asymmetric key in Key Vault for database encryption key protection.

Fig. 4. The work of TDE with support of BYOK

When we first set up TDE to use a key logger with Key Vault, the server sends Key Vault encryption keys from all databases that support TDE to create a key packet request. Key Vault returns the encryption key for the encrypted database and this key is stored in the user's database. Saved in the Azure Key Vault protector, TDE never leaves the Azure Key Vault. A logical server can only send key operation requests to the TDE protector key material within Key Vault, and never accesses or caches the TDE protector. The Key Vault Administrator has the right to revoke Key Vault permissions of the server at any point, in which case all connections to the server are cut off.

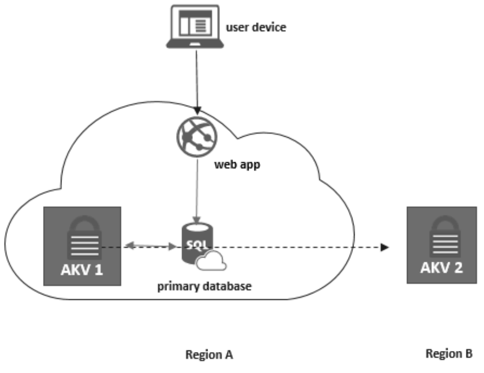


Fig. 5. Setting up a geographic emergency recovery for Azure Key Vault

The Azure Key Vault can be configured in a variety of ways: for a stand-alone database or logical server without geo-replication; for the database or logical server, configured denial handling groups or geo-replicated. At the same time, for each geo-replicated copy, a local Azure Key Vault is required within the refusal group to properly process geographically distributed bounces. In the first case, the high level of accessibility for a database and logical server without geo-replication can be configured by creating two different keystrokes for the server in two different regions, which will store the same key material. To do this, we create a protector TDE in the primary repository located in the same region with the logical server and a clone that is a key to the key store in another Azure region. Now the server will be able to use a secondary repository if primary problems arise during the operation of the database.

For geo-replicated SQL Azure databases, we create the appropriate Azure Key Vault configuration: one source database with a repository and one repository database for the repository in the same region; there must be at least one and no more than four recipient databases; secondary replica databases recipient (threading) are not supported.

For a new deployment of a SQL Server with a geographic disaster recovery, we will need to do the following: create two logical SQL servers in the same regions that previously created the repositories; select the TDE area for the logical server, and then for each SQL logical server select AKV in the same region and the key that can now be used as the protector of TDE. Each server will use a local copy of the protector's TDE. When we perform this operation on the portal, we will receive an AppID for the logical SQL server that allows you to assign a logical permission from the SQL server to access the key repository. After the active geo-replicate action is performed, the recipient database will be created.

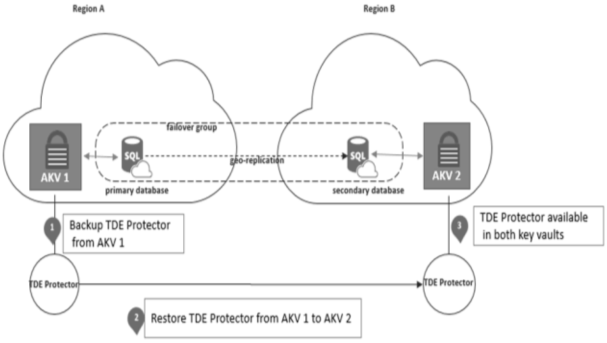


Fig. 6. Deploying SQL-Server with geographical disaster recovery using Azure Key Vault

In a script for SQL Geographic Disaster Recovery Database, you must create and maintain two Azure Key Vault repositories with identical content in the same regions that will be used for georeplication of the SQL database before turning on TDE with the keys stored in Azure Key Vault and the managed client. "Identical Content" here means that both key repositories must contain copies of one protector TDE so that both servers have access to the protector TDE used for all databases. We synchronize both key holders, i.e., we place identical copies of protector TDE in them after changing the keys; maintain the old versions of the keys that were used for log files or backups; preserve the same key properties for all subsequent TDE fuses; maintain the same permissions for accessing SQL servers in the repositories.

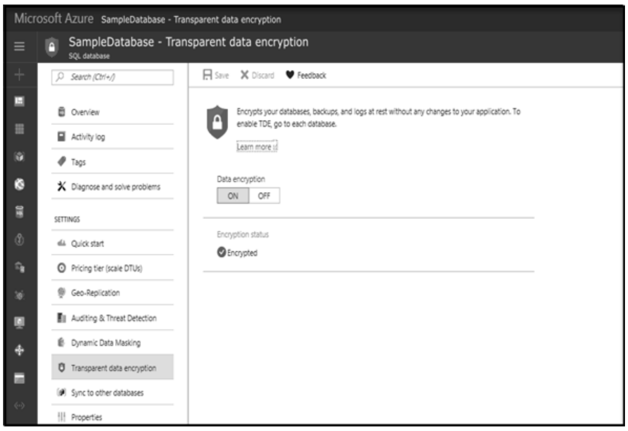
To restore a backup copy encrypted with a protector TDE from Key Vault, you need to make sure that the key is in the original repository with the same name. When the protector TDE changes for the database, the old database backups are not updated to the latest version of the protector TDE. We recommend you to keep all older versions of protector TDE in Key Vault to restore database backups. Saved backups of logs remain encrypted using the original TDE encoder, even if the TDE protector has changed and the database already uses the new TDE fuse. Both the key will be required to restore the database. If the log file uses a TDE fuse stored in the Azure Key Vault, this key will be required during the recovery, even if the database is transferred to the TDE fired service.

Supporting the creation of own keys allows us to independently manage the keys for TDE and establish who and when can access them. The cloud-based external key management system Azure Key Vault has become the first key management service in which transparent data encryption is integrated with the support for the creation of its own keys. Support for creating your own keys allows you to protect the key encryption database asymmetric key stored in Key Vault. This asymmetric key never leaves Key Vault. If the server has permission to access the key repository, the server sends requests for basic key operations in the corresponding Key Vault. The asymmetric key is configured at the server level and inherited by all databases on that server.

We can manage such key management tasks as changing a key and setting permissions for key repositories. You can also delete the keys and enable auditing for all encryption keys. Key Vault provides centralized key management and uses strict tracking with hardware security modules. Key Vault supports division of key management tasks and data to ensure compliance with regulatory requirements.

For operations within Azure, the database does not need to be decrypted. The TDE parameters are transmitted transparently from the source database to the recipient database. This applies to all the following operations: heavens; recovery at a certain point in time through the self-service interface; restore of a remote database; active geo-replication; creation of a copy of the database.

To configure TDE on the Azure portal, we connect on behalf of the owner, member, or SQL Azure security administrator. Transparent data encryption is configurable at the database level. To include TDE in the database, we enter the Azure portal with an administrator account or a member of Azure. The settings for transparent data encryption are displayed in the database user information section. By default, it is managed by a transparent data encryption service. For a server that has a database, the TDE certificate is created automatically.


Microsoft Azure
SampleDatabase - Transparent data encryption
SampleDatabase - Transparent data encryption
SQL database
Search (Ctrl+/)
Save
Discard
Feedback
Overview
Activity log
Tags
Diagnose and solve problems
SETTINGS
Quick start
Pricing tier (scale DTUs)
Geo-Replication
Auditing & Threat Detection
Dynamic Data Masking
Transparent data encryption
Sync to other databases
Properties
Learn more

Encrypts your databases, backups, and logs at rest without any changes to your application. To enable TDE, go to each database.

Data encryption: [✓ ON] OFF

Encryption status: Encrypted

Fig. 7. Set up the protector TDE on the Azure portal

The primary key of transparent data encryption (also called transparent data protection protector) is set at the server level. To use TDE to support our own keys creation and to protect databases using the key stored in Key Vault, we use TDE parameters for our server.

Thanks to BYOK support, users can independently manage their keys and perform operations such as changing keys, setting keys per key retention, removing keys, turning on audits and reporting through all protectors' TDE through Azure Key Vault. Key Vault supports centralized key management, utilizes carefully crafted hardware security modules (HSM), and allows you to share key management and data management responsibilities to ensure compliance with legal requirements.

Conclusion

A complete solution for encrypting inactive data assumes that data is never stored in unencrypted form. During use, when the server loads data into memory, data can be stored locally in a variety of ways, including the Windows paging file, a crash dump, and logging that the application can perform. To ensure that this data is encrypted during storage, IaaS applications can use Azure Disk Encryption on the Windows Azure IaaS virtual machine and virtual disk.

In IaaS applications, we often have to encrypt Azure disks and inactive data encryption settings provided by any Azure service used. In some cases, such as with non-standard encryption requirements or using non-Azure storage, the IaaS application developer may need to implement inactive data encryption on its own. IaaS solution developers can provide better integration with Azure management and meet user expectations by using specific components of Azure. We use the Azure Key Vault service to provide secure key storage and to provide our users with consistent key management options for most of the services of the Azure platform. In addition, our solutions must use the credentials of Azure-managed services to provide service accounts with access to encryption keys.

Protecting user data stored in Azure services is of particular importance to Microsoft. All services hosted in Azure are committed to providing inactive data encryption options. Basic services, such as Azure storage, Azure SQL Database and key analytics services already provide inactive data encryption options. Some of these services support user-managed keys and client-side encryption, as well as service-managed keys and encryption. Microsoft Azure services are constantly improving the availability of inactive data encryption, and new parameters will appear in the preliminary and public versions in the near future.

References

[1] Amies Alex H. S., Qiang Guo Tong, Guo Ning Liu.: Developing and Hosting Applications on the Cloud. IBM Press, 2012.

[2] Armbrust M., et al.: A view of cloud computing. Commun. ACM 53/2010, 50–58.

[3] Armbrust M., et al.: Above the Clouds: A Berkeley View of Cloud Computing, 2009.

[4] Buyya R., et al.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. 25/2009, 599–616.

[5] Casanova H.: Simgrid: a toolkit for the simulation of application scheduling. Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on, 2001, 430–437.

[6] Dean J., Ghemawat S.: MapReduce: Simplified Data Processing on Large Clusters. ACM OSDI, Dec. 2004.

[7] Hamdaqa Mohammad L. T., Ladan T.: A Reference Model for Developing Cloud Applications. Presented at the CLOSER, 2011.

[8] Krauter K., et al.: A taxonomy and survey of grid resource management systems for distributed computing. Software: Practice and Experience 32/2002, 135–164.

[9] Qaisar E. J.: Introduction to cloud computing for developers: Key concepts, the players and their offerings. Information Technology Professional Conference (TCF Pro IT), IEEE TCF, 2012, 1–6.

[10] Quiroz A., et al.: Towards autonomic workload provisioning for enterprise Grids and clouds. 10th IEEE/ACM International Conference on Grid Computing, 2009, 50–57.

[11] Rajkumar Buyya R. N. C.: Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities. Presented at the 7th High Performance Computing and Simulation Conference (HPCS), Leipzig, Germany.

[12] Trunfio P., et al.: Peer-to-Peer resource discovery in Grids: Models and systems. Future Gener. Comput. Syst. 23/2007, 864–878.

[13] Voorsluys W., et al: Introduction to Cloud Computing. Cloud Computing. John Wiley & Sons, Inc., 2011.

Ph.D. Beley Olexander Igorovych
e-mail: tiger_oles@i.ua, Oleksandr.I.Belei@lpnu.ua

Since 1998 he has worked as a software engineer and system administrator for various companies. Since 2000 he has been engaged in information technology, design, programming, database development and web pages, modeling and control of complex processes and systems, mining systems and forecasting data
He is the author of more than 100 scientific works, co-author of 6 monographs, 20 teaching and methodological works, 3 manuals. He participated in more than 50 international scientific-practical conferences.

ORCID ID: 0000-0003-4150-7425

otrzymano/received: 1.10.2018 *przyjęto do druku/accepted: 15.12.2018*