Tomasz HYLA, Imed El FRAY, Witold MAĆKÓW, Jerzy PEJAŚ

West Pomeranian University of Technology, Szczecin

doi:10.15199/48.2015.02.23

Implementation of Pairing-Based Cryptographic Trust Infrastructure in Mobile Environment

Abstract. *Current trends in information system design show that users should have access to services provided by information system offered on their mobile devices. Because many information systems store sensitive information, appropriate protection mechanisms must be deployed. This paper presents the software libraries (APIs) that can be used to implement pairing-based systems on mobile devices. Variety of mobile devices causes that is necessary to design a generic trust infrastructure that will allow to implement efficiently a system that uses parings. There are two basic paradigms that can be used: client-server or cloud-based. The analysis of pros and cons of the architectures showed that it is faster and easier to implement pairing application using cloud-based approach mainly because of the lower number of components required to implement, e.g., the library containing pairing calculations must be only prepared for one operating system instead of many that are using different technologies. The tests conducted using cloud-based demonstrator showed, that in case of documents signing and verification with auxiliary server instead of the mobile device, the pairing calculation time is marginally short in relation to the required to retrieve documents from a remote location.*

Streszczenie. *Aktualne trendy w projektowaniu systemów informacyjnych pokazują, że użytkownik powinien mieć dostęp do usług systemów IT za pomocą urządzeń mobilnych. W przypadku przechowywania informacji wrażliwej w systemach informacyjnych muszą być wdrożone odpowiednie mechanizmy zabezpieczeń. W artykule zaprezentowano biblioteki programowe (API), umożliwiające implementacje systemów wykorzystujących odwzorowania dwuliniowe na urządzeniach mobilnych. Różnorodność urządzeń mobilnych powoduje, że konieczne jest zaprojektowanie ogólnej infrastruktury zaufania, w szczególności przy założeniu wykorzystania odwzorowań dwuliniowych. W artykule zostały przeanalizowane dwa podstawowe podejścia bazujące na modelu klient-serwer i modelu bazującym na chmurze. Testy bazujące na demonstratorze wykorzystującym model chmury pokazały, że czas obliczeń odwzorowania przy podpisywaniu i weryfikowaniu podpisu cyfrowego jest bardzo mały w stosunku do czasu pobierania plików ze zdalnych serwerów. (**Implementacja kryptograficznej infrastruktury zaufania opartej na odwzorowaniach dwuliniowych w środowisku mobilnym**).*

Keywords: mobile device, bilinear pairing, trust infrastructure, cloud, digital signature.
Słowa kluczowe: urządzenie mobilne, odwzorowanie dwuliniowe, infrastruktura zaufania, chmura, podpis cyfrowy

1. Introduction

Nowadays people use hundreds of millions of mobile devices like tablets or smartphones. Mobile devices often contain sensitive information, e.g., personal data, address books, files with valuable information (e.g., contracts, orders, projects). The protection of such information is an important feature of mobile device information system and should be considered during the design of the appropriate protection mechanisms.

Many new cryptographic methods are based on pairings. A pairing is defined as a bilinear map between elements of two finite, cyclic and additive groups G_1 and G_2 to a third finite cyclic group G_T defined multiplicatively. Both of G_1 and G_2 are of prime order q, as it is in the case of G_T. In practice, pairing $\hat{e}$ allows to solve certain problem in one group, even if the problem is said to be hard in an-other group. Let $(G_1, +)$ and $(G_T, \cdot)$ be two cyclic groups of some prime order $q>2^k$ for security parameter $k \in N$. The bilinear pairing [1] is given as $\hat{e}: G_1 \times G_1 \to G_T$ and must satisfy the following three properties:

1. **Bilinearity:** $\hat{e}(aP, bQ) = \hat{e}(abP, Q) = \hat{e}(P, abQ) = \hat{e}(P, Q)^{ab}$ for all P, $Q \in G_1$ and all a, $b \in Z_q^*$; this can be restated in the following way: for P, Q, $R \in G_1$, $\hat{e}(P+Q, R) = \hat{e}(P, R)\hat{e}(Q, R)$ and $\hat{e}(P, Q+R) = \hat{e}(P, Q)\hat{e}(P, R)$.
2. **Non-degeneracy**: some P, $Q \in G_1$ exists such that $\hat{e}(P, Q) \neq 1_{G_2}$; in other words, if P and Q are two primitive elements of G_1, then $\hat{e}(P, Q)$ is a generator of G_2.
3. **Computability**: given P, $Q \in G_1$, an efficient algorithm computing $\hat{e}(P, Q)$ exists.

This paper presents the software libraries that can be used to implement pairing-based systems on mobile devices. Variety of mobile devices causes that is necessary to design a generic trust infrastructure that will allow to implement efficiently a system that uses parings and have the features similar to traditional Public Key Infrastructures (PKIs). There are two basic approaches that can be used. One that uses client-server paradigm and another one that uses cloud approach.

The paper is organized as follows. Section 2 contains description of pairing-based cryptographic libraries. Section 3 contains description of mPBC library. Next, Section 4 contains description of possible system architectures that allows to verify and create signature using bilinear pairings. The Section 4 also contains results from implementation of demonstrator version of the IE-RCIBS scheme [2]. The paper ends with conclusions.

2. Pairing-based cryptographic libraries

The first cryptographic library optimized for pairing-based cryptography was PBC (Pairing-Based Cryptography) library developed by Benjamin Lynn in C [3]. However, its performance is not optimal for all types of bilinear mappings. The second and the most efficient cryptographic library based on the bilinear mapping is MIRACL (Multiprecision Integer and Rational Arithmetic Cryptographic Library). MIRACL was developed in Ireland by the research team under the direction of Michael Scott [4]. This library is written in C/C++ and is widely used for studies requiring high performance. Other popular libraries that's support bilinear pairings are RELIC [5] and TEPLA(University of Tsukuba Elliptic Curve and Pairing Library) [6].

The PBC library is a free C library built on the GMP library, which performs the mathematical operations underlying pairing-based cryptosystems. The PBC library is designed to be the backbone of implementations of pairing-based cryptosystems. It provides routines such as elliptic curve generation, elliptic curve arithmetic and pairing computation. Pairings times are reasonable and C language

enables implementation on many operating systems environment. The API is abstract enough that the PBC library can be used even if the programmer possesses only an elementary understanding of pairings [3].

MIRACL Crypto SDK – is a software library written in C/C++ language, which is widely recognized by developers as the best open source standard for ECC (Elliptic Curve Cryptography). MIRACL enables to generate appropriate curve based on bilinear mapping using Cocks-Pinch method [7]. Since 2011 the library is commercial. Historical creator of this library Michael Scott in 2012 founded a start-up Certivox, aimed at promoting the use of a bilinear mapping in everyday life.

RELIC is a modern cryptographic tool that puts great emphasis on efficiency and flexibility [5]. It is used to create tools and cryptographic security at different levels, using a number of arithmetic calculations. RELIC Library is the main part of the project implementation TinyPBC [8], [9]. RELIC operates under the LGPL license, which allows commercial usage.

TEPLA [6] is a library supporting the development of applications or systems using cryptography based on bilinear mapping working on different platforms. TEPLA is also written in C. TEPLA the public library is open source under the BSD license, which runs on Windows platforms, Linux and MAC OS X.

One of the others libraries that's supports pairing-based calculations is Magma package [10]. Magma contains bilinear mapping based on Weil and Tate pairings on elliptic curves over finite fields. Apart from Weil and Tate pairings, Magma contains Eta bilinear mapping over supersingular curves of small characteristics, and bilinear mapping Ate of large characteristics. Thanks to the similarity between representations of Tate, Ate, and optimal Ate pairings, it is possible to perform all calculations using Magma. Magma is a very useful for testing newly developed codes. Another mathematical library, Pari [11] written in C, also offers since 2011, calculations based on the bilinear mapping. There are currently several other cryptographic libraries such as CHARM [12], [13], PANDA [14], which uses MIRACL'a, PBC or RELIC.

The MIRACL library is indicated mostly for tasks that go beyond the academic ranges (purely commercial). MIRACL supports a wide range of platforms, including: Intel, ARM, Texas Instruments, Sun, Oracle, ATMEL, MIPS TECHNOLOGIES, ANALOG DEVICES. The other two libraries; RELIC and TEPLA have attractive features useful to encrypt and decrypt messages. Starting with the basic mathematical operations and ending with complicated calculations. It can be noted, however, that RELIC has much more features, modules, and protocols that increase the range of possibilities for users of this library. Used in the library TEPLA calculations have very short times, second only to a few parameters PBC library, while the complexity of these calculations provides the best security in comparison to others. In turn, according to the tests [15], both in the first case and in the second, the size of the message and the encryption key greatly affect response times and RELIC MIRACL library. In both cases, the library RELIC has better results than MIRACL. According to the above information, it can be confirmed that the MIRACL library is better than the PBC, RELIC or TEPLA because it provide more functionalities, the code is faster and more transparent. MIRACL has a good performance and the use of RAM and CPU is very low.

Miracle library is current global standard in relation to the Elliptic Curve Cryptography over GF (p) and GF (2m). MIRACL also includes over twenty cryptographic protocols based on bilinear mapping. PBC Library is also suitable for solutions related to cryptography based on bilinear mapping and elliptic curves. PBC Library is fast and portable, and offers a lot of functions and cryptographic methods. Both libraries are very good, except that the PBC library is rather recommended for academic assignments or those in which it is not strictly put special emphasis on memory, CPU and execution time. They can be used by people who are very familiar with this type of cryptography and also those who have less knowledge about it but they are interested in this subject. The PBC library was chosen for the purpose of our demonstrator software mainly because of its wide usage by researchers of pairing-based cryptosystems and its open-source licence.

3. mPBC Library

The mPBC (mobile Pairing-Based Cryptography) library is written in C language and it uses PBC library developed by B. Lynn (Fig. 1.). It contains implementation of algorithms developed especially for MobInfoSec project [16], i.e., IE-RCIBS scheme [2], CIBE-GAS scheme [17, 18] and IE-CBE scheme [19] (hereinafter referred to as IE-SK-CBE scheme). The library uses programing convention used in PBC library and can be compiled using most of available C compilers. The mPBC hides the pairing calculation from users and provides high level API mainly with functions form implemented schemes. Hence, mPBC user does not need any knowledge about pairing-based cryptography. It contains also data structure definitions, import and export functions and tests that demonstrate basic functionality. The mPBC purpose is to provide implementation of cryptographic schemes that can be used directly or indirectly on mobile devices.

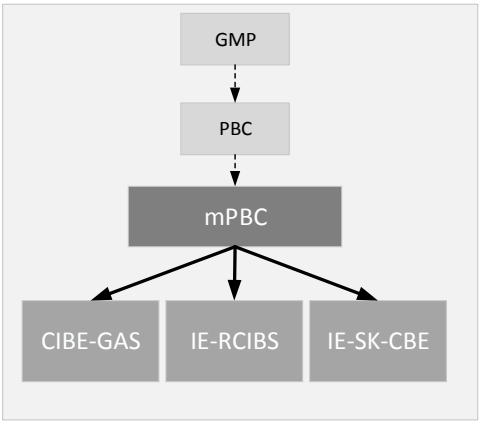

Fig.1. mPBC library components

The first module of mPBC contains IE-RCIBS scheme [2]. Each signing entity that follows in accordance with the scheme, creates a digital signature, which can be verified directly using explicit certificate and indirectly only using an implicit certificate. The scheme allows to sign a document in on-line mode, i.e., a certificate verification is a part of signing process. The scheme consists of 10 functions (*Setup, Create-User, Extract-Partial-Private-Key, Certificate-Generate, Cert-Revoke, Set-Public-Key, Set-Private-Key, Sign, Get-Cert-Status and Verify*). Also data structures and function that enables import and export from binary files were included.

The second module is CIBE-GAS [17]. The CIBS-GAS scheme is a certificate-based group-oriented encryption scheme with an effective secret sharing scheme based on general access structure and bilinear pairings. The CIBE-GAS module contains also modified version of the scheme, i.e., CIBE-GAS-H [18], which is designed to work with arbitrary length messages while original CIBE-GAS scheme

works with limited length messages only. CIBE-GAS scheme consists of eight algorithms (*Setup, SetSecretValue, CertGen, SetPublicKey, ShareDistribution, Encryption, Subdecryption, Decryption*) which were mapped to corresponding functions in the library module. Also, appropriate data structures, e.g., *CIBEGAS_Ciphertext*, *CIBEGAS_SharePublicParams* were created together with auxiliary functions.

The third module of the library contains implementation of IE-SK-CBE scheme. The IE-SK-CBE is an encryption scheme that preserves all advantages of Certificate-Based Public Key Cryptography (CB-PKC), i.e., every user is given, by the Trust Authority (TA), an implicit certificate as a part of a private key and generates his own secret key as well as corresponding public key. In addition, in the IE-CBE scheme the TA has to generate an explicit certificate for a user with some identity and a public key. The implicit and explicit certificates are related with each other in such a way that no one, even the entity of those certificates and their issuer (TA authority) should not be able to recreate an implicit certificate using the explicit certificate. The scheme consist of eight algorithms (*Setup, Create-User, Extract-Partial-Private-Key, Certificate-Generate, Set-Public-Key, Set-Private-Key, Encrypt and Decrypt*) that were mapped to six functions and eight data structures.

4. Trust Infrastructure

The trust infrastructure required to run IE-RCIBS functions provided by mPBC library can be basically created in two ways. First, is typical client-server architecture with mPBC library usage in the mobile device and in the trusted authority server. While the second approach is cloud based, i.e., it requires usage of the mPBC library only on trusted authorities servers. The main drawback of the first approach is the necessity to integrate an mPBC library with a native code on every mobile operating system. This might very time consuming approach, when using C language based library, because not all programing languages used in the mobile operating systems allow easy C-language integration and the library must be optimized for specific devices. Also, more computing power is required from mobile devices.

The main drawback of the second approach is the necessity to create another trusted auxiliary servers for pairing calculation with the same security level as trusted authority server that manages user certificates. This also requires online Internet connection with trusted authority servers during verification in IE-RCIBS scheme. Therefore, it is necessary to create trusted channels between a mobile device and additional servers.

4.1 Client-server architecture

Client-server version of the architecture (Fig.2.) consists of:

- a server managed by Trust Authority (TA), which runs application (*sTA*) providing four web services (*GetPublicParams, GenerateCertificate, RevokeCertificate, GetCetificateStatus*);
- any number of mobile devices with whichever operating system.

The *sTA* server application is implemented for operating system *os1* on server *S* and uses mPBC library in a version for *os1*. It possible to link it with PKI that can authenticate issued public parameters.

The native *sP* application provides crypto functionality related to documents signing and verification. The application is implemented for the specific mobile operating system and uses mPBC library in the version for this system. For example, on the Fig.2, we can see among others mobile device *M1* with operating system *os2*, containing matched version of the library and software: *mPBC.os2* and *sP.os2.* Besides providing of the crypto functionality, *sP* is responsible for communication with *sTA*. Each mobile device contains also three native applications that provide user interfaces:

- *CC* – certificate client, manages data of certificate' key owner;
- *SC* – signing client, enables documents signing;
- *VC* – verification client, enables documents verification;

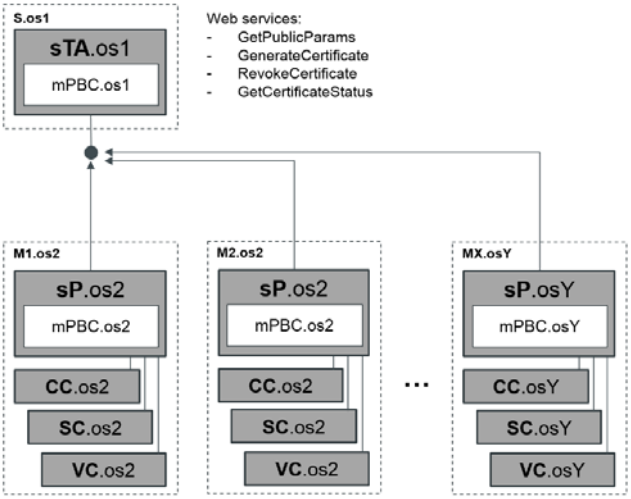

Fig.2. Client-server version of the architecture

Implementation of *sP* applications and applications (interfaces) *CC*, *SC* and *VC* are dedicated for each operating system. The *sP* and client applications *CC*, *SC* and *VC* have access to mobile device local files. Also, they have access to remote files (for example in cloud drives). On a mobile device *sP* does not have to share all of the functionality - installed and used can be for example only VC verification application. The user's private files (keys, parameters) are stored locally by *sP* on his device. Secure communication channel between *sP* and *CC*, *SC* and *VC* exists (communication inside the device).

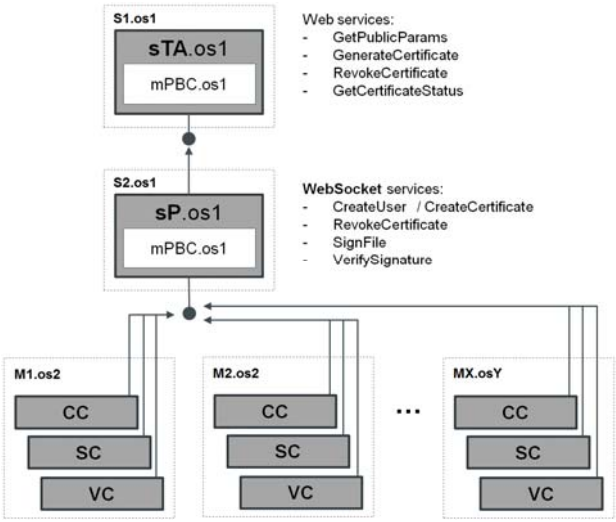

Fig.3. A cloud based architecture

4.2 Cloud-based architecture

In a cloud-based version of the architecture, the *sP* functionality is transferred out of mobile devices by providing an auxiliary logical server. This eliminates the need to port mPBC library for every mobile operating

system, but require to create secure communication channels between mobile devices and auxiliary *sP* server at *TA* site. There is another disadvantage of this solution - auxiliary server should be trusted (the mobile devices' users trust that server will not execute any unwanted actions, i.e., disclosure of private data, incorrect signature calculation or incorrect verification).

The *sTA* certification server is implemented for *os1* (Fig.3.) (deployed on the server *S1*) and uses mPBC library in the version for that operating system. The *sP* application server at *TA* site provides functionality related to a document signing and verification. It is responsible for communication with *sTA* server and operates sequentially according to the IE-RCIBS scheme.

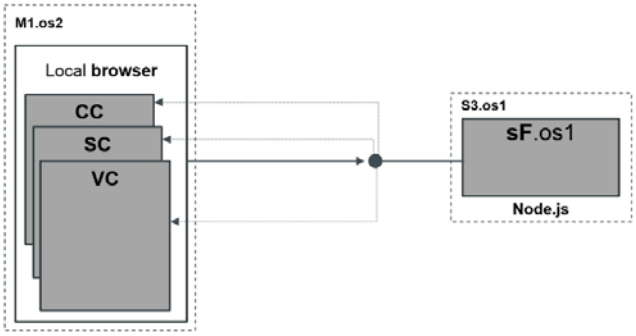

Fig.4. Web applications in a cloud based environment

A client application can be independent from operating system (Fig. 4.) and depends only on web browser. Client applications *CC*, *SC* and *VC* (described in Section 4.1) have only access to remote files (i.e., in the cloud drives). The web applications have no or restricted access to local resources. User's private files (keys, parameters) are stored locally by *sP* in the file system. User does not have direct access to them, uses them indirectly through the services of *sP*. Private file from all users are managed by *sP*.

4.3 Cloud-based demonstration system

We have created cloud based demonstration system to test the system efficiency and locate potential practical problems. The communication channel between mobile devices and TA servers is not secured to simplify system development. In working system probably technologies, like some version of SSL/TSL, would be used. The system consists of:

- the certification server (written in C# language using MS Visual Studio 2013 for Windows 8), which provides services described earlier;
- auxiliary server *sP* (written in C# language using MS Visual Studio 2013 for Windows 8), which provide functionality required to, among others, initiate certificates generation, signature generation and verification, communication w *sTA* server; the server uses WebSocket technology to provide that functionality.
- client applications *cClient*, *sClient* and *vClient* (relating, respectively to *CC*, *SC*, and *VC* application from the Fig. 4), which were written in the form of web applications (HTML5 plus JavaScript) and provided using a simple Node.js file server. They allow certificate management, document signing and signature verification.

Because client applications are run on mobile devices using local Internet browsers, they do not have access to a local file system. The file to be sign or to be verified (the file plus corresponding signature file) is downloaded from DropBox account indicated by the user. *sP* server also uses DropBox account (not related with users accounts) to temporary store generated files and then to share their public address to a specific authorised user.

Dependencies between Dropbox drives used in our demonstrator system presents the Fig.5 in a form of a signing scenario. The sample web application user interface is additionally presented in the Fig.6.

The signing process in the demonstrator is as follows (Fig.5.):

1. User who wants to sign a file using *sClient* web application, retrieve a public link to the file from his private DropBox account (a process mediated by the DropBox IERCIBS application).
2. User sends to *sP* the link to the file.
3. *sP* downloads the file from the link.

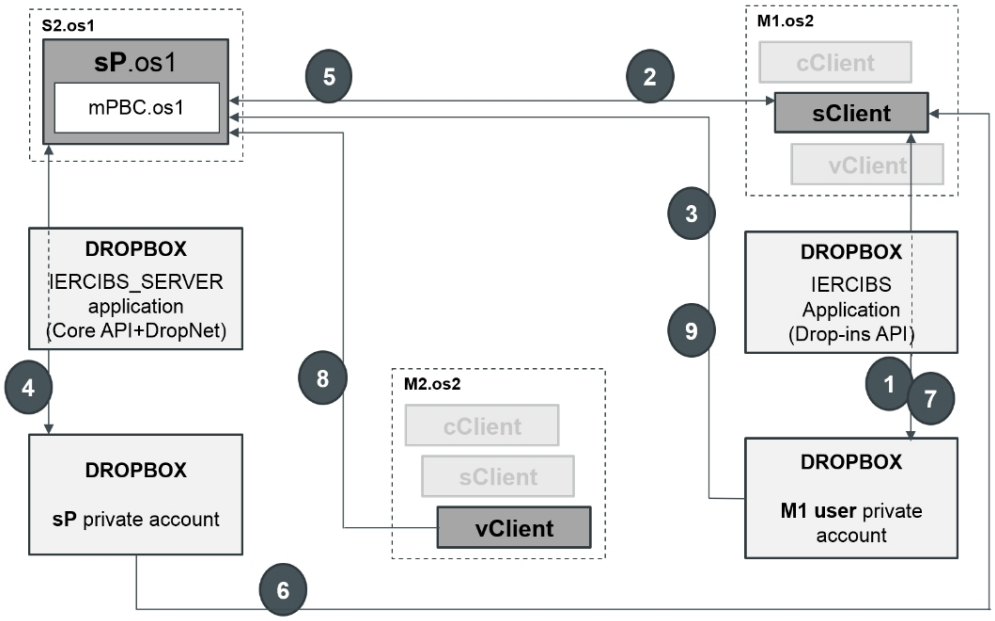

Fig.5. IE-RCIBS cloud-based demonstrator

4. *sP* signs the file using users' private key and saves the signature on its private DropBox account (a process mediated by the DropBox application IERCIBS_SERVER).
5. *sP* sends to *sClient* a public link to the signature file.
6. *sClient* downloads the signature file from the link.
7. *sClient* stores the signature file in user private DropBox account.
8. User verifies a file and a signature using *vClient* application. *vClient* sends the public links (of file and signature) to *sP* (user receives links to the file and signature outside the system).
9. *sP* downloads the file and signature from the links provided and then can verify them.

Fig.6. User interface of an exemplary web application

5. Conclusions

The analysis of pros and cons of the client-server and cloud-based architectures showed, that it is faster and easier to implement pairing application using cloud approach mainly because of the lower number of components required to implement, e.g., the mPBC library must be only prepared for one operating system instead of many that are using different technologies.

The test conducted using cloud-based demonstrator showed, that in cases of signing and verification of documents using auxiliary server instead of mobile device, the pairing calculation time is marginally short in relation to the required to retrieve documents from remote location (in our case from DropBox cloud drive). In case of production level tests it would be advisable to prepare a dedicated file server. Also, it is expected that the implementation of security mechanism that will protect communication channel between web-based applications and auxiliary sP server, will slow down communication. Additional problem that can be identified is creation of a trust model for auxiliary server, e.g., by being able to audit its operations.

Acknowledgment

This scientific *research work is supported by NCBiR of Poland (grant No PBS1/B3/11/2012) in 2012-2015.*

REFERENCES

[1] Galbraith, S. D., Paterson, K. G., Smart, N. P., Pairings for cryptographer, *Discrete Applied Mathematics*, 156 (16) (2008), 3113–3121
[2] Pejaś J., Implicit and Explicit Certificates-based Digital Signature Schemes in Infrastructure with Multiple Trust Authorities (in polish). Wyd. Stowarzyszenie Przyjaciół Wydziału Informatyki w Szczecinie. Series: *Monographs on Informatics,* Tom II, ISBN 978-83-936799-1-1, Szczecin, 2013
[3] Lynn, B., *Pairing-based cryptography library*, http://crypto.stanford.edu/pbc/, (2013), v-0.5.14. C language, LGPL license.
[4] Scott, M., MIRACL library, www.shamus.ie, 2011, V5.5.4.
[5] Aranha, D. F., Gouvêa, C. P. L., *RELIC is an Efficient LIbrary for Cryptography*. http://code.google.com/p/relic-toolkit/, (2013), v-0.3.5. C++ language, LGPL license.
[6] Laboratory of Cryptography and Inform. Security, University of Tsukuba (Japan) *University of Tsukuba elliptic curve and pairing library*, http://www.cipher.risk.tsukuba.ac.jp/tepla/.
[7] Cocks, C., Pinch, R.G.E.: Identity-based cryptosystems based on the Weil pairing (2001) (un-published manuscript)
[8] Oliveira, L. B., et al., TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks, *Computer Communications*, (2010)
[9] Oliveira, L. B., Scott, M., Lopez, J., Dahab, R., TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks, *5th International Conference on Networked Sensing Systems* (INSS'08), IEEE, (2008), Kanazawa/Japan, 173-179
[10] Bosma, W., Cannon, J., Playoust, C., The Magma algebra system. I. The user language. *J. Symbolic Comput.,* 24(3-4):235–265, 1997, Computational algebra and number theory
[11] Belabas, K., Cohen, H., *PARI/GP Library*, http://pari.math.u-bordeaux.fr, 2013, v-2.5.5. C language, GPL license
[12] Akinyele, J.A., et al., Charm: a framework for rapidly prototyping cryptosystems, *Journal of Cryptographic Engineering*, (2013), Volume 3, Issue 2, 111-128
[13] Charm, *A tool for rapid cryptographic prototyping*, http://charm-crypto.com
[14] Chuengsatiansup, C., et al. PandA: Pairings and Arithmetic, LNCS 8365, (2014), 229-250
[15] Peng, Z., Fang, J.J, Comparing and implementation of public key cryptography algorithms on smart card, (2010), DOI: 10.1109/ICCASM.2010.5622377
[16] Hyla, T., Pejaś, J., El Fray, I., Maćków, W., Chocianowicz, W., Szulga, M., Sensitive Information Protection on Mobile Devices Using General Access Structures, *ICONS 2014,* IARIA, pp. 192-196 (2014)
[17] Hyla, T., Pejaś, J.: Certificate-Based Encryption Scheme with General Access Structure. In A. Cortesi et al. (Eds.): CISIM 2012, LNCS 7564, pp. 41–55. Springer-Verlag (2012)
[18] Hyla, T., Pejaś, J.: A practical certificate and identity based encryption scheme and related security architecture. In: Saeed, K., Chaki, N., Cortesi, A., Wierzchon, S. (eds.), CISIM 2013. LNCS, Vol. 8104, pp. 178-193, Springer-Verlag, (2013)
[19] Hyla, T., Maćków, W., Pejaś, J.: Implicit and Expicit Certificates-Based Encryption Scheme. In: Saeed, K., Snasel, V.,(eds.), CISIM 2014. LNCS, Vol. 8838, pp. 651-66, Springer-Verlag, (2014)

***Authors**: Tomasz Hyla, PhD, E-mail: thyla@zut.edu.pl; Imed El Fray, PhD, E-mail: ielfray@zut.edu.pl; Witold Maćków, PhD, E-mail: wmackow@zut.edu.pl; Jerzy Pejaś, PhD, E-mail: jpejas@zut.edu.pl; West Pomeranian University of Technology in Szczecin, Faculty of Computer Science and Information Technology, ul. Żołnierska 52, 71-210 Szczecin, Poland.*