# Electronic safe for passwords storage

**Kamil ŚMIESZEK**[1], **Janusz FURTAK**[2]

[1]Institut of Teleinformation and Authomatics,
Military University of Technology
Warsaw, Poland
kamilsmieszek@gmail.com

[2] Institut of Teleinformation and Authomatics,
Military University of Technology
Warsaw, Poland
jfurtak@wat.edu.pl

ABSTRACT: This paper considers the problem of storing user's sensitive data in a safe manner. Those data include for example: user account names and passwords for any websites, mailboxes, bank account numbers, PIN and PUK codes. This had been solved by creating an application, which allows the user to create his own safe for passwords. The sensitive data are protected by the computer's TPM security features, including an asymmetric encryption algorithms, Root of Trust mechanisms and binding the encrypted data to a specific platform.

KEYWORDS: sensitive data protection, Trusted Platform Module, asymmetric encryption.

## 1. Introduction

Almost in every piece of life the human being has to deal with many services, where the authentication is necessary. This is required for example to access a e-mail server, bank accounts, cloud computing accounts, social websites and so on. the access without giving a login and password is completely impossible. It follows therefore that the human's brain has to remember many authentication data. It is not difficult when the amount of data is small. The serious problem arises when amount of the data is very high. It would be much more convenient for us to use a software tool which supports secure storing and making available the account data.

This paper considers a solution of the problem using an application that allows to create by an user the Electronic Safe for Passwords Storage (ESPS). The safe (containing data of one user) could to be stored on removable storage media and secured before an unauthorized access. It is easy to use, portable, quite cheap and does not create issues in cooperation with the popular computer operating systems. Those requirements are met by removable storage on Flash RAM. Beacuse continously deleting and modifying data stored on this kind of memory does not delete the "old" data, but marks the storage area as empty, it possible to read those data using the external software tools. For this reason the data stored in ESPS on a removable Flash RAM drive are always encrypted. In case of storing passwords or the other secrets there, the management application uses strong (dependable) authorization methods which allows users to receive the sensitive data from ESPS. For this purpose uses a Trusted Platform Module.

The application consists of two components. The first one is a database, in which the sensitive data are stored in an encrypted form (an AES symmetric encryption algorithm is used). Keys and the other necessary data (for example, database user's authentication data) are stored in Root of Trust supported by TPM (it is the second component). The default place for storing of the database and the Root of Trust is the hard drive, but user of ESPS can also change this place pointing a Flash RAM memory when using of ESPS is needed on the other machine.

The market of password-managing software is rather rich. An important issue is the level of security of such the sensitive data. The obvious thing is passwords and any other sensitive data are stored in databases encrypted and unable to read by an unauthorized user. Different security methods are used. The methods do not allow too easy to guess passwords in case of intercept the database - generally functions implementing encrypting and hashing are used. Examples of implementations of a passwords' safes are described below. These include:

- KeePass,
- LastPass,
- GNOME Keyring,
- Windows Credential Manager,
- OneKey Pro.

**KeePass**

One of the most popular free, open-source developed software for managing user passwords is KeePass. This application gained its popularity not only by free distribution, but also by a multiplatfom design (KeePass is available for: Microsoft Windows, Apple Mac OS X, Linux, Google Android, Apple iOS, Microsoft Windows Phone 7, RIM BlackBerry OS and J2ME), multilingual GUI, possibility of extending features by plugins and, what is most important, a constant and active development. The way that KeePass stores the entire sensitive data in a one tiny little database file. The file can be protected by:

- a Master password;
- a file with key (generated during creation of the database);
- a Windows local account authentication.

All of those things can obviously be applied together, but user can choose between options and decide for the one, that provides a convenient and strong passwords protection.

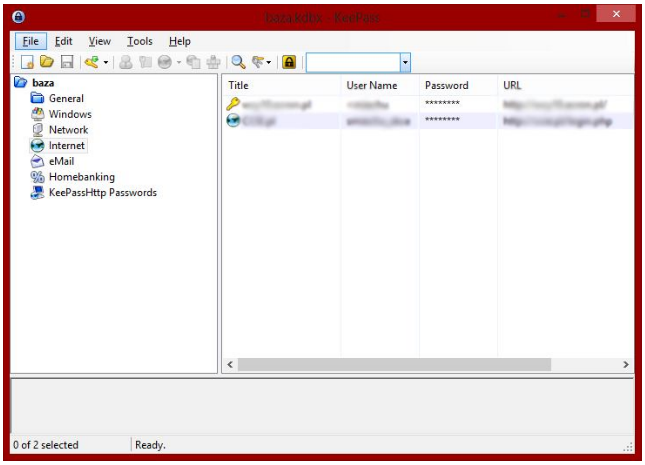


**Fig. 1 Screenshot from the KeePass 2**

A database file is encrypted by one of the two algorithms (i.e.AES or Twofish). Passwords stored in are also hashed by an SHA-256 hash function. All the encryption mechanisms protect the sensitive data from the dictionary and brute-force attacks, but only in a some way. User password is hashed by SHA-256 and then the result is N-times encrypted by an AES algorithm. Finally, there is generated for the cryptogram a SHA-256 hash again. All of that cannot be considered as 100% certain security solution, but make a very serious obstacle for a potential attacker. It should be taken into consideration that not only passwords are stored in a database but also the other user's information, such as account name or user personal notes and protected too. Even during the normal work all of the data are loaded to the computer's RAM memory encrypted.

## LastPass

The similar but a bit different functioning tool is LastPass. This password manager almost fully depends on the cloud computing solutions, so users can access their sensitive data using only an Internet browser with the appropriate plugins or mobile applications. Very basic advantage of that solution is a synchronization passwords between much variety of devices, that is very easy. The security mechanisms of LastPass are:

- a master password;
- a key file;
- a two-step verification with Google Authenticator or (hardware managed) with a Yubikey.

The idea of storing passwords is very different from a KeePass, but the encryption is rather similar. There are also used AES and SHA-256 with salt algorithms.

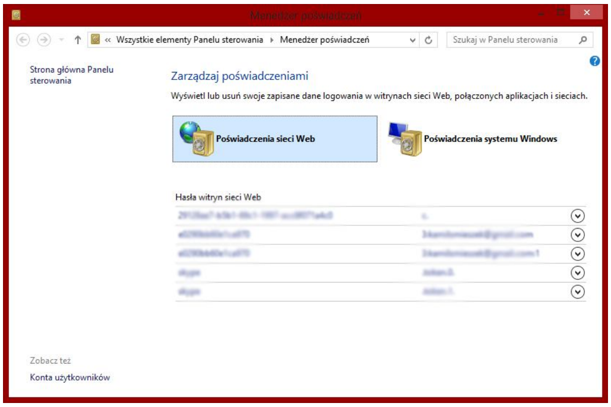
## GNOME Keyring

GNOME Keyring is a collection of components in a Unix/Linux GNOME environment that is able to store passwords, keys or certificates and make them available to user's applications. GNOME Keyring is in fact an OS user session deamon which combines the stored classified data with a logged-in user session. Every password is encrypted by the encrypting-hashing pair – AES-SHA256. The master password for the secured passwords storage is the same as the one that user requires to log in. GNOME Keyring is a part of the Linux PKCS#11 infrastructure, what makes possible to store storage keys on the smart cards.

## Windows Credential Manager

Like a GNOME Keyring in Linux OS, there is also a password manager in Microsoft Windows OS family (available in Windows 2000 and versions above). Windows Credential Manager allows for managing websites access and network access data. The security is close to the Unix/Linux solutions – the main password is the same password as the user logs in the OS.

Credential Manager uses DPAPI (Data Protection API) to encrypt data. The key for them (MasterKey) is encrypted based on PKCS#5 standard, which means generating key directly from the master password. This combined with the 3DES symmetric algorithm decides about the safety of the key stored in a user profile default folder.


Menedżer poświadczeń
Wszystkie elementy Panelu sterowania
Menedżer poświadczeń
Szukaj w Panelu sterowania
Strona główna Panelu sterowania
Zarządzaj poświadczeniami
Poświadczenia sieci Web
Poświadczenia systemu Windows
Hasła witryn sieci Web
Zobacz też
Konta użytkowników

Wyświetl lub usuń swoje zapisane dane logowania w witrynach sieci Web, połączonych aplikacjach i sieciach.

**Fig. 2 Screenshot from the Windows 8 Credential Manager**

For all of the application that were described above, one common issue can be notice: they are not benefit from the Trusted Platform Module features. For this reason, using the TPM means of encryption will be applied to the Electronic Safe for Passwords Storage application.

## 2. Concept of solution

### 2.1. The idea of storing sensitive data in a secure way

The place to store user's ID's and passwords is a relational SQLite database. This one that is being created for the electronic safe for passwords storage application purposes is simple (the database structure is shown on Fig. 3) and consists of two tables:

1. **tblCategories** – the table for storing passwords categories, into which an user can separate his sensitive data. Columns included in this table are:
   a) ID (data type: INTEGER) – category ID, primary key of the tblCategories table;
   b) Name (data type: BLOB) – category name (encrypted) (i.e. mobile phone, e-mail address, bank account);
   c) ParentID (data type: BLOB) – category parent ID (encrypted).
2. **tblPasswords** – the table for storing user's sensitive data, i.e. account names and passwords. Columns included in this table are:
   a) ID (data type: INTEGER) – classified data record ID, primary key of the tblPasswords table;
   b) authLogin (data type: BLOB) – user name/bank account number (encrypted);

c) authPasswd (data type: BLOB) – password/PIN code (encrypted);
d) catID (typ danych: BLOB) – category ID (encrypted), to which belongs encrypted password record, a foreign key connected to a tblCategories table.

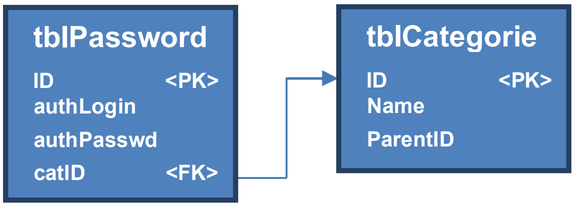


**Fig. 3 Safe for passwords storage – database structure**

Selected database management system does not provide strong security mechanisms for stored data. The unsecured database can be easy broken by simple SQL statements, what allows the potential attacker to have unauthorized access to sensitive data. The ESPS application has to protect those data in much more stronger way.

In the database can be stored data belonging to multiple users. All fields of database records belonging to a certain user (indicated in the above description as "encrypted") are saved in the database in encrypted form using a symmetric algorithm AES-256 and a key of the user (the key of user is stored in Root of Trust supported by TPM).

AES-256 is part of the implementation of an ESPS. Any attempt to access data from the database after reading the record from database needs to decrypt it using a correct key.

## 2.2. Idea of storing the keys

The keys necessary for reading the data from the database are stored in a special structure, which guarantees a safe storage of these keys. The keys necessary for reading the data from the database are stored in a special structure, which guarantees a safe storage of these keys. This structure is hierarchical and is called Root of Trust. Creating and using this structure is supported by the Trusted Platform Module (TPM)[1].

[1] TPM is an implementation of a standard developed by the Trusted Computing Group[6]. This module is designed to support the cryptographic procedures and protocols that can be used for securing data [7]. Trusted Platform Module provides the following functions: generating an asymmetric key pair, secure storage of keys, generating an electronic signatures, encryption and decryption and implementation of an operation defined by the standard PKCS #11.

On top of this hierarchy is asymmetrical Endorsement Key (EK). This key is generated once for each TPM (is not possible re-generation of the key). EK is stored in the TPM resources. The private portion of this key is not available outside of the TPM and is used to verify the signature Storage Root Key (SRK). Asymmetrical SRK is created during the procedure of taking over ownership of the TPM. Its private part is signed by public part of EK. SRK is also stored in the TPM resources. SRK is the parent of each key generated for individual users of ESPS. Sample structure Root of Trust, is shown in Fig. 4.

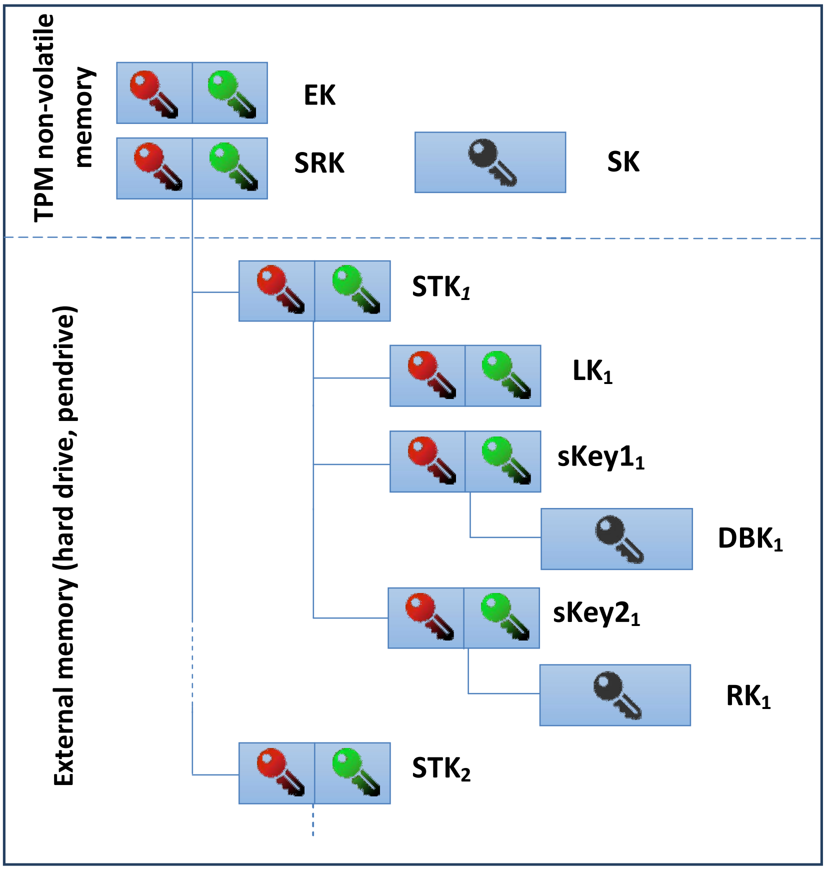


**Fig. 4 Sample structure Root of Trust**

For each user of ESPS is necessary to generate the following set of cryptographic keys:

- **RK** - symmetric key and initialization vector which are used for AES encrypting of records in database belonging to the individual user,

- **DBK** - symmetric key and initialization vector which are used for AES encrypting of the database file that contains only the user's data - the file can be stored for instance in Flash memory,
- **sKey1** and **sKey2** - asymmetric keys for signing keys respectively **DBK** and **RK**,
- **STK** - asymmetric key for signing keys of individual user.

Once the database is closed, but before closing the ESPS application, database file that contains all user data, is encrypted using the AES-256 algorithm and key SK, SK is owned by TPM owner and is stored in a secure NVRAM of TPM.

## 2.3. The concept of solution and application architecture

Application of the ESPS consists two subsystems [8]:

1. Subsystem for TPM key management, including a `.DLL` dynamic library used for TPM operations,
2. Subsystem for user's sensitive data management, including:
   a) application's GUI,
   b) `.DLL` library used for encrypting and decrypting data and database file with an AES algorithm,
   c) `.DLL` library to manage the database.

In order to provide full functionality all of those components must be stored in the same folder together with the Application executable file (on a hard drive or Flash memory).

In the implementation of ESPS are used the following software components:

- graphical interface,
- library functions to encrypt and decrypt data,
- library functions to handle SQLite database,
- library functions to handle the TPM.

Implementation of presentation layer of electronic safe uses the Windows Forms interface with .NET Framework 3.5 [3] This part includes:

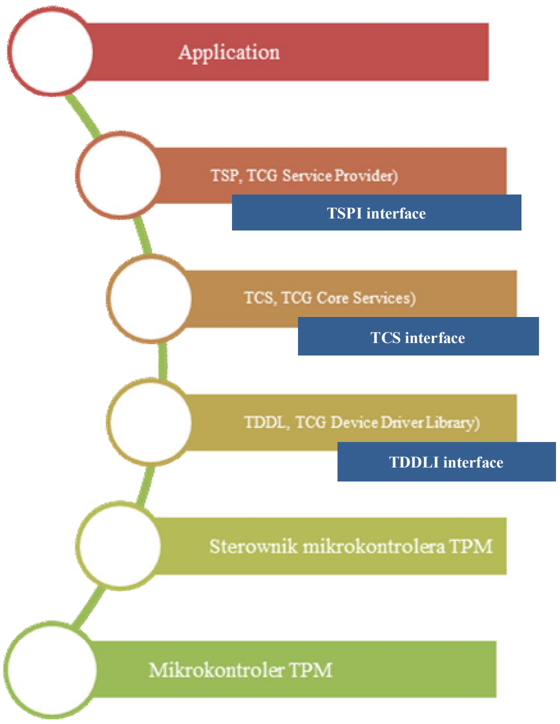
- TPM management (taking ownership, erasing ownership data, checking TPM state, etc.),
- operations on categories of the sensitive data management (adding, showing, modifying, deleting),
- sensitive data management (adding, showing, modifying, deleting),
- generating the TPM key hierarchy,
- erasing the TPM key hierarchy,
- saving the TPM key hierarchy and the database file on a Flash memory drive.

The encryption-decryption `.DLL` library uses an `Aes` from the .NET Framework `System.Security. Cryptography` namespace. This includes:

- encrypting the text or the byte stream with the AES-256 algorithm in CBC encryption mode using the IV vector and a key provided by the TPM,
- decrypting encrypted data using given IV vector and a key provided by the TPM.

The `.DLL` library that is used to manage the SQLite database uses the ADO.NET `System.Data.SQLite` adapter. It realizes the following jobs:

- executing SQL queries,
- adding SQL queries parameters,
- creation of the database,
- SQL database connection management,
- SQL database transaction management.



- **Fig. 5 The TSS structure**

The last `.DLL` library is the one that directs tasks of the TPM module. It was developed using the TSS standards (*Trusted Computing Group Software Stack*) [2]. The TSS library together with TPM module and `System.Security.Cryptography.Aes` class is the fully functional encryption subsystem which is used in the implementation of the electronic safe. The TSS was developed by IBM company and standardized by TCG. It is a specification that makes creating the TPM applications easier. Functions of the TSS are:

- sending and receiving low-level commands,
- sending and receiving data streams,
- the key storage management (those are being stored outside the TPM),
- registering and management of the occurred events.

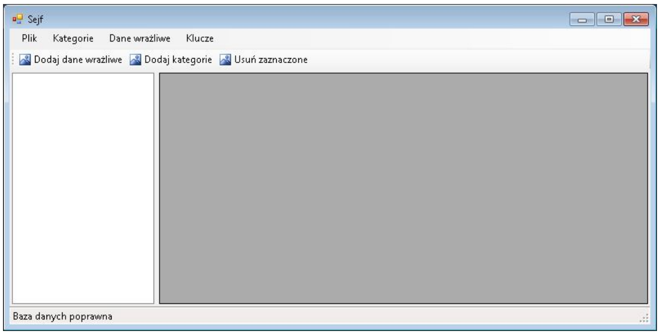
To improve functionality the TSS library was divided into separate layers connected with clearly defined interfaces. To provide a communication between the application and TPM, the data has to go through all of those layers. AT first the application communicates with the TSS service provider (TSPI interface). The TSPI-connected TCS layer manages the TPM resources such as session authorization or conversion of the TCS commands to byte streams understandable for the TPM. The TDDL library then decides about a communication with the TPM driver offering a small piece of instructions including opening and closing access to the TPM driver, sending and receiving data to/from it and requesting the driver about its properties and cancelling commands sent to TPM by the upper layers [4]. Communication with TPM is implemented using the TSS implementation, TrouSerS for Windows (0.3.6 version) [5].

## 2.4. Using of ESPS

The application of ESPS is a software to protect user's sensitive data. The first operations you must make are as follows: take ownership of the TPM computer and set a password for access to the key DBK stored in resources of TPM. The process of taking over ownership of the TPM is done using the same application of the electronic safe to store passwords. The user is asked to enter a password which is required during operations: resetting the TPM and generation of Root of Trust. In the second step the necessary hierarchy of asymmetric keys is generated. This process involves creating a Root of Trust and the database in which user's sensitive data will be stored. During this operation on your hard drive (or flash storage media) in the directory containing the application executable file are saved the following files:

- an empty database with structure ready for storing data, `PasswordSafeDB.db`
- DBK.key for **DBK** key,

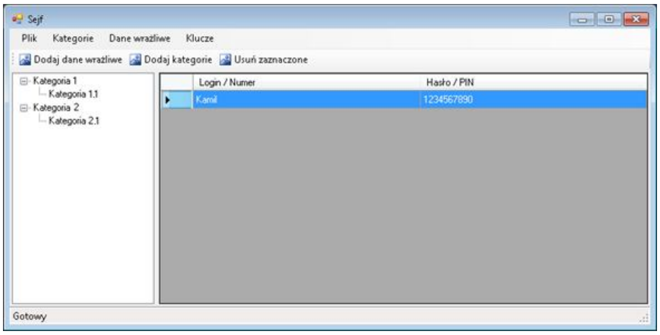
- RK.key for **RK** key.



Baza danych poprawna

**Fig. 6 ESPS after successful creation of all the files**

Without those files written on a hard drive or Flash memory, the user's classified data management is impossible. If files are created successfully, the proper menu fields in the application will be activated, which is shown on Fig. 6:



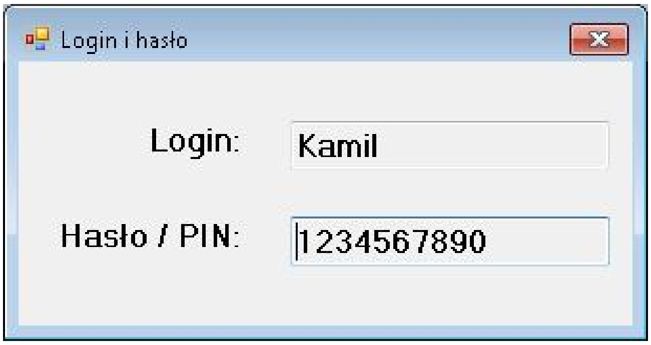
| Login / Numer | Hasło / PIN |
|---|---|
| Kamil | 1234567890 |

Gotowy

**Fig. 7 ESPS after adding two categories and one record of sensitive data**

User can insert his passwords or PIN codes to the previously created database just after adding at least one data category. Those categories can be primary or secondary level. The difference is, that the primary categories hasn't got another parent primary category. The secondary category is connected to one primary category. This structure reminds a file system in OS'es a bit. Entered data will be shown in the main application screen assigned to chosen category (Fig. 7 ). Double-click mouse event causes the opening of the window shown at figure Fig. 8 , which allows to copy and paste data i.e. on a website.

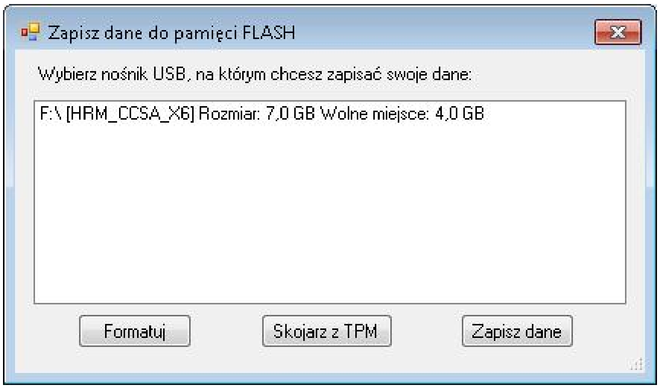
The database with user's sensitive data is stored in the same folder, together with the Application executable file. User can also store his data on external Flash memory drive. These data are just the copies of all the original application files (**PasswordSafeDB.db, DBK.key** and **RK.key**). To improve data security, files are also added to a ZIP archive and protected by a password, which is required to create an archive file. ESPS can read the content of this archive itself – user doesn't need to extract it manually.



Kamil

1234567890

**Fig. 8 A window that shows login and password/PIN number after a double-click on a record**


Zapisz dane do pamięci FLASH
Wybierz nośnik USB, na którym chcesz zapisać swoje dane:
Formatuj
Skojarz z TPM
Zapisz dane

F:\ [HRM_CCSA_X6] Rozmiar: 7,0 GB Wolne miejsce: 4,0 GB

**Fig. 9 A window that allows to create a ZIP archive file**

## 3. Conclusion

Using a Trusted Platform Module to protect user's sensitive data has the effect of significant rising the protection level of those data. The potential attacker cannot get access to them, if he doesn't know the password, even in case of stealing a computer. This is possible, because of one of the TPM features – a special counter for unsuccessful authentications [1]. If the factory-implemented counter value is exceeded, TPM locks itself before access from outside for some time, what is quite significant inconvenience for the attacker. The obvious problem of the latest safe's implementation is lack of possibility to export the whole key structure and import them in another's computer TPM module. Due to those circumstances that had occurred it will be impossible to recover any data secured by the TPM asymmetric encryption algorithms. It is very important to improve an application in that aspect, what will be the first job to do in later implementations.

## References

[1] *TPM Main Part 1 Design Principles. Specification Version 1.2. Revision 116*, Trusted Computing Group, Incorporated, 2011

[2] *TCG Software Stack (TSS) Specification Version 1.2 Part1: Commands and Structures* (http://www.trustedcomputinggroup.org /files/resource_files/6479CD77-1D09-3519-AD89EAD1BC8C97F0 /TSS_1_2_Errata_A-final.pdf).

[3] MAYO J., *C# 3.0 dla .NET 3.5. Księga esperta*, Helion, 2010

[4] *Trusted Platform Module Library. Part 1: Architecture* (http://www.trustedcomputinggroup.org/files/static_page_files/7F7F6AFE-1A4B-B294-D0EE43535A6176B2/TPM%20Rev%202.0%20Part%201%20-%20Architecture%2000.96%20130315.pdf),

[5] CHALLANER D., YODER K., CATHERMAN R.., SAFFORD D., VAN DOORN L., *A Practical Guide to Trusted Computing*, IBM Press, Boston, 2007.

[6] *TPM Main Part 1 Design Principles. Specification Version 1.2. Revision 116*, Trusted Computing Group, Incorporated, 2011

[7] KINNEY S., *"Trusted platform module basics: using TPM in embedded systems"*, Embedded Technology Series,Elsevier Inc., 2006

[8] VIEGA J., MESSIER M.: C I C++. *Bezpieczne programowanie. Receptury*, Helion, Gliwice, 2005.

## Elektroniczny sejf do przechowywania haseł

STRESZCZENIE: Artykuł dotyczy problemu bezpiecznego przechowywania wrażliwych danych użytkownika. Te dane mogą obejmować: nazwy kont i treść haseł do aplikacji internetowych lub do skrytek poczty elektronicznej, numery kont bankowych, numery PIN do kart kredytowych, numery PIN i PUK do telefonów komórkowych. Zaproponowane rozwiązanie pozwala użytkownikowi na utworzenie jego własnego sejfu na te dane. Dane w tym sejfie będą zabezpieczone kryptograficznie. Do wspomagania tych zabezpieczeń jest wykorzystany moduł Trusted Platform Modul, a w szczególności do tworzenia kluczy asymetrycznych, do zbudowania i utrzymywania drzewa zaufania dla aplikacji, szyfrowania/odszyfrowywania wrażliwych danych użytkownika oraz do uwierzytelniania użytkowników sejfu.

SŁOWA KLUCZOWE: : zabezpieczanie wrażliwych danych, Trusted Platform Module, szyfrowanie niesymetryczne.

*Received: 25.11.2014*