Mobile client security architecture: a practical approach

1. About the protection of cloud.
2. Mobile environment problems.
3. Offline mode problem.
4. Other works.
5. Our proposed methods: SSS, ABE, MOS
6. Our architecture.
7. Detailed workflow description.
8. Analytics in detail.
9. Common dangerous scenarios.
10. Analysis of security and complexity.
11. Implementation.

Cloud computing is a new paradigm in the world of distributed networking and computation. The basic feature of the cloud environment is providing the elastic, on-demand and secure service for the end-users. While the first two requirements are rather well supported by the cloud platforms in use, the security is a major concern of the cloud providers and governmental organizations as well as academia and research community [1], [2], [3]. For the small and medium-sized enterprises (SME) the cloud environment is often the most cost-effective and easily scalable solution. However, the security and privacy of the sensitive data in the cloud is a major issue.

A common practice to provide a stable security cloud solution is to use a specific type of cloud service: CASB – Cloud Access Security Broker or CAC - Cloud Access Control. These services are specifically designed to bring the security at a single access point and provide the coordination of the most important security measures. It is estimated by Gartner [4] that such systems will be used by 85% of companies by 2020. The reason for this is that the organization of the security measures at a single control point allows to control and to monitor the level of cloud protection much more effectively. The basic features of the CASM are discovery of cloud services, encryption (along with tokenization for better search properties), access control, DLP services, authentication and auditing/alerting services [5]. The protection of the confidential data, according to the standards of CASB deployment should be provided elsewhere, i.e. in transit and in the user side [6].

The additional security problems and requirements need to be considered when the mobile devices are actively used in the cloud environment [7]. Today the society lives in the BYOD (Bring-Your-Own-Device) world and the mobile devices pose a serious risk to the SME cloud platforms as the bottleneck of the information security system (ISS). While the enterprise web interface and the cloud environment can be protected by powerful third-party services, such as CASB and CAC, the mobile client is usually light-weighted and generally less protected. The protection scheme used on a mobile device should be both computationally secure as well as resource-constrained due to the battery power limitations [8]. Therefore, the encryption on a mobile device is not a good solution: the proposed schemes are computationally good, but lacks the security analysis in many cases [9]. The common practice is the shadow user activities monitoring [7]. However, one area of the mobile device usage stays unprotected in all the proposed scenarios.

Suppose, a SME uses CASB in order to protect data at rest (i.e. while being store on the server-side), in transit (communication with server), in use (while the client is connected to the network). However, what happens when the mobile client goes offline? And even worse - with some sensitive data on board? All powerful cloud-based tools cannot help and the mobile app has to secure itself with its own limited resources. Moreover, the critical point is the difference in strategy of online and offline protection. Due to its resource constraint, it is not possible to perform the extensive computation and encryption on the mobile device.

Observing the above described landscape, this paper considers the concept of the offline mobile client security. The proposal describes a novel approach based on powerful cryptographic preventive methods, such as secret sharing [10] and ABE encryption [11]. Also, the proposal includes the usage of the user behavior analysis in order to reduce the risks and the harm of the most common threats: the expired user misusing password, and the intruder attack. With this goal, the key expiration period is used and safely incorporated into the proposed system solution. The main target is to provide a maximum defense at the minimal resource cost.

The paper is structured as follows. Section 2,analyzes the most common security problems in the mobile cloud environment and the proposed solutions for them. The text mainly concentrates on the offline protection in the BYOD world. Section 3 gives the basic definitions and explanations of the used methods: SSS, ABE and MOS. Section 4 presents the complete proposed solution to the problem of offline mobile client security. Section 5 traces the workflow of the client activity and analyzes the common security breaches. As discussion about the security proofs for the proposed system is also provided in Section 5. Section 6 presents the practical implementation and analysis of complexity of the proposed solution, while Section 7 concludes the paper.

**Related Work**

The BYOD world requires more from the traditional data protection services compared to ordinary computing environment. Apart from authentication, DLP services and encryption (at rest in transit, in use) it is necessary to provide additional contextual methods [REF]. The contextual methods increase the security of the app at a maximum level with a minimum resource demand. The traditional ones are:

1. Using geo location
2. Expiration of an app
3. Secured transfer between apps
4. Restricted access to app
5. Expiring pass/pin
6. Counting of ailed tries
7. Offline protection approach.
8. Audit

Therefore, it is possible to conclude that it is harder to protect data on the mobile device so the owner or the SME should take more care of protecting its data leaving the organization. According to this perspective, the data on a mobile device should be considered as one leaving the organization. The most sensitive and confidential data should not be allowed to be transferred to the mobile device at all. However, what if the SME needs to allow the workers to work on such devices and even use them in the offline mode for the convenience and traffic reduce or even for a particular characteristic of the app or the business itself?

From the theoretical point of view on this problem several surveys can be considered [REFs]. The mobile cloud computing is a rapidly developing paradigm that poses many security and complexity problems. This type of systems requires new models of application and the new way of using data storage services. An analysis of the existing models of mobile cloud computing is presented in [8]. All the models and protection schemes concentrate on the encryption properties and either perform the computations on their own [12-15] or use the cloud provider to off-load the cryptographic operations [16-18]. It is natural that the mobile device cannot handle the operations securely without assistance of a cloud provider, due to the resource constraint and battery limitation.

When it is desired to make the device more independent and less dependent on the cloud provider (suppose, an application needs to run securely without connection to the network) it is possible to use only the schemes that function without putting load on a provider. All the currently known schemes of encryption, performing the computation, as it is discussed in [8], either use the a cloud provider [12], a third party trusted agent [13], a combination of both [14], or concentrate on computational complexity without taking care of user privacy and security[15].

In other words, the security schemes proposed so far, are not working offline. In many cases the industrial providers of the secured application the security functions API, that protects the mobile app like a wrap (like Mocona, operating along with SAP [16]), preferring to completely forbid the offline access to the protected app.

In some cases such access is still necessary. Due to such constraints as traffic load, travelling, ease of access and many more. So the question is how to protect the device/app, in an offline mode when the functions of data protection cannot be offloaded to a cloud or a trusted party. This problem was not approached neither in academia nor in the industry.

**Offline mode**

This paper proposes an open model of the mobile device protection mechanisms in which the security is supported both in online and offline modes. Currently, the systems of mobile device protection, to the best of the authors knowledge, follow the model where the protected mobile client can operate only when it is connected to the cloud, which is not always convenient for the end-user. The basic principles of the mobile device protection that is supported by the here proposed approach are:

1. Optimized communication with the cloud when the device does not need to be constantly connected to the server due to the resource constraint and necessity to secure this communication.
2. Implementation of reliable cryptography standards, i.e. the algorithms AES, ABE and SSS are approbated and well-known. The idea is not to invent obscure concepts or to invent new methods that should be evaluated before the proposed usage in the provided solution.
3. Optimized combination of the security mechanisms so that the mobile client does not need to perform complex computation like encryption and key generation due to its resource constraint.

The most important security issues in the proposed model arise when the device goes to the offline mode and the user is still allowed to get the access to the strategic organization documents. In this case, the server can neither monitor the user activity nor provide the protection methods. The security becomes the responsibility of the mobile client. Additionally, the maximum protection should be provided at the minimum resource cost.

In the online mode the device communicate with the server in order to check the validity of user password. On the contrary, the offline protection should implement a different approach. In other words, the authentication/authorization mechanisms in the offline mode should utilize the specific proof of the user identity. The requirements for the proof are as follows:

1. The proof is derived from the previous session in order to verify that the user is still authorized.
2. The proof should not give access to user password, i.e. it should not be stored in the device.
3. The proof shoud be temporary and have an expiration period.
4. It should not be directly used in communication with the server in order to prevent the malicious user from mimicking
5. It should be resilient to the off-line dictionary attack
6. It has to stay effective both in the scenarios of the malicious outsider and leakage of information when the formerly authorized user leaves the group.

If such proof can be constructed, then the offline mode can be secured. The most important requirement is that the password (or the proof of the password) cannot be stored on the client device, as it is not possible to guarantee that it is perfectly protected in this case. The client cannot be in possession of the user knowledge as these are the separate entities. Rather, this proof should be shared between the client and the user in a secure manner. This is the only way the client can support the security without performing complex computation or storing the function of the user password.

In the mobile client, the user keys are protected by the combination of user password and PIN as well as the ABE keys that have an expiration period. The additional argument against the traditional password verification is the necessity to check the PIN, which is very small, so the construction of valid one-way function resilient to the offline dictionary attacks is a difficult task.

**The algorithms and definitions.**

This proposal consists of a combination of secure and effective methods in order to protect the client as detailed explained in the following.

1. AES encryption to protect the files.

The protected files are encrypted on the server with the secure and well-known AES cypher, which is currently an industrial standard. Other options include Blowfish and Serpent. All these are block symmetric cyphers providing high level of safety. In other words without knowing the AES key it is impossible to decrypt the files.

The AES encryption key is normally used as a session key. It is not desirable to re-encrypt the files stored in the device unless there is a certain condition (user leaves the group and the file should not be accessible). That is why AES key in the presented notation is defined as FILE\_KEY. If it is not desirable that the user have access the file, the system kills the FILE\_KEY, so that user has to perform a hard cryptanalysis to reveal the file. Not that for each file the FILE\_KEY is randomly generated on server and unique. So, the user needs to perform the cryptanalysys for each file separately.

1. ABE encryption to protect the FILE\_KEY. The permanent file key is re-encrypted with each session. The KEY\_SET is defined as the set of all FILE\_KEYs. The KEY\_SET\_KEY protects this KEY\_SET (a symmetrical AES key). The proposed model supports both the authorization based on user groups and on the shares, i.e. there is a separate KEY entity corresponding to each share in the group.

The selective scheme for attribute-based encryption is as follows.

If at least one attribute in the set{t\_i}\_U is equal to the attribute in the set{t\_i}\_M, the corresponding user U can decrypt the text M.

As soon as user and share have one attribute in common – the user can get access to the share.

1. **Master-key (MK)** which is kept safely on server and accessible only for the domain administrator

*The values ti are randomly selected from the huge group Zp . They are the private keys corresponding to the group attributes. Note, that this is different from the usual PK encryption: the private keys are controlled by the admin and not by the users.*

1. **Public key (PK)** depends on the master key values and is kept in the clear allowing users to access the information:

Here e(g,g) is the bilinear pairing.

1. Secret user **KEY\_SET** depends on his attribute set. Here each Di (**GROUP\_KEY**) serves for decryption of the data of a single group of users, for example, related to some project:
2. **Encryption** of the text M (in this proposal the text is the **FILE\_KEY**, or the permanent AES symmetric key, the key is permanent in order to reduce the necessity to re-download the files on the device) is multiplication. The set of the public keys Ei (**PUBLIC\_SHARE\_KEY**) corresponding to the set of groups able to access the text is kept along with the encrypted text:
3. **Decryption** is division:

In order to perform this operation the user needs Di corresponding to the secret attribute ti and :

*The result of decryption is the FILE\_KEY - the symmetric AES key to decrypt the contents of file.*

1. The KEY\_SET\_KEY is protected via the secret sharing. In other words, it is considered as a secret value and it is split (by the modular sharing) into the set of 4 shares:

PASS+PIN+TIME+DEV\_PASS (where the PASS and PIN are predefined, as in [brest] and TIME is modified ). In order to get the KEY\_SET\_KEY the hacker has to get all 4 parts (otherwise he gets no information of the secret due to the perfect nature of SSS).

The proposed authentication system is based on the shared storing of the user key. Also, the device acts as a dealer in the SSS. Using the SSS ensures that the key can only be accessed by an authenticated user. The participants of the (2, 2)-threshold SSS are the user, device and the time mark. The user share  is computed based on the **-**and the PASS entered by the user. The TIME is the current time value. Let  and , where – is a one-way function that transforms the data into the string of the desired length:





According to the CRT:



Thus calculated DEV\_KEY is written to the permanent memory. The user share is not saved. Otherwise, it would allow an attacker to locally validate the restored private key.

1. MOS (description)

**Offline mode: proposed solution**

Figure XXXX shows the workflow of the proposed mobile application in the offline-mode:

The approach is based on the combination of AES-ABE-SSS methods. The key feature of this approach is that the client does not actually store any part of the user password to be verified. The client combines its own key with the user share (PIN and password-derived) in order to restore the initial KEY\_SET\_KEY. If the user provides the wrong share the client will not be able to recognize it, but will decrypt the files incorrectly.

*Security analysis*

The client does not store the user password, i.e. no information about the password leaks, and also there is no possibility for the malicious user to check if the password he is trying to enter is correct. The only possible scenario for the information leakage in the case of the malicious outsider is:

* 1. The hacker steals all the parts of KEY\_SET\_KEY
  2. The hacker steals the KEY\_SET
  3. The hacker try the brute force offline dictionary attack on all the previous values, they have to belong to ONE TIME SESSION (the values of ”a” and ”b” belong to one period of time i.e. TIME, DEV\_PASS, KEY\_SET).
  4. Steal the permanent FILE\_KEY (this is protected by the KEY\_SET).
  5. Steal the file and try to decrypt it with offline dictionary.

The hacker has to get 4 values: TIME, DEV\_PASS, KEY\_SET, FILE\_KEY - from one session. At the same time he should try the offline dictionary attack on PIN+PASS. Moreover, the 4 values provide access only to 1 single file. So practically, it is very difficult to perform such attack due to the key expiration period.

The temporary nature of all parameters obliges the user to connect to server when necessary and prevents the malicious actions from the user side. The only possible scenario when the bad-willed or expired user wishes to prolong his old credentials is:

* 1. He has to steal the DEV\_PASS and TIME synchronized with his credentials.
  2. He has to be able to combine.
  3. He has to steal the KEY\_SET synchronized with his credentials.
  4. He has to steal the protected FILE\_KEY (also synchronized) and file.
  5. He has to do everything without the client (because the client checks the TIME and renews the PUBLIC\_SHARE\_KEY)

Basically, for a malicious user there is practically no way to use the client with the old keys. The fact that the client does not contain any check data does not prevent the user from seeing the contents of decrypted files, but with the wrong password the decrypted files will be totally wrong.

The client still has to count the tries (to prevent the hacker brute force attack) within one session.

*Complexity analysis*

The client actions are:

1. Combine the PASS+PIN+TIME+DEV\_PASS=KEY\_SET\_KEY ------- SSS secret restoring
2. Decrypt the KEY\_SET with the KEY\_SET\_KEY --------- symmetric decryption
3. Select the SHARE\_KEY from the KEY\_SET -------------- no calculation
4. Decrypt the FILE\_KEY with the SHARE\_KEY ---------------- ABE decryption
5. Decrypt the file with the FILE\_KEY ----------------------------symmetric (AES) decryption
6. Modify the TIME periodically ------------------------------ timer
7. Count the tries within the TIME ---------------------------count
8. Modify or delete PUBLIC\_SHARE\_KEY

From this analysis, it is possible to observe that the client does not perform complex calculations and does not use too many resources due to the fact that the initial key is shared and the client performs only a decryption, which is not a time-consuming operation. The proposal supports the concept of the light-weighted client, i.e. the most consuming operations are ABE and AES decryption.

**Online mode**

**In the online mode the client acts differently. There is a support from server and the user credentials can be checked once the validity period of his keys expires. So the communication with server happens once the client discovers the time is expired.**

1. **Client checks the time counter and the keys validity.**
2. **If time is expired the client deletes the PUBLIC\_SHARE\_KEY of the user.**
3. **The user does not have access to the files anymore.**
4. **The client asks the user to connect to the server (or does it in the background)**
5. **User enter s the PASS and PIN**
6. **The client uses the safe protocol for key exchange (for the moment, J-PAKE) to establish the new KEY\_SET and KEY\_SET\_KEY**
7. **After receiving the KEY\_SET and TIME+DEV\_PASS the client sets up the time counter.**
8. **From this moment the communication with the server is not necessary. User just needs to enter his PIN in order to decrypt the protected files.**

**It is important to keep the track of time both on server and on client in parallel in order to avoid the malicious user behavior. In such concept of a synchronized communication with server, when the client does not need to be connected all the time (statistics can also be send to server with each session) the advantage is that the load on the device as network traffic (which is often a constraint) is set to minimum.**

The client has a token of the previous session DEV\_PASS, which is used for the server communication:

This value serves as a proof the device is the one with which server communicated.

To prevent the direct attack on the KEY\_SET, e the KEY\_SET\_KEY is received (for symmetric encryption) from the communication with the server. This value is split into PASS, PIN (permanent), TIME (temporary), DEV\_PASS (calculated from previous values). The value splitting is performed on a server. The task of a client is to receive the TIME and DEV\_PASS from server.

In order to perform the splitting, the server has posses the knowledge of the user PIN.

The client uses some balanced key exchange to communicate with server:

J-PAKE or SIS-EKE.

The client sends the token to server: DEV\_PASS +PASS+PIN – to prove his identity and identity of the user to server. The server establishes the safe session with the help of new SESSION\_KEY = KEY\_SET\_KEY, verifies the communication and sends the key set to the client.

Online workflow diagram:

*EKE description*

*This proposal uses a light-weighted EKE for server communication like J\_PAKE or SIS-based EKE. SIS is a public key encryption that avoids generating the big number of primes and thus can be used for the secure key exchange.*

There are generally 3 phases of any EKE:

1. Key establishment (authentication of the user PASS and PIN and device DEV\_PASS)
2. Key verification
3. Sending the data (everytime the TIME should be checked and the data should be send within the protected channel)

How to secure the data when the user leaves the group.

Reduce power consumption

Improve reliability

Enhance processing

Shamsi, J., Khojaye, M.A., Qasmi, M.A.: Data-intensive

cloud computing: requirements, expectations, challenges,

and solutions. Journal of Grid Computing **11**, 281–310

(2013)

2. Khan, A.R., Othman, M., Madani, S.A., Khan, S.U.: A

survey of mobile cloud computing application models.

Communications Surveys & Tutorials, IEEE **16**, 393–413

(2014)

3. Kumar, K., Lu, Y.H.: Cloud computing for mobile users:

Can offloading computation save energy? Computer **43**,

51–56 (2010)

4. Lorido-Botran, T., Miguel-Alonso, J., Lozano, J.A.: A

review of auto-scaling techniques for elastic applications in

cloud environments. Journal of Grid Computing **12**, 559–

592 (2014)

5. Khan, A.N., Mat Kiah, M., Khan, S.U., Madani, S.A.:

Towards secure mobile cloud computing: a survey. Futur.

Gener. Comput. Syst. **29**, 1278–1299 (2013)

[1] <https://cloudsecurityalliance.org/>

[2] <http://www.ciphercloud.com/blog/cloud-data-security-and-eu-data-privacy-rules-compliance-with-encryption-and-tokenization/>

[3] <http://www.gartner.com/technology/topics/cloud-computing.jsp>

[4] <https://www.skyhighnetworks.com/cloud-university/what-is-cloud-access-security-broker/>

[5] <http://www.gartner.com/technology/reprints.do?id=1-2RUEH70&ct=151110&st=sb>

[6] <http://www.ciphercloud.com/blog/cloud-data-encryption-easy/>

[7] <https://securityintelligence.com/how-to-protect-mobile-apps-essentials/>

[8] Abdul Nasir Khan, M. L. Mat Kiah, Mazhar Ali, Shahaboddin Shamshirband, Atta ur Rehman Khan. A Cloud-Manager-Based Re-Encryption Scheme for Mobile Users in Cloud Environment: a Hybrid Approach

[9] Abdul Nasir Khan, M. L. Mat Kiah, Mazhar Ali, Sajjad A. Madani, Atta ur Rehman Khan, Shahaboddin Shamshirband. BSS: block-based sharing scheme for secure data storage services in mobile cloud environment

[10] TODO: reference to SSS papers

[11] TODO reference to ABE papers

[12] Zhao, G., Rong, C., Li, J., Zhang, F., Tang, Y.: Trusted data sharing over untrusted cloud storage providers, presented at the IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom’10), Washington, DC, USA (2010).

[13] Yang, J., Wang, H., Wang, J., Tan, C., Yu, D.: Provable data possession of resource-constrained mobile devices in cloud computing. Journal of Networks **6**, 1033–1040 (2011).

[14] Itani, W., Kayssi, A., Chehab, A.: Energy-efficient incremental integrity for securing storage in mobile cloud computing, presented at the International Conference on Energy Aware Computing (ICEAC ’10) Cairo, Egypt (2010).

[15] Ren, W., Yu, L., Gao, R., Xiong, F.: Lightweight and compromise resilient storage outsourcing with distributed secure accessibility in mobile cloud computing. Tsinghua Science & Technology **16**, 520–528 (2011).

[16] SAP Mobile Platform Secure Mobile with Mocana: webinar whitepaper