BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 64, No. 3, 2016
DOI: 10.1515/bpasts-2016-0055

Secure development model for mobile applications

A. MAJCHRZYCKA and A. PONISZEWSKA-MARAŃDA*

Institute of Information Technology, Technical University of Lodz, 215 Wólczańska St., 90–924 Łódź, Poland

Abstract. The development of mobile applications plays an increasingly important role in everyday lives of a visibly growing number of smartphone and tablet users. This trend contributes to changes in corporate management techniques, which now turn towards establishing new standards of data management and extending the functionalities of existing systems to enable the users to benefit from the newest technological advances by creating mobile applications. The paper presents a review of existing solutions for one of the most popular mobile platforms, Apple's iOS, and proposes secure development model as a tool to overcome existing threats faced by mobile application developers.

Key words: mobile application development, security of mobile applications, IOS platform, Android platform.

1. Introduction

Until now the main concern in mobile application development revolved around the functional usefulness of the solutions which were being developed. Together with the growth of the popularity and the number of the smartphone users, more and more real-life applications for the mobile devices were invented and brought to life. Seeing this trend many companies incorporated their own mobile systems, thus establishing new standards of data management as well as extending the functionalities of their xisting systems to enable the users to benefit from the newest technological advances by creating mobile applications. Nevertheless, after the initial phase of going into raptures over the simplicity and multiple possibilities which mobile applications give to their users, the more serious issue of the information security was raised.

Are mobile applications secure enough to entrust them with passwords, account numbers, etc.? Are there any development strategies which should be used to prevent capturing sensitive confidential data by the third-party applications installed on our smartphones? In order to answer these questions it is necessary to talk about the possible mechanisms to secure the mobile applications, to talk about the development strategies which should be used to prevent capturing sensitive confidential data by third-party applications installed on our smartphones.

The increase in mobile Internet traffic is mainly due to the increasing possibilities of mobile devices offered to the users. Nevertheless, these possibilities would not be achieved if it was not for the enhanced development features represented by the newest versions of the operating systems. For the purposes of this paper the security issues of one selected operating systems will be discussed [1, 2, 13, 15, 21].

The choice of the platform seems to be easy, looking at the current trends and usage statistics. Currently the market is led by two most popular mobile operating systems – Apple's iOS and Google's Android. Mobile applications written for these two platforms constitute the majority of the overall number of created applications with the stable position of iOS [4, 14] on the market recently interrupted by the increasing popularity of Android [3, 7].

iOS owes its popularity both to the luxurious brand status as well as to its reputation of being more secure and less prone to external attacks than any other system. That is why the major focus of this paper is put on the existing mechanisms facilitating secure application development and creating new possibilities for their faster and more efficient implementation. Any developer of mobile application for this platforms should be aware of the threats and vulnerabilities that it carries and should adjust the development strategies in such a way so that the optimal level of safety is assured especially when interaction with confidential or sensitive data is required [12, 13].

The presented paper is composed as follows: section 2 presents a concept of a security model which aims to facilitate the work of the iOS developers by enabling the usage of a range of security mechanisms. Section 3 gives the outline of technical details of a tool – a unified framework incorporating safety procedures at different levels of application operations which can be created according to the rules of the model described in the previous section.

2. Secure development model

Preventing the threats connected with development of iOS or Android applications and vulnerabilities characterizing these platforms is not a task which can be realized by a single targeted solution. Some attempts have been made to establish the policy of securing the mobile applications each of them embracing different regions of security breaches [4–6, 17–19]

As it was presented in many works the field of security of mobile applications requires the elevated attention because

*e-mail: aneta.poniszewska-maranda@p.lodz.pl

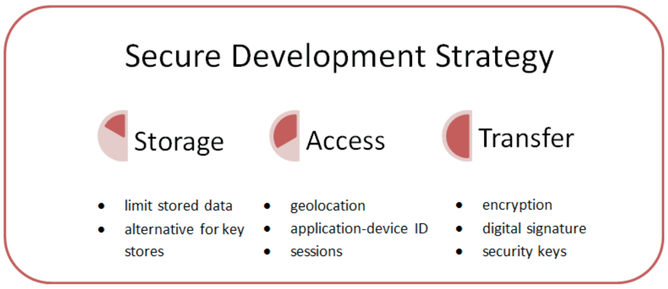


Fig. 1. Pillars of Secure Development Strategy for mobile applications

of the privacy issues of millions of users of smartphones and the lack of adequate solutions to assure the security of data. Having this in mind it is worth to think about more ways of how to reduce the risks connected with the mobile security in the context of the development process itself. One of the most crucial aspects which will be the core of the proposed solution is to create a model which will assure less probability of capturing the sensitive data by the malicious software or hackers [8, 11, 15, 21].

The developed model called Secure Development Strategy (SDS) was introduced for building mobile applications so that they would be less vulnerable to external attacks and leaks of sensitive data (Fig. 1).

The existing approaches to mobile application security focus mainly on the transmission of sensitive data to external services, whereas using the SDS approach equal focus is put on all aspects of the sensitive data management. Safe data transfer between mobile and external devices is undoubtedly a crucial link in the process of securing the applications, however not the only one.

The idea of Secure Development Strategy assumes that application should conform to predefined security standards embracing three main areas: storage (of sensitive data on the mobile device), access and transfer of sensitive data. Conformation to the standards should be achieved by implementing threefold security pattern for each of the mentioned areas. The model specifies the assumptions on how to achieve a proper level of security in each field and provides necessary details on the implementation of mechanisms which will allow achieving desired safety effects.

The foundation of data storage pattern should be based on the limitation of the data stored on the device and (in case the storage is inevitable) proper level of protection achieved by encryption. As far as typical storage mechanisms are concerned they assure the most reliable degree of security, however in case of capturing the device extraction of data from built-in key stores does not constitute a major difficulty. The most desired practice would be then limitation of the amount of the data stored especially this of critical importance. However, as it is not possible to avoid it entirely, additional precautions should be considered, encryption seeming to be of the greatest significance. The main focus for accessing data is put on the identification and verification of the device and the user who attempts to gain an access to the application resources. Also additional access privilege verification process can be implemented for specific methods which require such access.

The last component which is data transfer is dependent on the operation of the system as a whole, therefore the most concerning aspects that will be taken into account are the format, the encoding and the integrity of the data which are going to be transferred. All of the mentioned mechanisms should be supported by appropriate encryption techniques.

2.1. Data storage security model. The first pillar of the SDS storage concerns solely the client-side of the system i.e. the mobile application. The major assumptions of data storage pattern embrace sensitive data encryption, limitation and restricted access (Fig. 2).

While designing mobile applications the developer has two possibilities on where to store application data. He can choose external server where data will be stored in databases and special firewall mechanisms will block access to it. On the other hand some of the information which the application uses are necessary to be stored on the device. The most common reason for this, is application working in an offline mode when there is no communication with the server and the external database and login mechanisms with "remember me" feature.

The first option seems to be a better solution as it eliminates the risk of losing data when the device is damaged. Nonetheless, it requires a large amount of data traffic between the application and the server. In that light the second option comes in handy – it reduces the amount of data transfer. However it seems to be less practical, as the data to be valid need to be updated. Moreover, the storage space of the device is also limited. Thus, the combination of both solutions comes from the need of keeping the data up-to-date and accessible by many devices at any time simultaneously giving the possibility to store a little number of crucial information on the device.

The data stored on the device has mainly device-specific values which enable to identify the users. It is also worth noting that mobile applications are always parts of more complex systems which include server applications, database(s) and web or administration modules.

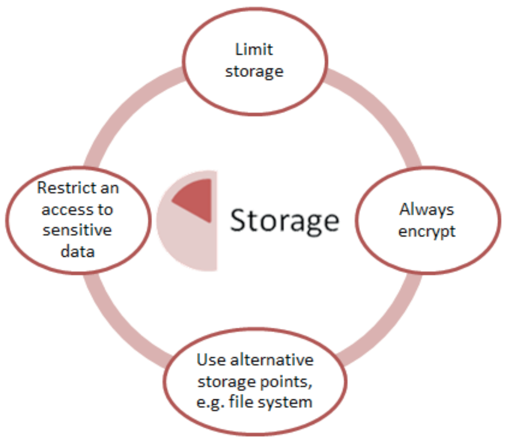

Limit storage
Restrict an access to sensitive data
Storage
Always encrypt
Use alternative storage points, e.g. file system

Fig. 2. Assumptions of data storage model

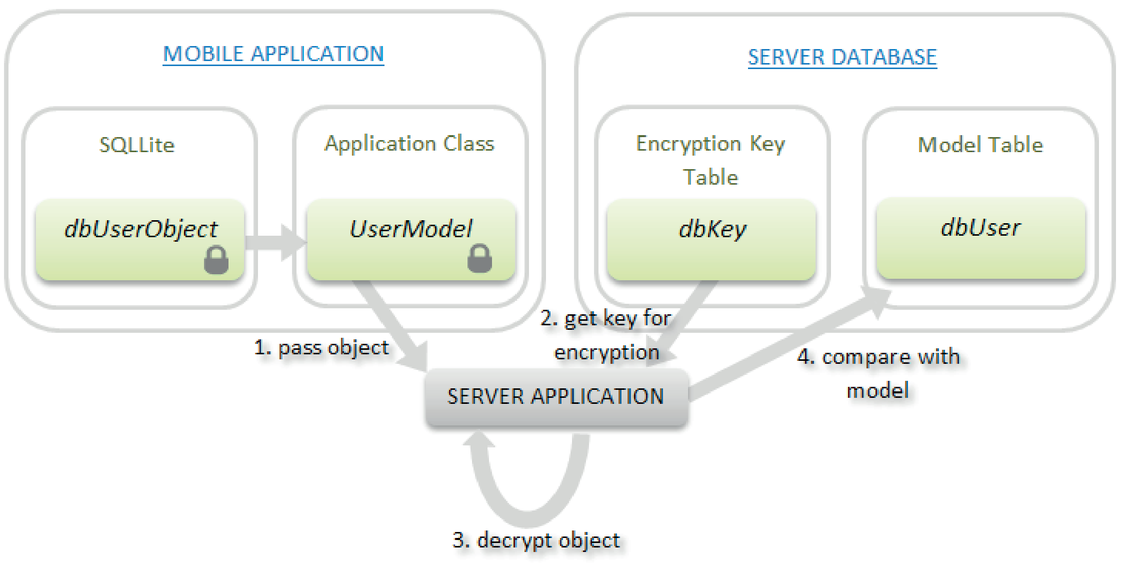


Fig. 3. The object structure of sensitive data stored in the internal SQLite database

The storage mechanisms depend on the place of where the data is saved on the device. Two places for data storage which are also a potential risk points can be discussed for SDS: key-chain and device file system. Three rules regarding the data storage can be formulated as follows:

- sensitive data should never be stored as plain text, but they should always be encrypted and stored as such in key-chains and any other storage places,
- sensitive data could be stored within the application database files and encrypted using encryption keys stored in the external server databases to limit the risk of reading the data,
- access to the internal database objects should be restricted only to the privileged functions (function calls).

2.1.1. Key stores. Mobile platform developers introduced special mechanisms for storing data in a secure manner – key stores. The assumption is that the applications can access only those key store values which they have privileges to. For jailbroken devices there is no problem in reading the key store values with the use of special dumping applications. This is the reason why the first assumption of SDS states that: sensitive data should never be stored as plain text, but they should always be encrypted and stored as such in key-chains and any other storage places.

This is the first step in securing the data, so that even in case they are captured it will be more difficult to use them. This seems to be an obvious guideline, nonetheless many developers do not pay enough attention to the key-chain data encryption because they seem to believe that this mechanism is attack-resistant by default. It is not uncommon that data stored in key-chains is not encrypted or too weak algorithms like MD5 are used for this purpose. One has to remember that the key-chain is the first place the attacker will look for sensitive data and that is why proper care should be taken to secure it.

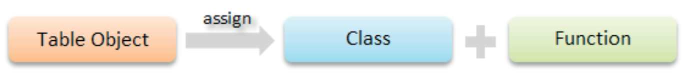
As the suggested encryption algorithm the SDS assumes the usage of AES encryption algorithm as it was chosen as a standard by The National Institute of Standards and Technology [2] for its reliability. AES is a symmetric key algorithm, which means that it uses the same keys for encryption and decryption. The public encryption key should not be stored or hardcoded within the application itself but it should be securely stored on the server database. Moreover, a good practice is that the key was different for every device using the application – the server application should generate the key when the application is installed on the device.

The other suggestion for storing sensitive data is to transfer the responsibility from key-chains which are known to store valuable information, especially remembered passwords, to the internal database of the application. This is expressed as the second rule for data storage: sensitive data could be stored within the application database files and encrypted using encryption keys stored in the external server databases to limit the risk of reading the data.

Both platforms – iOS and Android – enable the developers to make use of SQLite databases and store information within them. They are most often used to keep frequently exploited data, so that it not have to be downloaded from the server any time it is needed. Data in such database is kept independently from the application being active, so that it is preserved on the device all the time. On the other hand the sandboxing mechanism imposes that such database can only be accessed by the application it was created for.

2.1.2. File system. Sensitive data could be stored within the internal database as custom structures known only for developers of a specific application. This is not a common practice at the time. The SQLite data records are stored in a file on the device, thus it may seem a hazardous thing to store the passwords there. That is why credentials and simple character variables should be still stored with the use of key-chain stores, however in case of other structured data one should definitely use the internal database of the device. According to SDS if certain conditions of securing such data are abode the database will constitute an equally safe place to store it (Fig. 3).

The conditions to securely store data within the database state that all data stored in such a way should be encrypted us-



```
[DBManager assignPermissionOfObject:<DBTableName>
                          forClass:<ClassName>
                       andFunction:<FunctionName>];
```

Fig. 4. Table object-function permission assignment

ing the appropriate hashing functions and for records containing sensitive information the AES encryption should be used. Each table should have a corresponding class defined within the application structure. For each class a corresponding table on the server database should exist. Additionally, for each table which stores sensitive data on the server database, a unique public key used for encryption should be generated. Thus, additional table storing public keys should be used for this purpose.

When using the encryption described above, even in case of accessing the file by the unprivileged third-parties it would be impossible to read them without the decryption key. Additionally, the database objects should only be accessed by privileged functions specified during application development process. The permission-based model similar to the one encountered in Android could be implemented. The permission should be assigned as a pair, as visible in Fig. 4, defined in a separate file and stored as XML.

The aforementioned suggestions for data models do not prevent attackers from capturing the data from key-chains or files, but they aim to secure this data, so that even in case of capturing they would be of no real value.

2.2. Data access security model. The second pillar of SDS strategy concerns the access to data. This comes from the fact that mobile applications need to communicate with external services and other applications.

The major assumptions of this area of security embraces three mechanisms which aim to enable identification of the user requesting access to application resources (Fig. 5).

The access to the application resources can be controlled at different levels. Firstly, one may consider an access to the resources by application functions which is vital in case if anyone tries to modify the behaviour of the application by for instance swizzling method. This issue was partially addressed in the previous section regarding permission-based model to the database objects which may contain vital information. The other point of view is an access to data by unprivileged services or remote calls to the server from parties impersonating valid devices. The SDS strategies for such threats include the use of geolocation and the device identifier as means of distinguishing suspicious behaviours.

The fundamental SDS rules for the data access are:

- the mobile application should inform about the current location of the device every time it requires an access to sensitive data,
- the mobile application should always present itself with a digital signature composed of unique device identifier,
- server should always check whether the device session is open before it realizes any requests.

2.2.1. Geolocation. The geolocation became a common mechanism for locating devices and performing certain services which enable to display proper language or time zone for the user. Recently Google has used this mechanism to secure access to private user accounts. The Google procedure works in such a way that the location of the computer together with date and time of the registration and consequent logins to the account are registered and then used for validation. The location of the computer is obtained on the basis of the IP address of the computer. If the current and the last activity data are significantly divergent i.e. 1 hour after logging from Warsaw someone tries to login from Seoul, then automatically an e-mail or an SMS is sent to the user informing about such attempt. This enables to track who tried to access the account and records all login attempts.

A similar mechanism can be implemented for a mobile application. Having a unique identifier of the device, its location could be registered and sent to the server with every request issued by the application. It would serve as a frame of reference for future logins. In case a suspicious attempt would take place a message should be sent to the user. This would require a mobile phone number or email specification by the user or the application could notify itself about the attempt to login with the use of push notifications. Additional structure on a server should be implemented as visible in Fig. 6.

The geolocation service is an increasingly popular one and used by many companies for location purposes. However, it can still contribute to increasing access security of data within the

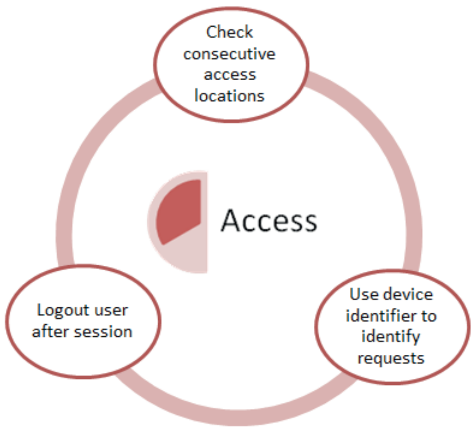
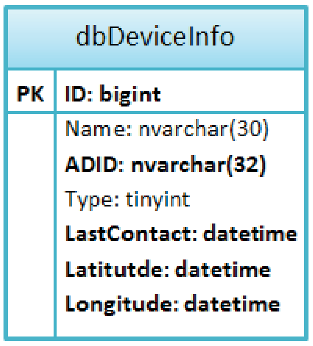

Check consecutive access locations
Access
Logout user after session
Use device identifier to identify requests

Fig. 5. Assumptions of data access models



dbDeviceInfo	
PK	**ID: bigint**
	Name: nvarchar(30)
	ADID: nvarchar(32)
	Type: tinyint
	LastContact: datetime
	Latitutde: datetime
	Longitude: datetime

Fig. 6. Structure for storing device information data

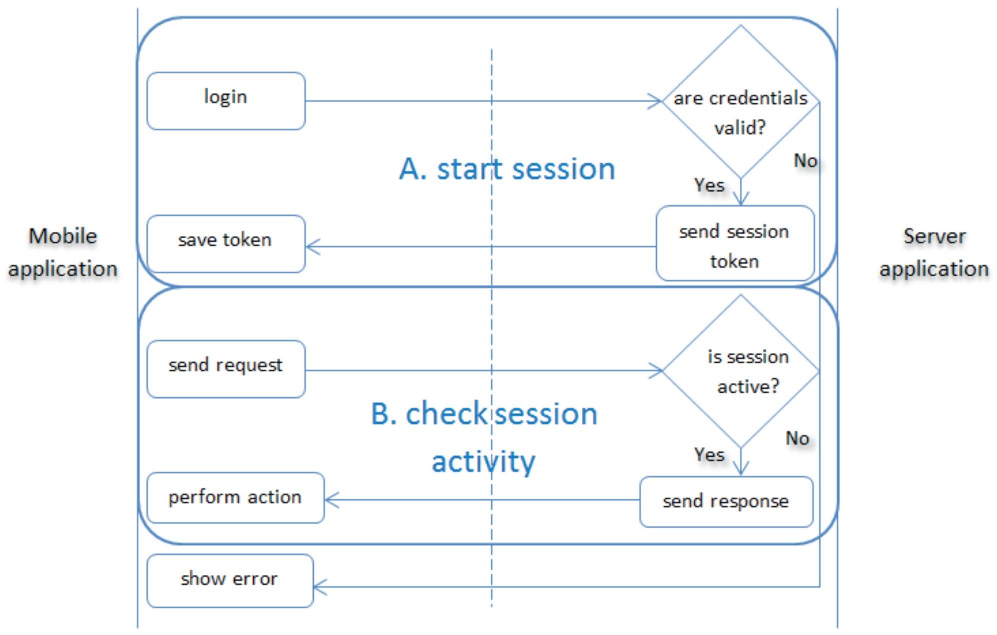


Fig. 7. Scheme of session mechanism

application. Any modern mobile device nowadays is equipped with a GPS system and offers its own location services and built-in mechanisms. They are most frequently sufficient for the purposes described above as it is not required to locate the device with the accuracy of one meter. Some developers may decide to use Google Maps API which provides a more complex and accurate solution. Google Maps API is available for all mobile platforms and it can be easily implemented. Of course a built-in iOS MapKit is also a sufficient solution for this purpose.

2.2.2. Unique device identifier. Every device is assigned a unique identifier which enables to distinguish it from the others. The transmission of a unique identifier can somehow violate the privacy as it could be captured by unwanted parties. Nevertheless, Universal Device Identifiers (UDIDs) seem to be the most reliable way which can be used to identify the requests for the data sent to the server. Together with the geolocation coordinates they can prevent external invalid requests to be served.

Recently, Apple has raised the issue of too aggressive aggregation of UDIDs for marketing purposes. In the earlier releases of iOS 6 the old interface methods for UDID were deprecated and Apple suggested that developers should find another way of identifying the devices. They stressed the fact that devices may be lost, destroyed or stolen, so the user accounts cannot be stiffly linked to device addresses. That is why they suggested to use rather the username/password scheme instead of device identifiers. However, still there exists the possibility to use UDID characteristic for the given vendor.

In order to conform to Apple standards it is best not to use the default device UDID, but it does not change the fact that we do not want to resign from the possibility of identifying a specific device. One has to remember that many users may log in to the application from the same mobile appliance. This is the reason why SDS suggests that a custom unique identifier was generated for the device during the installation phase. This unique identifier will be referred to as an Application-Device Identifier (ADID) and it will be characteristic for a single application only. That is why it will not be possible to use it for any other purposes than for the application in subject. ADID should be a 32-bit string generated randomly. Its uniqueness should be checked during the generation phase and it should be preserved at server-level, i.e. there may exist two devices with the same device ID if they are served by different servers.

ADID could serve as a “username” of a device and only by confirming it in the requests it would be possible to access the data. If the uniqueness of ADIDs for the entire application, independently from server instances was to be preserved, one may consider combining a modified part of the real UDID to the generated string.

Reading the aforementioned description one may wonder why geolocation and unique device identifiers are used even though earlier it was stated that sending too much information about the device violates privacy of the user. This situation presents a trade-off between security and privacy issues. One has to remember that in order to stay secure some form of control is required on what is happening within the application. Both parts of the system – server and mobile application – are prone to attacks and that is why both of them need to exchange the information concerning their identity. Otherwise, the lack of such mechanisms would enable unwanted parties to impersonate valid devices without consequences. Furthermore, as according to the SDS rule, the created custom unique identifier ADID is not transmitted directly but encrypted, therefore it prevents to identify the device by third-parties.

2.2.3. Sessions. Another precaution which should be taken into consideration while building a mobile application is the use of the session mechanism (Fig. 7). Sessions are commonly used in

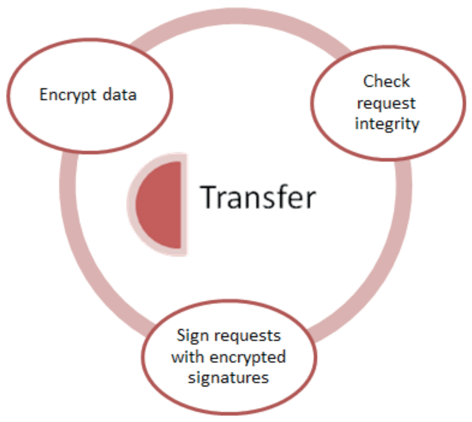


Fig. 8. Assumptions of data transfer model

case of web applications. During login the user opens a session on the server. The timeout for the session should be defined. If for that specified amount of time no action is taken the user can be logged out. The other option is that if the request comes after the specified time the user is asked to input the password again and only then the request can be realized.

2.3. Data transfer security model. Data transfer pattern refers to all mechanisms which involve the exchange of data between the mobile application and the external services. These mechanisms should incorporate in their action flow the additional security procedures – data encryption, the use of security keys and the verification of the requests integrity (Fig. 8).

Although this is the last one of the described pillars of SDS, it is absolutely not the least important one. On the contrary data transfer seems to be the weakest link in the entire process of the mobile application development. It comes from the fact that the requests are travelling over the Internet in an unprotected space and they are prone to a special kind of attacks called the "man in the middle".

This attack means that between the mobile device and server application a third-party may be listening and waiting for the exchange of information. There exist three types of attacks for "man in the middle" scenario:

- attack on the privacy of data – stealing confidential information,
- attack on the integrity of data – changing the content of the message,
- impersonation – impersonating other device/user.

2.3.1. Encryption. Firstly, the data sent to the server should always be encrypted. Special care should be of course taken of data which are considered to be sensitive like passwords, credit card numbers and others alike. Encryption is crucial in case of the first attack scenario regarding privacy – even in case of capturing the data, the content of the message will be difficult to read. The quality of the encryption depends of course on the used algorithm and the method of securing the encryption key.

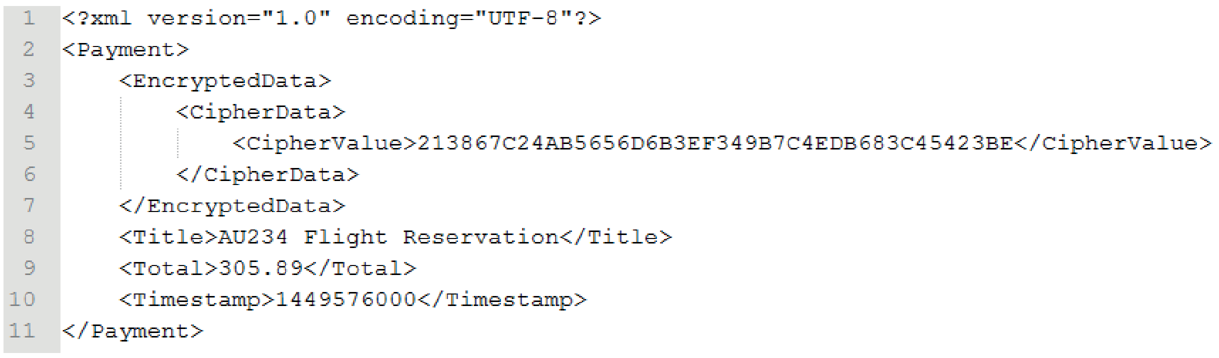
All mobile platforms enable to use built-in libraries for encryption of string data. The messages sent to the server are most frequently transferred in the JSON or XML formats, but these libraries also have appropriate interfaces to handle such kinds of arguments. Figure 9 presents XML message sent from a mobile application to the server with the AES encoding on confidential data applied respectively.

2.3.2. Digital signatures. As far as the integrity of the data is concerned it is a good and common practice to use the digital signatures. They enable to check whether the message received is exactly the same as the message sent and if it was not modified on the way to the receiver. The digital signature mechanism relies on the encryption which should assure the authentication.

The *Digital Signature Standard (DSS)* relies on a private key algorithm. Only the owner of the message knows the private key and the public key is known. It is however possible to use other encryption techniques and implement them instead. One of the oldest encryption algorithms are different checksum algorithms. The checksum transforms an input using a checksum function and produces a numerical output of a small size. A good checksum function should provide a significantly varied checksums even if the message changed only slightly. The disadvantage of this method is that checksum algorithms are known and could be tricked and reversed.

As far as relation between encryption of a message and digital signature comes into view there exist four types of situations (Fig. 10):

- message is only encrypted – it can only be read by a person having the key, but the integrity of the data is not known,



```
<?xml version="1.0" encoding="UTF-8"?>
<Payment>
    <EncryptedData>
        <CipherData>
            <CipherValue>213867C24AB5656D6B3EF349B7C4EDB683C45423BE</CipherValue>
        </CipherData>
    </EncryptedData>
    <Title>AU234 Flight Reservation</Title>
    <Total>305.89</Total>
    <Timestamp>1449576000</Timestamp>
</Payment>
```

Fig. 9. Encrypted XML message sent to the server

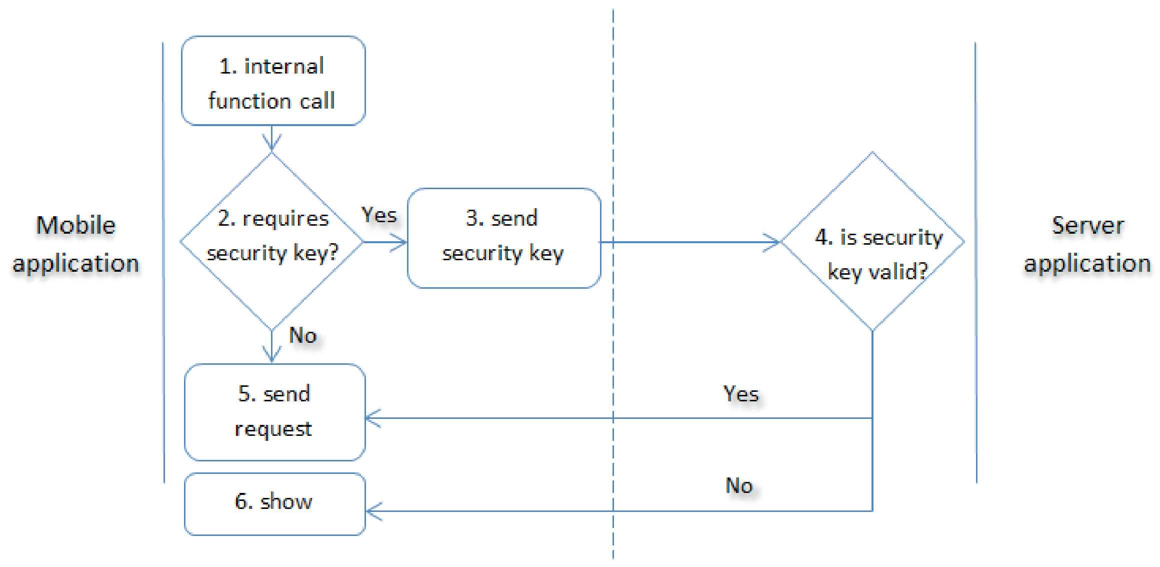


Fig. 10. Data transfer using encryption and digital signatures

- message is digitally signed – the integrity can be checked, but everyone can read the content,
- message is first encrypted, then digitally signed – no one can read it, the integrity is preserved,
- message is first digitally signed, then encrypted – no one can read it, the integrity is preserved.

2.3.3. Security keys. The last suggested mechanism represents security keys which seem to be valuable in case of using checksum algorithms for digital signatures. Security keys introduced by SDS are 128-bit strings which should be sent to the server before the request for sensitive data. The security key can be unified for the entire application and independent from the device. When transferred it should also be encoded with chosen cryptographic algorithm. The process of exchanging data with the use of security keys should look as follows (Fig. 10):

1. the user invokes a function which requires communication with the server,
2. the application checks that the request to be issued requires the security key confirmation,
3. the security key compliance request is issued to the server,
4. the server responds with a message,
5. if the security key is invalid, the operation cannot be performed – alert message is returned to the user,
6. if the security key is valid, the request for the initial operation is issued.

3. iSec framework as the implementation of the SDS model

The aforementioned model for building the secure mobile applications specifies the guidelines which should be taken into consideration while creating such applications by the developers. It is however common knowledge that the conditions under which most of the developers work – tight schedules, close deadlines, performance efficiency – do not serve well assuring the impeccable security.

The implementation of ready-to-use classes and methods which would assure security of any mobile applications seems to suit the current needs for simultaneous time-efficiency and safety. That is why in the scope of this research the prototype framework *iSec* was developed. Its main components correspond to three pillars of the security model presented in the previous section: storage, access and transfer. Additionally, addressing a practical need for a fast login mechanism, the component generating classes and controllers indispensable while creating login views was designed and incorporated as a part of this framework. This mechanism enables also the use of roles, which are frequently applied for the mobile applications.

All the major components of the framework use the encryption module which provides basic and complex encryption methods gathered together for the simplicity of use. Its user will be able to specify the default encryption algorithm to be used or he will use different ones for different purposes. Another component common for others is the *XMLParser* which will enable to read and write the security configuration stored in an external XML file. This configuration refers to the variables set from all three basic modules, with the most significant usage of function privilege information. The component model of created framework is presented in Fig. 11.

4. Conclusions

While designing and building mobile applications there will always be a trade off between functionality, optimization and security. In the long run, however, none of these areas should be omitted and treated with less importance than the others as only balanced combination of those three will enable to create the reliable and useful solutions.

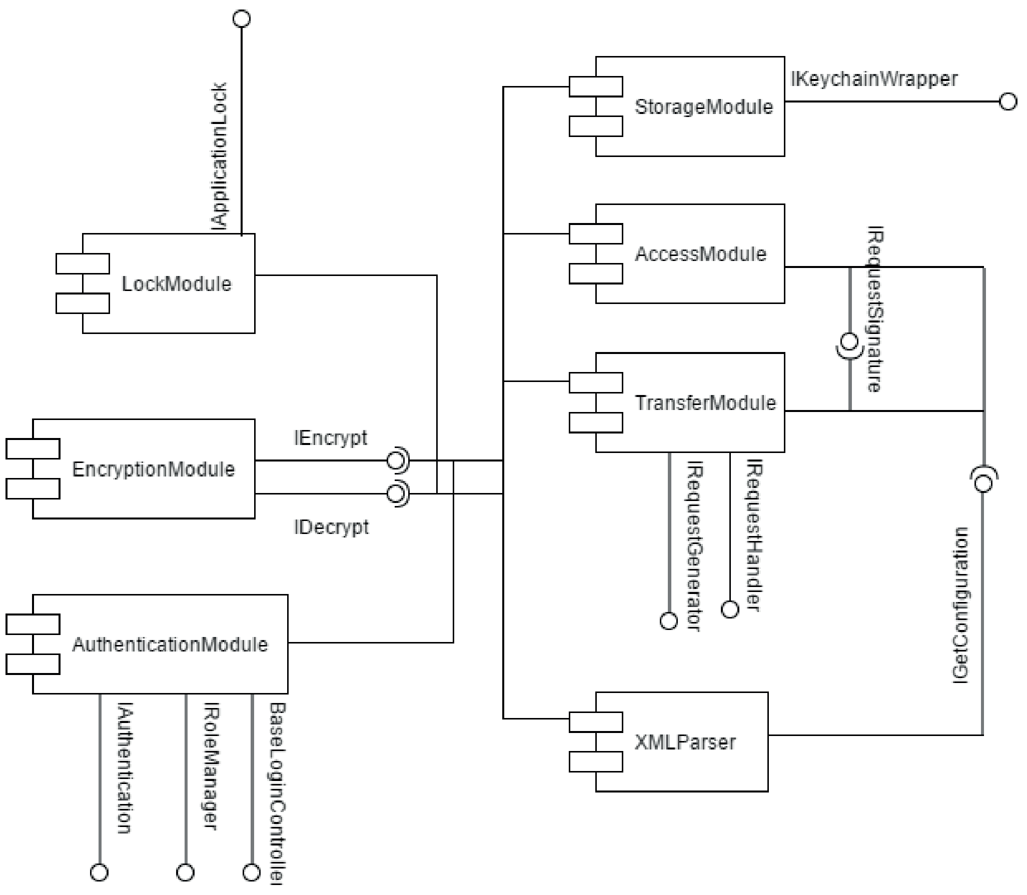


Fig. 11. Components of the *iSec* framework

Currently there is no standardized solution or methodology for developing the applications which would be ultimately threat-resistant. Together with the invention of newer and newer strategies to prevent the security breaches, more sophisticated threats emerge and new vulnerabilities are detected by the platform developers.

The Secure Development Strategy for mobile applications, presented in the paper, introduces three pillars which should be taken into consideration while designing and implementing the mobile applications. All these pillars: data storage, data access and data transfer should be treated as equally significant throughout the entire development process. The SDS provides details on its assumptions and mechanisms which should be implemented within the application framework in order to provide the mobile security.

No strict standards are provided, rather suggestions for the use of technologies and solutions. The most vital feature of SDS is that it embraces all crucial aspects of mobile application development and systematizes them by defining the concepts and categorization. The main assumptions of SDS include using the geolocation and custom device identifier ADID for accessing data, storing the sensitive information encrypted in database files and securing the requests sent over the Internet with digital signatures and security keys. The SDS does not aim to make the application fully attack-resistant, but to disable the potential attackers from encoding sensitive data and to reduce the amount of potential risk points in the application.

The practical implementation of the components of SDS strategy is realized with the use of created security framework – *iSec*. The *iSec* framework constitutes a tool which enables the usage of a variety of security mechanism without the necessity to use additional external sources. It provides the implementation of the functionalities defined by the SDS. Its features aim to simplify the process of implementing the security into mobile applications.

References

[1] A. Porter Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, “A survey of mobile malware in the wild”, *Proc. 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, 3–14 (2011).

[2] M. P. Souppaya and K. A. Scarfone, “Guidelines for managing the security of mobile devices in the enterprise”, *NIST* (2013).

[3] Y. Zhou and X. Jiang, “Dissecting Android malware: Characterization and evolution”, *Proc. 33rd IEEE Symposium on Security and Privacy* (2012).

[4] Y. Agarwal and M. Hall, “ProtectMyPrivacy: Detecting and mitigating privacy leaks on iOS devices using crowdsourcing”, *Proc. 1th Annual International Conference on Mobile Systems, Applications, and Services*, 97–110 (2013).

[5] T. Werthmann, R. Hund, L. Davi, A. Sadeghi, and T. Holz, "PSiOS: Bring your own privacy and security to iOS devices", *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, 13–24 (2013).

[6] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "PiOS: Detecting privacy leaks in iOS applications", *Proceedings of the Network and Distributed System Security Symposium* (2011).

[7] T. Vidas, D. Votipka, and N. Christin, "All your droid belong to us: A survey of current Android attacks", *Proc. 5th USENIX Workshop on Offensive Technologies* (2011).

[8] N. Seriot, *iPhone Privacy*, Black Hat DC, Arlington, 2010.

[9] W. Enck, M. Ongtang, and P. McDaniel, "Understanding Android security", *IEEE Security & Privacy* 7 (1), 50–57 (2009).

[10] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, "Semantically Rich Application-Centric Security in Android", *Proceedings of the 2009 Annual Computer Security Applications Conference*, 340–349 (2009).

[11] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A study of Android application security", *Proc. 20th USENIX Security Symposium* (2011).

[12] W. M. Fitzgerald, U. Neville, and S. N. Foley, "MASON: Mobile autonomic security for network access controls", *Journal of Information Security and Applications* 18 (1), 14–29 (2013).

[13] S. Khan, M. Nauman, A. T. Othman, and S. Musa, "How secure is your smartphone: an analysis of smartphone security mechanisms", *Proc. International Conference on Cyber Security, Cyber Warfare and Digital Forensic*, 76–81 (2012).

[14] J. Zdziarski, *Hacking and Securing iOS Applications*, O'Reilly Media, 2012.

[15] M. Alhamed, K. Amir, M. Omari, and W. Le, "Comparing privacy control methods for smartphone platforms", *Proc. Engineering of Mobile-Enabled Systems* (2013).

[16] L. Kastenson, *Security of Mobile Devices*, 2013.

[17] M. Elkhodr and S. Shehrestani and K. Kourouche, "A proposal to improve the security of mobile banking applications", *Proc. 10th International Conference on ICT and Knowledge Engineering* (2012).

[18] D. Floyd, "Mobile application security systems (MASS)", *Bell Labs Technical Journal* 11 (3) (2006).

[19] A. Zaheer, F. Lishoy, A. Tansir, C. Lobodzinski, D. Audsin, and J. Peng, "Enhancing the security of mobile applications by using TEE and (U)SIM", *Proc. 10th International Conference on Ubiquitous Intelligence and Computing* (2013).

[20] X. Feng, Y. Wu, and X. Yan, "Mobile application protection solution based on 3G security architecture and OpenID", *Proc. 7th International Conference on Software Security and Reliability Companion* (2013).

[21] A. Michalska and A. Poniszewska-Marańda, "Security risks and their prevention capabilities in mobile application development", *Information Systems in Management, WULS Press* 4 (2), 123–134 (2015).