Noname manuscript No.
(will be inserted by the editor)

Offline Mode for Corporate Mobile Client Security Architecture

Tatiana Galibus · Thiago P. B. Vieira · Edison P. de Freitas · Robson O. Albuquerque · João Paulo C. L da Costa · Rafael T. de Souza Júnior · V. Krasnoproshin1 · Anton Zaleski · H.E.R.M. Vissia · Giovanni del Galdo

Received: date / Accepted: date

Abstract Preventing data leakage on the mobile device is a crucial security problem. Therefore, additional control and protection measures should be taken for the confidential data on the mobile devices that leave the boundaries of the organization. In this scenario, such devices can be used on the offline mode for the convenience and traffic reduce or even for a particular characteristic of the app or of the business. This paper presents a novel approach to the security of the corporate mobile devices, in particular the offline mode. The important part of the presented approach is the conceptualization and the definition of the core methodology to solve the problem of offline mobile security, i.e. the protection of the confidential data in use when the mobile client is not connected to the corporate cloud. The protection of the sensitive data is provided by the combination of the cryptographic methods, such as AES file encryption, ABE authorization based both on user and share attributes, user key secret sharing (SS) based protection between the device and the user as well as MOS-based analytics methods to prevent the malicious user behavior. The proposed security architecture supports the basic mobile device protection principles: minimized traffic load and reduced communication with the cloud; usage of the light-weighted operations and an optimized combination of the security methods.

Keywords mobile security · offline mode ABE · Model Order Selection (MOS) · user behavior analysis · secret sharing · protected cloud

Tatiana Galibus
Belarusian State University, Minsk, 4, Nezavisimosti, 220030, Belarus

Tatiana Galibus · Thiago P. B. Vieira · Robson O. Albuquerque · João Paulo C. L da Costa · Rafael T. de Souza Júnior
University of Brasilia, UnB, FT, ENE, CP: 4386, 70910-900, Brasília, Brazil

Edison P. de Freitas
Federal University of Rio Grande do Sul, UFRGS, INF, CP: 15064, 91501-970, Porto Alegre, Brazil

João Paulo C. L da Costa · Anton Zaleski · H.E.R.M. Vissia · Giovanni del Galdo
Institute for Information Technology, Ilmenau University of Technology, Ilmenau, Germany

João Paulo C. L da Costa · Giovanni del Galdo
Fraunhofer Institute for Integrated Circuits IIS, Erlangen, Germany

????
Byelex Multimedia Products BV Argon 1, 4751 XC Oud Gastel, The Netherlands

1 Introduction

Cloud computing is a new rapidly evolving paradigm in the world of distributed networking and computation. The basic features of the cloud environment is providing the elastic, on-demand and secure services for the end-users. While the first two requirements are rather well conceptualized and supported by the majority of the cloud platforms in use, security is a serious concern of the cloud providers and governmental organizations as well as academia and research community [1, 16, 12]. Although for the small and medium-sized enterprises (SME) the cloud environment is often the most cost-effective and easily scalable solution, the security and privacy of the sensitive data in cloud platforms are also not fully conceptualized leading to obscure and incomplete security paradigms and solutions.

A common practice to provide a cloud solution with stable security is to use a specific type of cloud service, such as Cloud Access Security Broker (CASB) or Cloud

Access Control (CAC). These services are specifically designed to bring security at a single access point and provide the coordination of the most important security measures. It is estimated by Gartner [29] that such systems will be used by 85% of companies by 2020. Obviously, the reason for this is that the organization of 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 CASB are: discovery of cloud services, encryption along with tokenization for better search properties, access control, Data Loss Prevention (DLP) services, authentication, and auditing and alerting services [25]. The protection of the confidential data, according to the standards of CASB deployment should be provided elsewhere, i.e. in transit and at the end-user, i.e. end-client, side [5].

Additional security issues and requirements have to be considered when mobile devices are actively used in corporate cloud environment [34]. Today more and more organizations and enterprises are functioning in the Bring-Your-Own-Device (BYOD) paradigm. The uncontrolled usage of the mobile devices represents a serious risk to the development of secure SME cloud platforms being the bottleneck of the corporate information security system (ISS). While the enterprise cloud infrastructure based on the web interface can be protected by powerful third-party services, such as CASB and CAC, the corporate mobile app is usually lightweighted and generally less protected. The protection scheme used on a mobile device should be both computationally secure as well as resource-constrained due to battery power limitations [23]. Therefore, encrypting files and generating keys on a mobile device is not considered a good solution. On the other hand, the protection schemes with good computational qualities lack the security analysis in many cases [22]. The common practice is the shadow user activities monitoring [34]. However, the mobile device usage stays unprotected in all the proposed scenarios while in offline mode.

Suppose that a SME uses CASB in order to protect data at rest, i.e. while stored on the server-side, in transit, communicating with server, and, in use, while the client is connected to the network. In this case, when the mobile client goes offline with the sensitive corporate data on board all powerful cloud-based tools cannot help and the mobile app has to secure itself with its own limited resources. Moreover, due to the resources constraint, there is a crucial difference in strategy of online and offline mode protection. For example, offline mode does not allow performing the extensive computation and encryption on the mobile device. Observing the above described landscape, this paper outlines the concept of the offline mobile client security. This paper proposes a novel approach based on powerful cryptographic preventive methods, such as secret sharing [8] and ABE encryption [13]. Moreover, our proposed approach proposal includes the usage of the user behavior analysis based on Model Order Selection (MOS) [30], in order to highlight possible threats, and to reduce the risks and the harm of the most common threats, which are the expired user misusing password and the intruder attack. The key expiration period is safely incorporated into the proposed system solution in order to enhance security. Additionally, the behavioral analysis can indicate well known malicious behaviors, their variations, as well as novel attacks, which present low or high variance in comparison to legitimate user behaviors. The main target of our proposed solution 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 related to the offline protection in the BYOD world. Section 3 outlines the basic processes and the module security infrastructure of the proposed solution as well as the outline of online mode security. Section 4 presents the detailed scheme of the proposed solution to the problem of offline mobile client security. Section 5 provides the schemes of the main security blocks including encryption and key usage hierarchies as well as explanations of the core security methods: SSS, ABE and MOS. Section 6 discusses the security proofs including the outline of adversary model, common threat scenarios and the proposed analytics to discover abnormal usage. Still in Section 6, complexity analysis and practical implementation results are included. Section 7 concludes the paper.

2 Related works

The increasing usage of BYOD demands more sophisticated data protection services compared to ordinary computing environment. A common practice is to provide additional contextual methods apart from authentication, DLP services, and encryption, which can be at rest, in transit and in use [34, 23, 24, 21]. The contextual methods increase the security of the app at a maximum level with minimum resource requirements. The most commonly used are: Using geolocation of the device to trace its usage; Setting up the expiration period of an app; Secured transfer politics between apps; Restricting access to the corporate app; Setting up the expiration period of app pass pin; Setting up the counter of failed tries; Restricted or prohibited offline access; Logging and auditing. Preventing data leakage on the mobile device is a crucial security problem. Therefore,

administrators should take additional control and protection measures for the confidential data on the mobile devices that leave the boundaries of the organization. Generally, the most sensitive and confidential data should not be permitted to be transferred to the mobile device. However, what if the SME need to allow their employees to work on such devices and even use them on 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 of this problem, there are several surveys, whose common point is the mobile cloud computing as a rapidly developing paradigm that poses many security and complexity problems [34, 23, 24, 21]. An analysis of the new models of mobile cloud computing and new ways of using data storage services is presented in [23, 21]. Commonly, all the models and protection schemes concentrate on the encryption properties and either perform the computations on their own [35, 33] or use the cloud provider to off-load the cryptographic operations [18, 28]. Obviously, it is natural the mobile device cannot handle all operations securely without the assistance of a cloud provider, due to resources constraint and battery limitation. The necessity to use schemes that function without putting load on a provider arises when it is desired to make the device less dependent on the cloud provider, i.e. corporate app continues to provide the secure access to the sensitive data without connection to the network. As discussed in [23], all the currently known schemes of encryption, performing the computation, either use a cloud provider [35], a third party trusted agent [33] or a combination of both [18]. In some cases, they concentrate on computational complexity without taking care of user privacy and security [28]. Therefore, according to [23, 30, 21], the state-of-the-art mobile cloud security models do not consider the problem of the offline security mode. In many cases the industrial providers of the mobile security API avoid the problem by completely forbidding the offline access to the protected app, i.e. SAP Mocona, which operates as a secure app wrapping layer [32]. Due to various constraints such as traffic load, travelling and ease of access, the SME business procedure may require such access. To the best of our knowledge, the offline mode security problem has not yet been deployed, neither in academia nor in the industry [34, 23, 24, 21]. Therefore, the main concern of this proposal is the protection of the device and app in offline mode when the functions of data protection cannot be offloaded to a cloud or a trusted party.

3 The client security architecture

The approach proposed in this work describes and implements the complete lifecycle of the mobile app security infrastructure. The security processes depend on the key expiry period, and are used to access the protected storage. Once the user keys expire, the user is requested to enter his valid credentials, i.e. PIN and password. The client app then sends the credentials to the server for verification. Once the new set of access keys is received, the user can access the protected files in the offline mode, without the access to the server. This means that no further communication with the server is needed until the key expires. The core set of functions and protocols can be divided into three sets of operations as shown in Figure 1.

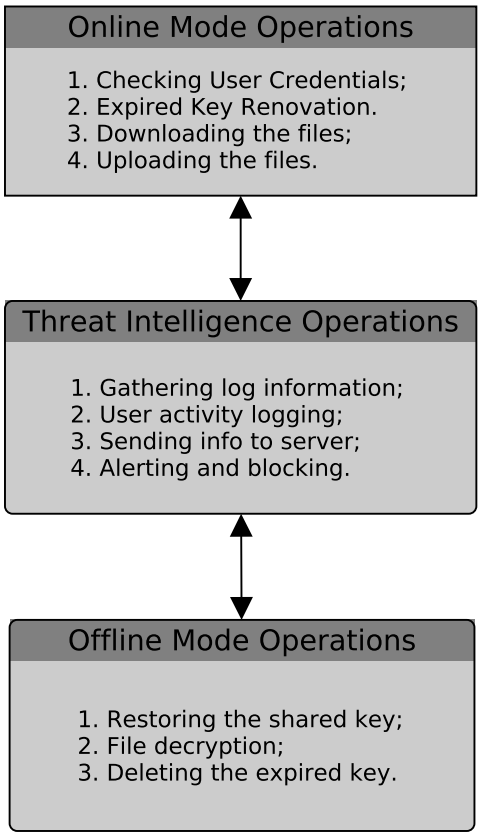

Fig. 1 The core set of functions and protocols of the mobile app security infrastructure

These operations function based on the mobile client architecture, which is depicted in Figure 2. This architecture consists of the modules of cryptographic functions, threat intelligence infrastructure, communication with server and storage.

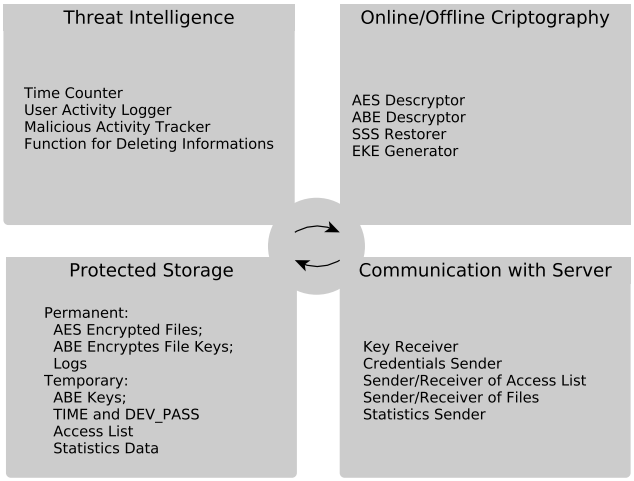

Fig. 2 The Mobile Client Architecture

The cryptographic functions include the decryption (for AES and ABE) key restoration for SSS and EKE token generation procedure. While the AES-ABE hybrid encryption scheme functions both in the online and offline modes, the SSS is used only in the offline mode and the EKE generation and executions is used in the online mode. These operations are described in Subsections 5.1-5.3 and are part of the Figure 1, related to online and offline mode operations.

The threat intelligence infrastructure takes into account simple actors such as the time counter for the key expiry period, the counter of unsuccessful tries in order to protect from brute force attacks, and more elaborate MOS-inspired statistics analyzer. Functions such as alerting and deleting the expired key belong to this block as well. These functions are described in Subsection 5.4.

The communication with server includes the separate sender and receiver to check the user credentials and receiver new keys which acts based on the EKE protocol. The remaining data like access list and files can be sent via unprotected channels. Subsection 3.1 briefly discusses the communication with the server, but it is also an intention to make it a subject of our next work.

The storage is divided in two parts – temporary and permanent. While the files and file keys are stored permanently in order to reduce unnecessary traffic and the resource usage on client, the ABE keys and the key storage protection data are temporary. The unprotected files stored in the mobile client app memory are not the subject of the security architecture.

3.1 Online mode

This subsection briefly outlines the main processes and functions in the online mode. The connection to the network allows verifying the user credentials with the help of the protected cloud server. The communication with the cloud is secured by means of the encrypted key exchange protocol [14, 15] and serves to receive the new set of keys, files and send the collected log information. The client app operations in the online mode are as represented in Figure 3.3.

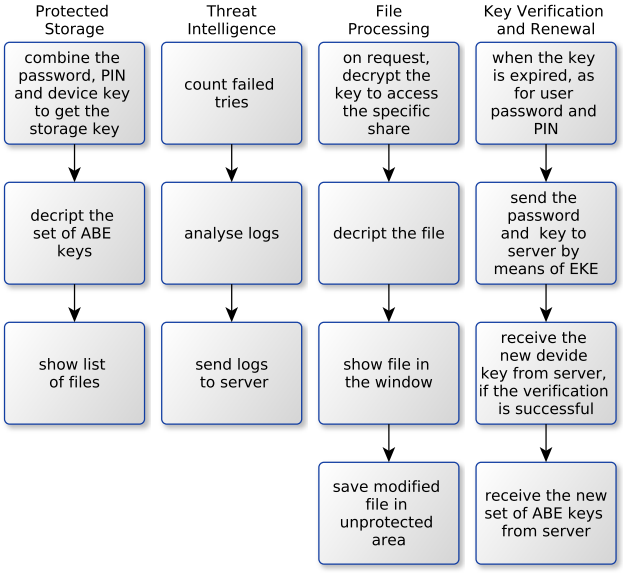

Fig. 3 Online Mode Operations

The communication with server is necessary once the key is expired. This allows optimizing the load on the network once the traffic is reduced to the discrete sessions. In such environment it is important to keep track of the time synchronization both on server and on client in parallel in order to avoid malicious user behavior messing up with time attacks.

As depicted in Figure 4, the operation of the online mode starts from the client app sending a token to server by means of EKE: DEV_PASS + TIME + PASS + PIN in order to validate his identity and the identity of the user to the server. This value serves as a proof that the device and the user are the ones with which server communicated previously. After performing the verification the server sends the new calculated DEV_PASS and TIME to the mobile app by means of established EKE session. Afterwards the server sends the new encrypted key set and encrypted files to the client app in the clear.

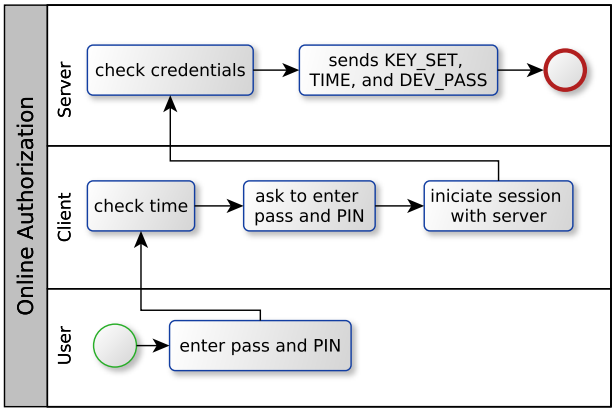

Fig. 4 Online Authorization Workflow

This model proposes to use light-weighted EKE for the fast communication with server communication like J-PAKE [14, 15] or SIS-based EKE [4]. SIS is a public key encryption that avoids generating the big primes, thus it can be used for the secure key exchange [4, 3]. EKE is performed as soon as the key is expired and the client asks the user to synchronize with the server. Note that the EKE is not required to send the encrypted files or statistics. The current implementation of mobile app uses J-PAKE protocol. We refer to [14, 15, 31] for a complete description and analysis of the J-PAKE protocol.

4 Offline mode: our proposed model and solution

This section proposes an open model of mechanisms for the mobile device protection in which the security is supported both in online and offline modes. Currently and to the best of our knowledge, the systems of mobile device protection follow a 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 herein proposed 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. 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;
3. Behavioral analysis of user's operations on mobile client, which can indicate anomalous or automated activities performed by attackers.

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 protected SME documents. In this case, the server can neither monitor the user activity nor provide the protection methods. The security should be performed at the mobile client. Additionally, the maximum protection should be provided at the minimum resource cost.

In the online mode the mobile device uses the secure communication with the server in order to verify the validity of user's credentials. On the contrary, the offline protection model should be approached independently. Thus, the proposal is that the authentication/authorization mechanisms in the offline mode should utilize the derived 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 on the device;
3. The proof should 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 offline dictionary attacks;
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.

Such proof cannot be stored on the client device, as it is not possible to guarantee its protection in the offline mode. Therefore, the most effective way to secure the proof on the offline mode without performing complex computation is to share this proof between the client and the user in a protected manner. In this case, there is no need to store the function of the user password. 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 proposed offline mode architecture is represented in Figure 5.

The infrastructure in Figure 5 includes:

1. **The protected storage**: the storage is protected with the shared user key and contains the ABE keys giving access to the file keys which allow decrypting the stored files. The outline of the hybrid encryption scheme is presented in the Section 5.
2. **Threat Intelligence Manager (TIM)**: most attacks incur into significant variation on the legitimate behavior of information systems, or they adopt

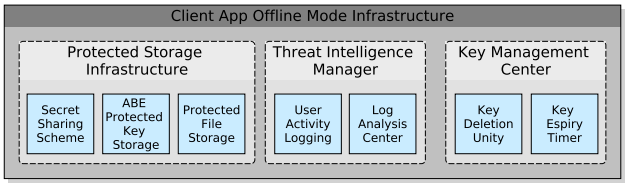

Fig. 5 Online Authorization Workflow

well-known patterns that can be easily detected by monitoring the system in the case of the offline mode. Signal processing techniques have been successfully applied to anomaly detection [26, 17] and have become a solution to a problem of improving detection accuracy, adaptability and computational cost for application on resource-constrained scenarios. Therefore, signal processing can be applied in offline mobile client security, for evaluating anomalies on user's behavior according to the scenarios in Section 6. Moreover, Model Order Selection (MOS), which is an effective signal processing technique to separate noise components from the principal components, can be applied into anomaly and attack detection [30], to identify and separate malicious behaviors from the legitimate ones.

3. **Key Management center**: it includes the functions for maintaining the key expiry period and deleting the expired keys.

4.1 Offline mode workflow

The user performs the following operations in the offline mode:

1. Enters the PASS and the 4-digit PIN;
2. Views the list of authorized files kept on the mobile storage;
3. Open the protected encrypted files;
4. Modify the protected files and save them in an unprotected storage.

The client app performs all the cryptographic calculations in the shadow. These calculations include the key storage, key restoring, decryption and showing the decrypted files in the client area. Note that the client app does not check the password or PIN validity as it does not hold the verification proof for the above. Also, the app keeps track of the user activities and the key expiry. The complete list of the app activities is presented in the Figure 6.

To finalize the description of the offline mode, Figure 7 shows the complete workflow of the proposed mobile application in the offline-mode.

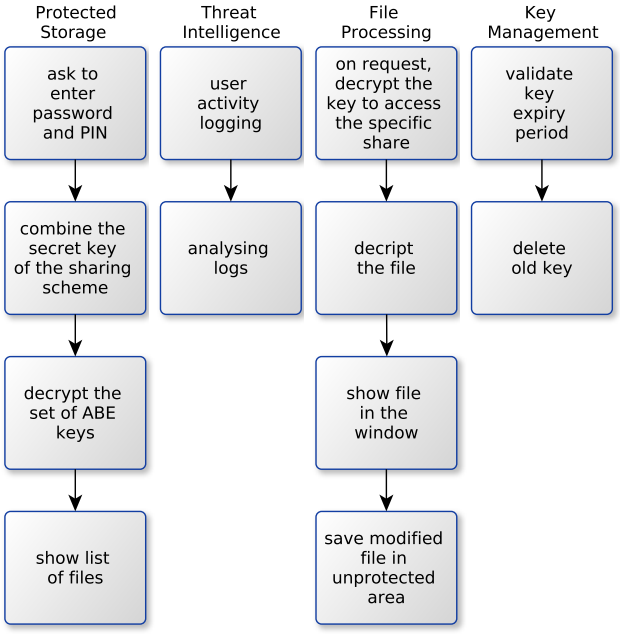

Fig. 6 Offline Mode Operations

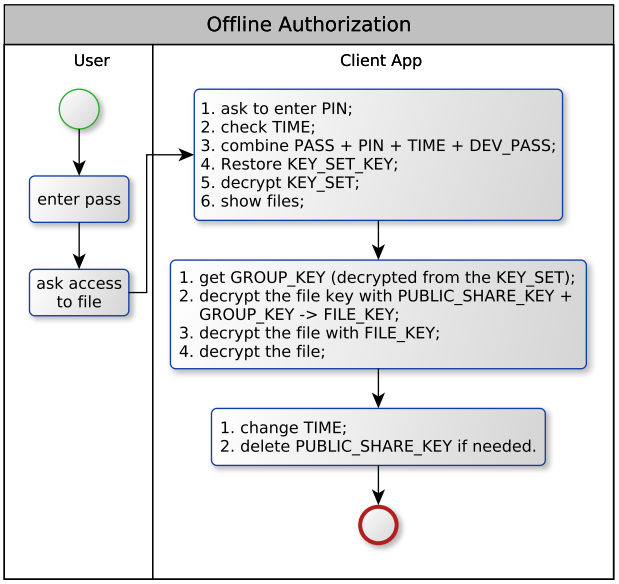

Fig. 7 Offline mode authorization workflow

The core cryptographic module of the implemented solution is based on the combination of AES-ABE-SSS methods, described in the Section 5. The key feature of the offline security is that the client app does not actually store any part of the user password to be verified. The client app 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. Additionally, the password entering is tracked and too many tries in a short time are considered a threat.

4.2 Offline Behavioral Analysis

In the proposed client security architecture, the Threat Intelligence Manager (TIM) is responsible for receiving logged user operations, feature extraction, data modeling and malicious behavior analysis in order to identify possible threats, in offline mode. Figure 8 depicts the Threat Intelligence Manager for offline behavioral analysis.

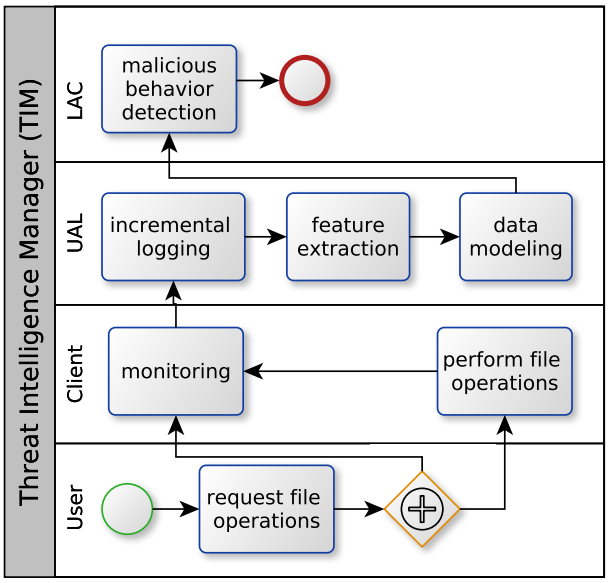

Fig. 8 The Threat Intelligence Manager Workflow

As depicted in Figure 8, users request operations are logged so that the main features can be monitored in the client app. The user behavior, trying or effectively executing operations, shall be incrementally captured and logged, making possible to monitor the main features that can reveal malicious behaviors, as well as to identify unexpected behaviors that can reveal possible threats. Therefore, the user operations are monitored by the client app, which sends the information of the captured operations to the User Activity Logging (UAL).

The UAL is responsible for the incremental logging, feature extraction and data modeling for malicious behavior detection, through the Log Analysis Center (LAC). The logged information shall be decomposed into selected features and modeled as matrices, composed of the number of occurrences of the selected features by its location and by the occurrence time. The resultant data is submited to the LAC, for anomaly detection.

The LAC performs the behavioral analysis through eigenvalue analysis and MOS schemes, which identify anomalies on sparse, subtle or abrupt number of user operations. The malicious behavior detection is detail described in 5.4.

5 The algorithms, key usage and data protection methods

In the proposed approach, the kernel encryption scheme in the mobile device is a combination of several methods of security. The files are encrypted with 128/256-bit AES, while the permanent file keys are encrypted with the attribute-based encryption. The set of expiring ABE keys corresponding to the set of files accessible by user in encrypted with a single expiring AES key (KEY_SET_KEY). This key is expiring and is splited by the server into 4 parts (2 are stored on the device and 2 belong to the user) by the method of secure secret sharing. The encryption workflow is outlined in the Figure 9.

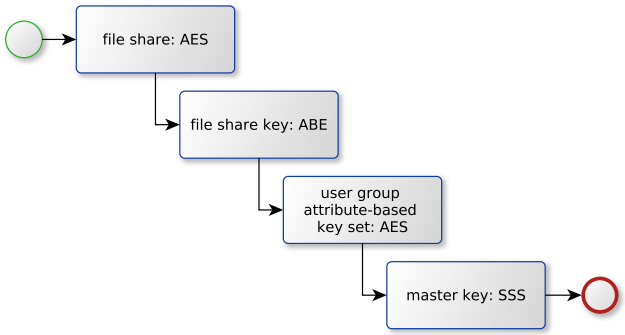

Fig. 9 Encryption workflow

The key hierarchy can be explained by Figure 10 where the key represented in Figure 9 serves for decrypting the key for each level represented in Figure 10.

This proposed encryption scheme is called as a hybrid encryption since it uses a combination of different encryption methods in order to support all requirements to the encryption scheme used in the proposed infrastructure. Then it has:

1. AES provides a fast encryption of the data files;
2. ABE permits to support the authorization policy on the encryption level;
3. SSS allows protecting the key storage by avoiding the necessity to store the key proof.

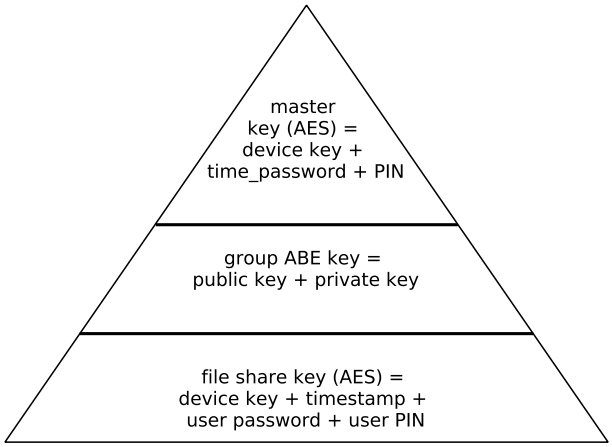

Fig. 10 Encryption key hierarchy

In Subsections 5.1-5.3, a detailed description of the building blocks of the hybrid encryption is provided, and the proposed malicious behavior detection scheme is detail described in 5.4.

5.1 AES file encryption

The protected files are encrypted on the server side with the secure 128/256 bit AES encryption, which is currently an industrial standard. Nevertheless, domain administrator can choose to encrypt the file storage with Blowfish or Serpent which also provide a high level of safety [19].

The symmetric encryption on a server side is performed in two steps. First there is a single AES key for preserving the server storage. Second, when the encrypted file is sent from the server to the mobile app or desktop device another unique AES key is generated. This key serves for the encryption on the client side. In other words, along with the encrypted file the user gets his own unique AES key for decrypting the files received. This key is protected by attribute-based public key encryption (ABE).

The scheme of the server-side AES encryption is represented in Figure 11.

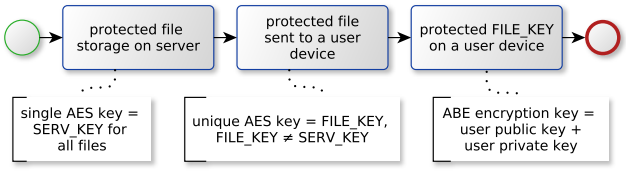

Fig. 11 Server-side encryption

Usually, the AES encryption key is used as a session key. In our setting, it is not desirable to re-encrypt the files stored in the device unless there is a certain specific condition (for example, the user leaves the domain or a specific group in the domain (the group of users in the domain contains the users with the equal access rights, i.e. the same set of accessible file shares) and the file should not be accessible for this user anymore). This is why AES key in the presented notation is permanent and defined as FILE_KEY. The randomly generated FILE_KEY is unique for each file or set of files stored on a client.

5.2 ABE encryption to protect the FILE_KEY

With each user session, the permanent FILE_KEY (unique AES key) is re-encrypted. The set of FILE_KEYs is protected with the corresponding ABE keys. The unique ABE model that we propose supports the attribute policy based on user groups and on file shares, i.e. the key attributes correspond both to the groups and the file shares.

The model supports the simple selective ABE scheme [9, 7]. The selective scheme for attribute-based encryption is as follows: if at least one attribute in the set $\{t_i\}_M$ is equal to the attribute in the settiM, the corresponding user U can decrypt the text M. In other words, as soon as the user and share have one attribute in common – the user can get access to the share. The components of the ABE encryption are:

1. **Master-key (MK)** which is kept safely on server and accessible only for the domain administrator and is given by $MK = (t_1, t_2, \cdots, t_n, y)$, where the values t_i 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.
2. **Public key (PK)** depends on the master key values and is kept in the clear allowing users to access the information: $PK = (g^{t_1}, g^{t_2"}, \cdots, g^{t_n}, e(g,g)^y)$, Here $e(g,g)$ is the bilinear pairing function corresponding to an elliptic curve.
3. Secret user **KEY_SET** depends on his attribute set. Here each D_i (**GROUP_KEY**) serves for decryption of the data of a single group of users, for example, related to some project: $\{t_i\}_U \rightarrow D = \{D_i = g^{\frac{yw}{t_i}}\}$.
4. Encrypted text M, in our context, $M = FILE\_KEY$, or the permanent AES symmetric key, which allows to avoid the file re-encryption. Encryption procedure is multiplication. The set of the public keys

E_i (PUBLIC_SHARE_KEY) corresponding to the set of groups able to access the text is kept along with the encrypted text E: $E = Me(g,g)^{ys}, \{E_i = g^{\frac{t_i s}{w}}\}, \forall i \in \{t_i\}_M$.

5. Decryption is division: $M = \frac{E}{Y^s}$.

In order to perform this operation the user needs the pair of private key D_i and public key E_i corresponding to the attribute t_i:

$$Y^s = e(g,g)^{ys} = e(E_i, D_i) = e(g^{\frac{yw}{t_i}}, g^{\frac{t_i s}{w}}) = e(g,g)^{ys}. \quad (1)$$

The result of decryption is the FILE_KEY - the symmetric AES key that permits to decrypt the contents of protected file.

5.3 Secret sharing scheme to protect the key storage

The attribute-based private keys D_i should be protected while being stored in the device memory. Therefore, server encrypts the set of Di with a single AES key before sending it to the user device. This AES key (master key) is denoted by the value KEY_SET_KEY in our notation. KEY_SET_KEY is a secret value and it is split by the secure method of polynomial modular secret sharing [11, 10] into the set of 4 shares: KEY_SET_KEY = PASS + PIN + TIME + DEV_PASS.

Since the underlying sharing scheme is perfect [10], hence the adversary cannot get any information of the KEY_SET_KEY unless he possessing all 4 key parts. Here the values PASS and PIN are predefined similar to the construction in [11].

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 and device. The user share $s_1(x)$ is computed based on the PIN and the PASS entered by the user. Additionally, the current time value TIME is used in the calculation of the share. Let $s(x) = d$ and $s_1(x) = f(PIN + PASS + TIME)$, where f is a one-way function that transforms the data into the string of the desired length:

$$\begin{cases} S = s \bmod p_0 \\ S = s_1 \bmod p_1 \end{cases} \quad (2)$$

According to the CRT:

$$S \equiv s_1 p_0 {p_0}^{-1}{}_{p1} + s_1 p_1 {p_1}^{-1}{}_{p0} + \bmod p_0 p_1 \quad (3)$$

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

5.4 MOS for Threat Intelligence

In the context of anomaly-based schemes for attack detection, the proposed behavioral analysis approach applies signal processing techniques, such as Principal Component Analysis and Model Order Selection schemes [30], for automatic identification of attacks or malicious behaviors.

Model Order Selection is an effective signal processing technique for several applications, allowing separating the only noise components from the principal components applying a rank reduction of the data. Applying MOS to the analysis of user operations can be effective in order to reveal the occurrence of malicious behavior during an offline session.

MOS for threat intelligence requires that the target features, which can be user operations, are modeled as a matrix composed by the number of occurrences by location and time, and grouped into Q time frames. Therefore, the framework considers the time variations of the matrix $\mathbf{X}^{(q)} \in \mathbb{R}^{M \times N}$, with $q = 1, \ldots, Q$, in order to detect the occurrence of malicious behaviors. For example, one element of $\mathbf{X}^{(q)}$ can represent the number of file readings on folder m during the minute n, from file operations logged by the client app.

The MOS schemes can rely on sample covariance of zero mean variables (called as zero mean covariance for the sake of simplicity) and sample covariance of zero mean and unitary standard deviation (called as zero mean and standardized covariance for the sake of simplicity) variables, where the former is useful to identify abnormalities caused by large amounts of operations during a period, while the latter is applied to identify anomalies on sparse or subtle number of file operations.

Classical approaches to model order selection require the computation of the sample covariance matrix $\hat{\mathbf{R}}_{yy}^{(q)}$ and of its eigenvalues, obtained from the measurement matrix $\mathbf{X}$ of the zero mean samples given by

$$\mathbf{y}_m^{(q)} = \mathbf{x}_m^{(q)} - \bar{\mathbf{x}}_m^{(q)}. \quad (4)$$

The set of obtained vectors $\mathbf{y}_m^{(q)}$ composes the zero mean matrix $\mathbf{Y}^{(q)}$, then the zero mean covariance matrix $\hat{\mathbf{R}}_{yy}^{(q)}$ can be calculated as follows

$$\hat{\mathbf{R}}_{yy}^{(q)} = \frac{1}{N} \mathbf{Y}^{(q)} \mathbf{Y}^{(q)^{\mathrm{T}}}. \quad (5)$$

For MOS based on sample covariance of zero mean and unitary standard deviation, in order to identify anomalies with sparse or subtle behavior, it is required, for each variable, to make the standard deviation unitary as follows

$$\mathbf{z}_m^{(q)} = \frac{\mathbf{x}_m^{(q)} - \bar{\mathbf{x}}_m^{(q)}}{\sigma_m^{(q)}}. \tag{6}$$

The set of vectors $\mathbf{z}_m^{(q)}$ composes the matrix $\mathbf{Z}^{(q)}$, then the zero mean and standardized covariance matrix $\hat{\mathbf{R}}_{zz}^{(q)}$ can be calculated via

$$\hat{\mathbf{R}}_{zz}^{(q)} = \frac{1}{N}\mathbf{Z}^{(q)}\mathbf{Z}^{(q)^{\mathrm{T}}}. \tag{7}$$

Once the $\hat{\mathbf{R}}_{yy}$ or $\hat{\mathbf{R}}_{zz}$ have been obtained for MOS in order to anomaly detection, for the sake of simplicity, we refer to $\hat{\mathbf{R}}_{yy}$ or $\hat{\mathbf{R}}_{zz}$ as a matrix $\mathbf{C}$. Therefore, the next step of the algorithm is the eigenvalue decomposition (EVD), calculated according to $\mathbf{C}^{(q)} = \mathbf{V}^{(q)}\mathbf{\Lambda}^{(q)}\mathbf{V}^{(q)^{\mathrm{T}}}$,, in order to obtain the vector of eigenvalues e, as following:

$$\mathbf{e}^{(q)} = \mathrm{diag}(\mathbf{\Lambda}^{(\mathrm{q})}), \tag{8}$$

The eigenvalues should be sorted in descending order, as defined by $\lambda_1^{(q)} > \lambda_2^{(q)} > \lambda_3^{(q)} > \cdots > \lambda_m^{(q)}$, to make possible the selection of the first eigenvalue in the obtained sequence, represented by $\lambda_1^{(q)}$, which is the largest eigenvalue of the data evaluated for attack detection.

The process of obtaining the $\mathbf{X}^{(q)} \in \mathbb{R}^{M\times N}$ and the matrix $\mathbf{C}^{(q)}$, finding the largest eigenvalue for each q-th time frame, should be repeated until $q = Q$, in order to obtain the largest eigenvalue of all time frames, as presented by

$$\mathbf{E} = \begin{bmatrix} \lambda_1^{(1)} & \lambda_1^{(2)} & \lambda_1^{(3)} & \cdots & \lambda_1^{(Q)} \\ \lambda_2^{(1)} & \lambda_2^{(2)} & \lambda_2^{(3)} & \cdots & \lambda_2^{(Q)} \\ \lambda_3^{(1)} & \lambda_3^{(2)} & \lambda_3^{(3)} & \cdots & \lambda_3^{(Q)} \\ \vdots & \vdots & \ddots & \vdots & \\ \lambda_m^{(1)} & \lambda_m^{(2)} & \lambda_m^{(3)} & \cdots & \lambda_m^{(Q)} \end{bmatrix}. \tag{9}$$

Since $\lambda_1^{(q)} > \lambda_2^{(q)} > \lambda_3^{(q)} > \cdots > \lambda_{m-1}^{(q)} > \lambda_m^{(q)}$, then the first line of the matrix $\mathbf{E}$ contains the largest eigenvalues of each q-th time frame, which is the expected input for MOS schemes and can be expressed as

$$\boldsymbol{e}_{\max} = \boldsymbol{E}\{:, 1\} = [\lambda_1^{(1)}, \lambda_1^{(2)} \cdots \lambda_1^{(Q)}] \tag{10}$$

Once obtained the largest eigenvalues of each q-th time frame, it is possible to apply a selected MOS scheme to estimate the model order $\hat{d}$, which is the estimated number of time frames with malicious behavior. Therefore, $\boldsymbol{e}_{\max}$ is used as input parameter for MOS schemes, according to the equation

$$\hat{d} = \mathrm{MOS}(\boldsymbol{e}_{\max}) \tag{11}$$

Note that some MOS schemes may also require the amount of time that compose a time frame, such as $\hat{d} = \mathrm{MOS}(\boldsymbol{e}_{\max}, M)$. For more information about MOS schemes, interested readers are referred to [6].

6 Results and analysis

This section provides the detailed analysis of the results the proposed approach regarding security and complexity analysis.

6.1 Security analysis

The security analysis of the proposed model was performed both from the point of implemented cryptographic mechanisms and from the user behavioral analysis. Two common attack scenarios were analyzed. First, the malicious outsider trying to infect or steal the important files. Second, the malicious expired user trying to steal the important files.

6.1.1 Adversary model

The proposal is to use a restricted adversary model. In other words, the actions of the adversary are predicted within the outlines model and scenarios. For example, it is not possible to guarantee the security of the decrypted document once the user copies its content to another file. In the analyzed scenarios, it is assumed that the adversary can:

1. Steal the user keys one-by-one while the app performs decryption, if he is outsider;
2. Use expired key to decrypt the files, if he is insider;
3. Does not have possibility to perform offline dictionary attack and cryptanalysis, i.e. does not have the possibility to verify user credentials apart from entering them in the client app;
4. Steal the encrypted files but not the decrypted ones.

6.1.2 Security analysis

Offline Mode

In the offline mode the client does not store the user password (see Section 4.1 for details), i.e. no information about the password leaks, and therefore there is no possibility for the malicious user to check if the password he is trying to enter is correct or not. The only possible scenario for the information leakage in the case of the malicious outsider is if:

(a) The hacker steals all the parts of KEY_SET_KEY;
(b) The hacker steals the KEY_SET;
(c) 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);
(d) Steal the permanent FILE_KEY (this is protected by the KEY_SET);
(e) Steal the file and try to decrypt it with offline dictionary attacks.

Still, the hacker has to get 4 values from one session: TIME, DEV_PASS, KEY_SET, FILE_KEY. 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. In accordance with the conditions 1 and 3 of our adversary model, the above actions are not possible for such adversary in the conditions presented.

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 malicious or expired user wishes to prolong his old credentials is:

(a) Steal the file and try to decrypt it with offline dictionary attacks.
(b) He has to be able to combine;
(c) He has to steal the KEY_SET synchronized with his credentials;
(d) He has to steal the protected FILE_KEY (also synchronized) and encrypted file;
(e) 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 data to be checked or validated does not prevent the user from seeing the contents of decrypted files, but with the wrong password the decrypted files will be different from the original ones.

The client still has to count the tries (to prevent the hacker to perform brute force attack) within one session. In accordance with the conditions 2 and 4 of the adversary model, the above actions are not possible for such adversary.

Online mode

In the online mode the protection of the mobile system is backed up by the communication with server. The key expiry period and the usage of J-PAKE [14, 15] guarantees that the sensitive data sent over the network cannot leak. The detailed security analysis of J-PAKE is presented in [31]. The data sent in the clear i.e. encrypted files and the public keys of ABE does not need to be protected additionally as the AES and ABE guarantee its security.

6.1.3 Common threat scenarios

This section provides the detailed description of the common scenarios in which the log and behavioral analysis is provided. The behavioral analysis can help to keep the user or administrator informed of the threat and take actions, as well as it can be useful in order to implement threat preventions or reactive actions to avoid threat propagation.

Scenario 1 *An attacker uses a valid password to perform operations on a bulk of files.*

The session time defines the period when operations can be performed until the next session renewing. During this period, it is still necessary to identify attacks and malicious behavior on file operations, in order to avoid fast attacks to perform unauthorized access to information or data modification. Some attacks present behavioral patterns based on abrupt number of operations, such as the ransomware attack, which is a growing attack [27] that blocks the access to valuable resources and requires a payment in order to unblock the content. The access to the resources can be blocked by the attacker through some techniques, when the content is encrypted by the attacker, the ransomware attack can be called cryptoransomware [20].

MOS schemes based on zero mean covariance analysis are effective to reveal abrupt changing of behaviors over time [30], making possible to identify intense malicious behaviors on offline mode of mobile clients, such in case of ransomware attack or bulk access to sensitive data.

The large number of operations over time is a well-known pattern of some attacks, due to the efforts on security measures to make the attacks infeasible over time. In this context, the operations can also be evaluated in contrast to the estimated required time for operations done by legitimate behaviors, such as the evaluation of the mean time between operations, highlighting the occurrence of infeasible behaviors in comparison to legitimate user activities.

Sparse or subtle file operations, with low number of operations distributed over different files or directories, during short period of time can indicate anomalies in contrast to the required time for legitimate directory navigation. MOS and zero mean and standardized covariance analysis can be suitable if applied to evaluate the time and location of operations, in order to identify unreachable navigation, if compared to legitimate navigation

The MOS based on zero mean covariance analysis indicates abnormalities caused by large amounts of operations during a period. The eigenvalue analysis based on the zero mean and standardized covariance highlights massive or concentrated operations over time or folder location, which is evaluated by MOS schemes in order to identify the number of malicious behaviors during the evaluated time.

This threat scenario, where an attacker uses a valid password and session to perform operations on a bulk of files, can have its steps described as:

(a) The hacker has access to the mobile client and is able to perform operations;
(b) The session time is valid;
(c) The hacker tries to perform legitimate operations, such as file decryption, encryption, reading, writing or directory navigations;
(d) The client incrementally append each operation attempt time into the logging;
(e) The MOS module evaluates the logging of legitimate operations, applying zero mean and standardized covariance analysis to identify anomalies on sparse or subtle number of file operations, highlighting the occurrence of infeasible behaviors in comparison to legitimate user activities;
(f) The MOS module evaluates the logging of legitimate operations, applying zero mean covariance analysis to identify abnormalities caused by massive operations during the session time.

Scenario 2 *Usage of expired password to perform unauthorized operations.*

In the offline mode, the session time is used to restrict the operations during a specified period, although it is possible to manipulate the current time in mobile clients, to emulate a period in which the session was valid. The log analysis by MOS can deal with this kind of threat, through the incremental logging of the time when each operation was performed, followed by the behavioral evaluation of operations over time.

The incremental logging assumes that new logged operations shall have equal or bigger time than the last logged operation, the violation of this rule means that the system is out of sync and indicates a malicious behavior. Additionally, a large amount or sparse operation performed at the same time, or during a short period, can indicate the use of backtrack techniques to maintain a valid session during necessary time to perform an attack. Massive, subtle or sparse malicious operation performed during a valid session time can be identified by MOS schemes based on covariance analysis.

Applying MOS to the analysis of the time between user operations can be effective in order to reveal the occurrence of malicious behavior during an offline session. The MOS based on zero mean and standardized covariance analysis identifies anomalies on sparse or subtle variation in the number of file operations, since the this eigenvalue analysis highlights the unexpected number of sparse (such as file operations on diverse folders) or subtle operations. Consequently, the result of the eigenvalue analysis is applied to MOS schemes, in order to identify the occurrence of malicious behaviors during the valid session.

The MOS based on zero mean covariance analysis indicates abnormalities caused by large amounts of operations during a period. The eigenvalue analysis based on the zero mean covariance matrix highlights massive or concentrated operations over time or location, which is evaluated by MOS schemes in order to identify the number of malicious behaviors during the evaluated time.

This threat scenario, where the attacker uses expired password to perform unauthorized operations, can have its steps described as:

(a) The hacker steals the operating system;
(b) The hacker modifies the time of the operating system to a period when the session was valid;
(c) The hacker has access to the mobile client and is able to perform operations;
(d) The hacker tries to perform legitimate operations, such as file decryption, encryption, reading, writing or directory navigations;
(e) The client incrementally append each operation attempt time into the logging;
(f) The client verifies if one logged time is older than the last operation time. If it is true, the MOS module classifies the evaluated operation as malicious;

(g) The MOS module evaluates the logging of legitimate operations, applying zero mean and standardized covariance analysis to identify anomalies on sparse or subtle number of file operations;
(h) The MOS module evaluates the logging of legitimate operations, applying zero mean covariance analysis to identify abnormalities caused by massive operations during the session time;

6.1.4 Data Modeling for Behavioral Analysis

MOS schemes are used in order to identify anomalous behavior that can indicate an attack and be used to prevent or avoid attack propagation. Therefore, it is necessary to analyze the data that can be collected from user operations on mobile client, to identify features that can be modeled and submitted to MOS schemes, according to described in Section 5.4.

The selected features shall be modeled as matrices which represents a signal superposition containing noise, legitimate and malicious behavior [30], grouped into time frames $\mathbf{X}^{(q)} \in \mathbb{R}^{M \times N}$, with $q = 1, 2, 3, \ldots, Q$, where M defines the decomposition of a selected feature, N defines the time decomposition and represents the number of occurrences of the feature m during the time n.

In offline mode, the user is still allowed to get access to operations that do not require communication with the server side. These operations and their selected features shall be incrementally logged by the mobile client, in order to be evaluated to identify malicious behaviors. This work proposes to evaluate the following features.

File Access (Time and File System Location), i.e. data access to selected files in offline mode, accessing the data stored on the mobile client. The file access feature can be decomposed into more detailed features, which are:

1. file decryption;
2. decrypted file reading;
3. decrypted file execution.

Therefore, it is necessary to generate three matrices for malicious behaviors analysis, such as:

(a) massive file access, which can reveal data leakage and be identified by MOS schemes based on zero mean covariance analysis;
(b) low file access into several folders, characterized by sparse operations that can reveal unreachable navigation performed by automated file accesses in order to avoid the massive file access characterization;
(c) Malicious sparse file accesses can be identified by MOS schemes based on zero mean and standardized covariance analysis.

File Update (Time and File System Location), i.e. writing operations into selected files in offline mode, writing the data stored on the mobile client. The update feature can be decomposed into:

1. file encryption;
2. decrypted file writing.

Therefore, it is necessary to generate two matrices for malicious behaviors analysis, such as:

(a) massive file update, which can reveal ransomware or similar attacks and be identified by MOS schemes based on zero mean covariance analysis;
(b) low number of file update into several folders, characterized by sparse operations that can reveal unreachable navigation performed by automated file accesses in order to avoid the massive file access characterization. Malicious sparse file accesses can be identified by MOS schemes based on zero mean and standardized covariance analysis.

File Download (Start Time, End Time and File System Location), i.e. download requests in online mode, evaluated by the mobile client. The file download feature shall be modeled as the matrix of number downloads by file location over time, in order to perform malicious behaviors analysis, such as:

1. massive data leakage or similar attacks, which can be identified by MOS schemes based on zero mean covariance analysis;
2. low number of file download from several folders, characterized by sparse operations, which can reveal unreachable navigation performed by automated file download in order to avoid the massive file download characterization. Malicious sparse file download can be identified by MOS schemes based on zero mean and standardized covariance analysis.

File Upload (Start Time, End Time and File System Location), i.e. upload requests in online mode, evaluated by the mobile client. The file upload feature can reveal attempts of ransomware or similar attacks and be identified by MOS schemes based on covariance analysis. Therefore, it is necessary model the matrix of number uploads by file location over time, in order to perform malicious behaviors analysis, such as:

1. massive file upload, similar to ransomware attack, which can be identified by MOS schemes based on zero mean covariance analysis;
2. low number of file upload to several folders, characterized by sparse operations, which can reveal unreachable navigation performed by automated file

upload in order to avoid the massive file upload characterization. Malicious sparse file upload can be identified by MOS schemes based on zero mean and standardized covariance analysis.

6.2 Complexity analysis

The complexity analysis depends on the operation that the user and the mobile app perform in order to keep the device protected. In the offline mode the client app performs the following actions:

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 128/256 AES 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 128/256 AES → decryption;
6. Modify the TIME periodically → timer;
7. Count the tries within the TIME → count;
8. Modify or delete PUBLIC_SHARE_KEY → no calculation;
9. Malicious behavior detection → eigenvalues calculation.

It is easy to check that in the offline mode the client app does not perform complex calculations and does not use the resources extensively due to the fact that the initial key is shared and the client app only performs decryption, which is not a time-consuming operation. Thus, the proposal supports the concept of the lightweighted client, i.e. the most consuming operations are ABE and AES decryption.

Similarly, in the online mode the client app does not perform resource-consuming operations apart from J-PAKE construction which is used to renew the user/app master key (DEV_PASS):

1. Generates keys for J-PAKE -¿ this can be a resource-consuming operation. In the future we suggest to replace J-PAKE with SIS-based PKE [4, 3];
2. Sends and receives data in the clear;
3. Performs operations 1)-7);
4. Performs the operation 8) and sends the log data for server side analysis.

The proposed concept of mobile client security has been implemented in the Storgrid protected cloud environment [2]. Therefore, the approach is correlated with the practical usability requirements: the corporate user continues to use the mobile storage app in offline and does not need to reload the files every time the key is renewed. This methodology can be used in other mobile apps. The common advantage is that the mobile client performs the operations both in the offline and online mode and uses the key expiry and ABE to protect the privacy of the corporate data. The Table 1 presents the results of mobile app security workflow testing. Table 1 demonstrates that the device decrypts the data with the acceptable speed. One can compare the speed of ABE/AES decryption (keys and actual data) and the speed of the whole decryption process.

Table 2 demonstrates the results of testing the speed of the mobile device for receiving and displaying the list of files. The app was tested for displaying up to 1000 files. Around 100 requests were generated from 10 threads.

The device receives and displays the list of files depending on the user attributes with acceptable average speed and the proposed protection scheme is usable. Table 3 demonstrates the results of performance testing of the keys generation procedure on server:

It is easy to observe that the performance does not decrease drastically with the increase of particular user shares. Each key sample generated contains the domain member code, key data and the sharing key value for each file sharing code. It is possible to conclude that the proposed client is not overloaded with calculations due to the carefully selected mathematical operations. It can be successfully used and provides acceptable level of security.

The log analysis of the Log Analysis Center (LAC) has been implemented and evaluated for offline anomaly detection in mobile clients, making it possible to apply anomaly detection techniques in a lightweight fashion, considering low processing requirements for deal with the resource constraints of mobile clients. The evaluation considered the required processing time for anomaly detection from log analysis, measuring the data modeling time through the UAL, the eigenvalue decomposition time and the required time for the EDC MOS scheme execution, which is the scheme that requires less processing capacity and provides more anomaly identification accuracy [6, 30]. The experiments were performed in two mobile devices, Galaxy GT-I9300 and Galaxy Tab SM-T800, with variations of log size and window size. The Galaxy GT-I9300 has Quad-core 1.4 GHz Cortex-A9 processor and 1 GB RAM, while the Galaxy Tab SM-T800 has its processing capacity composed by Quad-core 1.9 GHz Cortex-A15 and quad-core 1.3 GHz Cortex-A7, and 3 GB RAM.

Table 4 presents the data modeling time and the processing time of eigenvalues decomposition calcula-

Table 1 Speed of decryption on a client device

Device Model	File set size	Mean time of getting decrypted stream (ABE)	Mean time of decrypting content (AES)	Mean time of copying file without decrypt	Mean time of decryption (the whole file)
Samsung SM-T320 Galaxy Tab Pro 8.4	10 mb	1041 ms	4230 ms	298 ms	5271 ms
Samsung SM-T320 Galaxy Tab Pro 8.4	1 mb	811 ms	319 ms	23 ms	1130 ms
Samsung SM-T320 Galaxy Tab Pro 8.4	1 kb	1078 ms	1 ms	0 ms	1080 ms
Samsung GT-I9190 Galaxy S4 Mini	1 b	1167ms	2 ms	0 ms	1169 ms
Samsung GT-I9190 Galaxy S4 Mini	1 kb	1141ms	2 ms	0 ms	1144 ms
Samsung GT-I9190 Galaxy S4 Mini	1 mb	1138 ms	368 ms	35 ms	1506 ms
Samsung GT-I9190 Galaxy S4 Mini	10 mb	1131 ms	3618 ms	324 ms	4750 ms

Table 2 Speed of receiving the list of files

Files quantity	Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Error (%)	Throughput	KB/sec	Avg. Bytes
1	1000	131	29	321	31.56	0.00%	74	161.32	2233
10	1000	154	37	269	32.24	0.00%	63.4	659.79	10656
100	1000	396	68	1055	87.56	0.00%	25	2323.2	95308
1000	1000	2879	475	4634	496.02	0.00%	3.4	3178	946311

Table 3 Speed of generating the user keys

Label	Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Throughput	KB/sec	Avg. Bytes
Login 5 shares available	1000	982	225	1990	271.88	20.1	146.36	7459
Login 10 shares available	1000	1140	340	4176	459.3	17.3	223.88	13218
Login 25 shares available	1000	1267	887	2303	192.75	15.5	294.55	19466

tions to be applied to anomaly detection from user operation logs of Storgrid mobile client. The information presented by column are the device model, the log size in megabytes, the window size in minutes, the data modeling time in milliseconds, the average of eigenvalue decomposition time in milliseconds, the standard deviation of eigenvalue decomposition time in milliseconds, the minimum of eigenvalue decomposition time in milliseconds and the maximum of eigenvalue decomposition time in milliseconds.

The results show that the lower window size leads to the larger eigenvalue decomposition time, but the largest eigenvalue decomposition time, which was the maximum of 421 milliseconds with average of 347.42 milliseconds. This result highlights an acceptable speed even for the worst evaluated scenario, which is the Galaxy GT-I9300 processing 6MB with window size of 10 minutes.

Table 5 presents the processing time of EDC MOS calculations applied to anomaly detection from user operation logs of Storgrid mobile client. Table 5 respectively presents the device model, the log size in megabytes, the window size in minutes, the average of EDC calculation time in milliseconds, the standard deviation of EDC calculation time in milliseconds, the minimum of EDC calculation time in milliseconds and the maximum of EDC calculation time in milliseconds.

It is possible to observe that the processing time increases with the window size decreasing, similar to the results for eigenvalue decomposition time. The longest processing time measured is lower than 200 milliseconds, even considering window size of 10 minutes or processing 6 MB of user operation log. This result rep-

Table 4 Data Modeling and Eigenvalue Decomposition Time

Device	Log Size (MB)	Window (min)	Modeling (ms)	Avg. Eig. (ms)	Std. Eig. (ms)	Eig. Min. (ms)	Eig. Max. (ms)
Galaxy GT-I9300	6	60	107	209.52	18.58	183	276
Galaxy GT-I9300	6	40	115	227.26	18.13	191	289
Galaxy GT-I9300	6	20	89	268.14	21.94	229	315
Galaxy GT-I9300	6	10	90	347.42	24.11	304	421
Galaxy GT-I9300	4.1	60	20	60.90	15.19	37	106
Galaxy GT-I9300	4.1	40	20	68.72	15.71	43	114
Galaxy GT-I9300	4.1	20	34	89.04	16.78	54	133
Galaxy GT-I9300	4.1	10	21	117.24	14.36	96	171
Galaxy GT-I9300	1.4	60	10	159.82	15.82	125	197
Galaxy GT-I9300	1.4	40	10	168.06	15.90	139	220
Galaxy GT-I9300	1.4	20	11	204.4	20.46	176	269
Galaxy GT-I9300	1.4	10	13	259.00	21.34	220	315
Galaxy Tab SM-T800	6	60	7	59.30	6.55	54	74
Galaxy Tab SM-T800	6	40	8	62.56	7.05	56	80
Galaxy Tab SM-T800	6	20	10	73.28	8.59	65	95
Galaxy Tab SM-T800	6	10	8	93.48	9.13	83	130
Galaxy Tab SM-T800	4.1	60	11	18.64	4.51	16	38
Galaxy Tab SM-T800	4.1	40	11	19.64	5.12	17	38
Galaxy Tab SM-T800	4.1	20	12	25.12	5.55	21	46
Galaxy Tab SM-T800	4.1	10	12	32.32	7.29	27	55
Galaxy Tab SM-T800	1.4	60	4	49.08	6.01	42	62
Galaxy Tab SM-T800	1.4	40	5	51.42	7.36	44	74
Galaxy Tab SM-T800	1.4	20	5	51.12	7.80	54	91
Galaxy Tab SM-T800	1.4	10	7	75.24	7.71	65	90

Table 5 EDC MOS scheme processing time for anomaly detection

Device	Log Size (MB)	Window (min)	Avg. EDC. (ms)	Std. EDC. (ms)	Min. EDC. (ms)	Max. EDC. (ms)
Galaxy GT-I9300	6	60	5.27	4.04	3	20
Galaxy GT-I9300	6	40	10.78	6.37	6	34
Galaxy GT-I9300	6	20	32.62	12.44	21	88
Galaxy GT-I9300	6	10	115.08	17.45	88	158
Galaxy GT-I9300	4.1	60	5.68	4.18	3	23
Galaxy GT-I9300	4.1	40	10.76	5.31	7	27
Galaxy GT-I9300	4.1	20	37.58	10.30	23	61
Galaxy GT-I9300	4.1	10	125.98	18.56	101	191
Galaxy GT-I9300	1.4	60	4.92	3.49	3	17
Galaxy GT-I9300	1.4	40	9.00	4.23	6	25
Galaxy GT-I9300	1.4	20	30.14	9.21	19	62
Galaxy GT-I9300	1.4	10	100.62	15.83	69	163
Galaxy Tab SM-T800	6	60	1.84	0.65	1	3
Galaxy Tab SM-T800	6	40	3.26	1.24	2	7
Galaxy Tab SM-T800	6	20	10.90	2.40	9	21
Galaxy Tab SM-T800	6	10	41.86	7.33	34	60
Galaxy Tab SM-T800	4.1	60	1.85	0.60	1	3
Galaxy Tab SM-T800	4.1	40	3.62	1.10	2	8
Galaxy Tab SM-T800	4.1	20	12.04	2.79	9	22
Galaxy Tab SM-T800	4.1	10	40.16	6.48	35	60
Galaxy Tab SM-T800	1.4	60	1.98	0.89	1	6
Galaxy Tab SM-T800	1.4	40	3.30	1.16	2	7
Galaxy Tab SM-T800	1.4	20	10.48	2.90	8	21
Galaxy Tab SM-T800	1.4	10	34.52	4.08	30	45

resents an acceptable processing time for anomaly detection in mobile devices.

7 Conclusion and future work

An important security issue faced by corporations that use cloud-based systems is how to provide security mechanisms to support offline corporate mobile devices. Once a mobile device releases the connection with the corporate cloud, no security measure implemented in the cloud infrastructure assures the protection of sensitive data stored in the mobile device. Aware of this problem and its importance, this work presented a proposal to address the offline mobile security problem combining a different cryptographic methods. Moreover, this proposed approach also prevents malicious user behavior by applying a MOS-based analytic method.

As prove of concept, a fully working mobile application was developed to test the proposed security solution and acquired results provide evidence that besides achieving the desired security features, the solution also has positive results in terms of performance. This fact is due to the usage of lightweight operations and the optimized combination of the selected security methods. The proposed approach is a practical application to be used in the corporate mobile environment. It is implemented as a fully working mobile app and can be used for any type of enterprise. Also, part of concept is seed for new security solutions for big data apps.

Future works in the area can further explore enhancements in the analytics methods as well as to extended the approach to be used by mobile devices with even more severe resources constraints.

Acknowledgements The authors thank the Brazilian research and innovation agencies FAPDF (Research Support Foundation of the Federal District), FINEP (Agreement RENASIC / PROTO 01.12.0555.00), CAPES and CNPq under the FORTE Project - CAPES Forensic Sciences Announcement 25/2014, under the productivity grant number 303905 / 2014-0, under the PVE project number 88881.030392 / 2013-01 for their financial support on this research, and under the project number 207644/2015-2 for the PDE scholarship in the context of the program CSF- Aerospace Technology.

References

1. (2016) Cloud security alliance. URL https://cloudsecurityalliance.org/, accessed: 2016-01-15
2. (2016) Storgrid protected cloud storage security whitepaper. URL http://www.storgrid.com, accessed: 2016-01-15
3. Bellovin SM, Merritt M (1992) Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on, IEEE, pp 72–84
4. Bogos S, Boureanu I, Vaudenay S (2013) Primeless factoring-based cryptography. In: Applied Cryptography and Network Security, Springer, pp 552–569
5. Campbell, M (2015) Cloud data encryption is easy. Cloud Cyphercloud blog URL http://www.ciphercloud.com/blog/cloud-data-encryption-easy/, accessed: 2016-01-15
6. Da Costa J, Thakre A, Roemer F, Haardt M (2009) Comparison of model order selection techniques for high-resolution parameter estimation algorithms. In: Proc. 54th International Scientific Colloquium (IWK'09), Ilmenau, Germany
7. Galibus T (2014) Access control for the cloud storage. In: Proceeding of the 3rd Belarus-Korea Forum Science "Innovation, Production", Minsk
8. Galibus T, Matveev G (2007) Generalized mignotte sequences in polynomial rings. ENTCS 186:39–45
9. Galibus T, Vissia H (2015) Cloud storage security. In: Network Security and Communication Engineering: Proceedings of the 2014 International Conference on Network Security and Communication Engineering (NSCE 2014), Hong Kong, December 25–26, 2014, CRC Press, p 123
10. Galibus T, Matveev G, Shenets N (2008) Some structural and security properties of the modular secret sharing. In: Symbolic and Numeric Algorithms for Scientific Computing, 2008. SYNASC'08. 10th International Symposium on, IEEE, pp 197–200
11. Galibus T, Gafurov S, Kaganovich D, Vissia H (2015) Mobile security based on the secret sharing. The journal of Brest state technical university 5:33–36, in Russian
12. Gartner (2015) Key challenges in cloud computing. Cloud Computing URL http://www.gartner.com/technology/topics/cloud-computing.jsp, accessed: 2016-01-15
13. Goyal V, Pandey O, Sahai A, Waters B (2006) Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on Computer and communications security, Acm, pp 89–98
14. Hao F, Ryan P (2010) J-pake: authenticated key exchange without pki. In: Transactions on computational science XI, Springer, pp 192–206
15. Hao F, Ryan PY (2008) Password authenticated key exchange by juggling. Springer
16. Higashi M (2015) Cloud data security and eu data privacy rules compliance with encryption and tokenization. Cloud Security, Compli-

ance URL http://www.ciphercloud.com/blog/cloud-data-security-and-eu-data-privacy-rules-compliance-with-encryption-and-tokenization/, accessed: 2016-01-15
17. Huang CT, Chang RK, Huang P (2009) Signal processing applications in network intrusion detection systems. EURASIP Journal on Advances in signal Processing 2009(1):1–2
18. Itani W, Kayssi A, Chehab A (2010) Energy-efficient incremental integrity for securing storage in mobile cloud computing. In: Energy Aware Computing (ICEAC), 2010 International Conference on, IEEE, pp 1–2
19. James N, Elaine B, Lawrence B, William B, Morris D, James F, Roback E (2000) Report on the development of the advanced encryption standard (aes). NYST URL http://csrc.nist.gov/archive/aes/round2/r2report.pdf, accessed: 2016-01-15
20. Kaspersky (2014) Mobile cyber threats. Kaspersky Lab & INTERPOL Joint Report URL http://media.kaspersky.com/pdf/Kaspersky-Lab-KSN-Report-mobile-cyberthreats-web.pdf, accessed: 2016-01-15
21. Khan AN, Kiah MM, Khan SU, Madani SA (2013) Towards secure mobile cloud computing: A survey. Future Generation Computer Systems 29(5):1278–1299
22. Khan AN, Kiah MM, Ali M, Madani SA, Shamshirband S, et al (2014) Bss: block-based sharing scheme for secure data storage services in mobile cloud environment. The Journal of Supercomputing 70(2):946–976
23. Khan AN, Kiah MM, Ali M, Shamshirband S, et al (2015) A cloud-manager-based re-encryption scheme for mobile users in cloud environment: a hybrid approach. Journal of Grid Computing 13(4):651–675
24. Khan AR, Othman M, Madani SA, Khan SU (2014) A survey of mobile cloud computing application models. Communications Surveys & Tutorials, IEEE 16(1):393–413
25. Lawson C, MacDonald N, Lowans B (2015) Market guide for cloud access security brokers. Gartner research URL http://www.gartner.com/technology/reprints.do?id=1-2RUEH70&ct=151110&st=sb, accessed: 2016-01-15
26. Lu W, Ghorbani AA (2009) Network anomaly detection based on wavelet analysis. EURASIP Journal on Advances in Signal Processing 2009:4
27. McAfee (2015) Mcafee labs threats report. URL http://www.mcafee.com/us/resources/reports/rp-quarterly-threats-aug-2015.pdf, accessed: 2016-01-15
28. Ren W, Yu L, Gao R, Xiong F (2011) Lightweight and compromise resilient storage outsourcing with distributed secure accessibility in mobile cloud computing. Tsinghua Science & Technology 16(5):520–528
29. Skyhigh (2015) What is cloud access security broker. Skyhigh Cloud University URL https://www.skyhighnetworks.com/cloud-university/what-is-cloud-access-security-broker/, accessed: 2016-01-15
30. Tenório DF, da Costa JPC, de Souza Júnior RT (2013) Greatest eigenvalue time vector approach for blind detection of malicious traffic. ICoFCS 2013 p 46
31. Toorani M (2014) Security analysis of j-pake. In: Computers and Communication (ISCC), 2014 IEEE Symposium on, IEEE, pp 1–6
32. Van Lelyveld, A (2013) Sap mobile platform secure mobile with mocana. SMP Enterprise Grade Mobility URL http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/8063ed15-0713-3110-c584-e75ac0395b20?QuickLink=index&overridelayout=true&58725087881424, accessed: 2016-01-15
33. Yang J, Wang H, Wang J, Tan C, Yu D (2011) Provable data possession of resource-constrained mobile devices in cloud computing. Journal of networks 6(7):1033–1040
34. Yovel, Y (2014) Essential ways to protect my mobile apps. Security Intelligence e-magazine URL https://securityintelligence.com/how-to-protect-mobile-apps-essentials/, accessed: 2016-01-15
35. Zhao G, Rong C, Li J, Zhang F, Tang Y (2010) Trusted data sharing over untrusted cloud storage providers. In: Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, IEEE, pp 97–103