Cross-Platform End-to-End Encryption of Contact Data for Mobile Platforms using the Example of Android

Markus Hofmarcher, Michael Strauß, and Wolfgang Narzt
Department of Business Informatics – Software Engineering
Johannes Kepler University Linz
markus.hofmarcher@gmail.com, michael@strauss.eu.com, wolfgang.narzt@jku.at

***Abstract*— Storing and synchronizing personalized or business-related data, especially contact data, is increasingly done by cloud services. Thereby, control over personal data is subject to the technical conditions and measures of the cloud service providers. However, abstaining from the utilization of cloud services is not an alternative as their amenities are indispensable both for private and business users. End-to-end encryption, as it is already maturely applied for various communication services, would enable users to still keep their contacts on remote storage nodes, but save them in encrypted form. Although, the principal concepts for this kind of security measure are well-studied, there is still no service for protecting cloud-based contact data by end-to-end encryption using mobile platforms. This paper presents the ideas and the architecture for end-to-end encryption of contact data using the example of Android.**

Keywords – end-to-end encryption; cloud services; Android; contact data; cross-platform

I. INTRODUCTION

The list of one's own contacts is the backbone for modern communication. Cloud services offer powerful interfaces for conveniently managing this data on remote servers, from where they can be synchronized to arbitrary clients and platforms. They have become indispensable for present communication systems, from which private as well as business users cannot escape.

However, utilizing cloud services is potentially unsafe and simultaneously means giving up control to the service providers due to varying international security laws and regulatory frameworks. Users are unaware of the location and the amount of physical server nodes where their data is stored on. As a consequence, legal positions and applicable laws are not transparent to the users, especially for the private user sector [1]. Deleting data can also be challenging, as the physical deletion is not executed in many cases even when users request it [1]. Moreover, recent press releases have pointed out the drawbacks of cloud services in respect to security and compliance [2][3]. Missing transparency and the lack of control and security mechanisms are mentioned as the major issues.

Nevertheless, cloud services provide a valuable method for managing and synchronizing data and are even obligatory nowadays for many business companies [4]. Hence, we propose a method for protecting cloud-based data by (well-proven) end-to-end encryption. In particular, we refer to contact data to be securely exchanged and synchronized with mobile devices. The advantages of cloud services remain; only the data is stored encrypted. We present a generic cross-platform architecture, show an exemplary implementation for the Android platform, where users can select those apps, which are granted access to the decrypted contact data, and finally discuss re-utilization of the concept for other mobile platforms. Simplicity and usability are major aspects to be considered in our work, in order to raise acceptance among the users, whereby the main reasons people refrain from using encryption for email and file transfer are the lack of knowledge and the additional time and effort for installing and using security features [5].

The paper is organized as follows. Section II discusses related work and similar approaches. Section III introduces our architecture followed by its implementation in Section IV. Section V shows the results of our investigation. Sections VI and VII discuss possible solutions for other platforms and concepts for synchronization. Finally, in Section VIII we draw conclusions and formulate ideas for future work.

II. RELATED WORK

Among the scientific investigations concerning privacy protection and information security in cloud systems, research is mainly focused on strategies and mechanisms offered by the cloud service provider [6][7][8] or is even restricted to the selection of trustworthy providers. Security issues concerning cloud services are thoroughly analyzed with respect to architecture, data delivery models and from the stakeholders' perspective [9].

In order to overcome the uncertainties of cloud services, Puttaswamy et al. [5] propose to insert a trustworthy (because self-managed) organizational node in between client and cloud in order to pre-encrypt data before transferring them to the cloud. The organizational node holds the keys and additionally identifies functionally encryptable data, i.e., data never interpreted by the cloud and not breaking application functionality. It automatically encrypts them and generates and provides appropriate access keys for its users. Thus, the authors' proposal represents an approach of minimizing the weak points of cloud services by an intercalated server instance offering end-to-end encryption. End-to-end encryption, in general, is the preferred method for storing data in cloud services [10][11].

Practically applied end-to-end encryption can be found, e.g., in PrivacyCrypt for Facebook [12], a special security feature for encrypting messages in the social network Facebook. The public key used for asymmetric encryption

 ISBN: 978-1-61208-347-6

has to be exchanged with the target users who are allowed to decrypt and read those messages. A Firefox extension contains the implementation for this security feature. Fig. 1 illustrates its principle scheme. The left part in Fig. 1 shows the message in plain language, which is encrypted by the Firefox plugin at the client side using the public key(s) of the receiver(s). The encrypted message is then transferred to and stored on the server ("Facebook cloud"). Facebook consequently receives encrypted messages, which can only be decrypted by those having the appropriate private key.

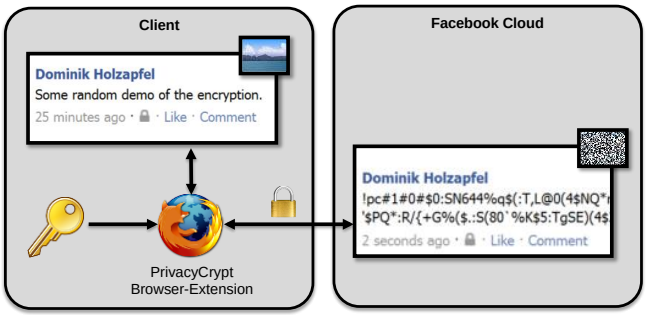

Figure 1. PrivacyCrypt for Facebook

Boxcryptor [13] uses a similar approach. This application supports users encrypting their data stored on cloud servers such as Dropbox, Google Drive or Microsoft SkyDrive. First, the files are encrypted locally on a user's device and afterwards sent to and stored on a remote server. Boxcryptor is available for Microsoft Windows, Mac OSX, iOS and Android. Furthermore, there is an extension for Google Chrome which enables users to access and decrypt their data via a browser instance.

Fig. 2 briefly explains Boxcryptor's support for two exemplary platforms (desktop with Google Chrome and Android) having a comparable working procedure to PrivacyCrypt as both approaches use end-to-end encryption.

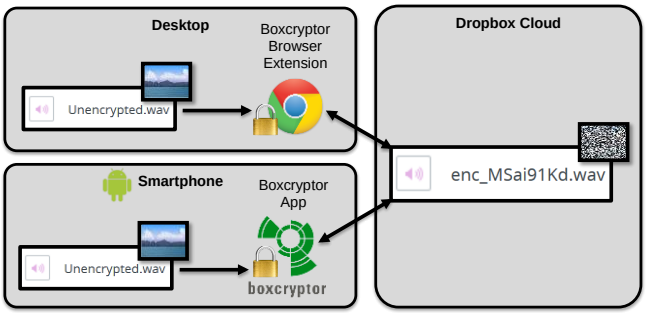

Figure 2. BoxCryptor

End-to-end encryption is also our proposed method to securely store (contact) data on cloud servers, allowing its users to keep control over their own data. Privacy and information security is therefore guaranteed by end-to-end encryption [11]. Even insufficient protection mechanisms of cloud service providers have no negative effects in respect to data security as the data can only be decrypted by the owners themselves as long as they use state-of-the-art keys [14].

There are numerous existing solutions for data encryption. For instance the first version of Pretty Good Privacy (PGP) was released in 1991 and is nowadays, beside S/MIME, an established standard of e-mail encryption. Although several implementations are available, PGP or S/MIME are not commonly used, despite the high need of e-mail encryption. A case study pointed out that it is hardly possible to set up or use PGP without specialized knowledge [15]. Lack of knowledge and inconvenience are the main reasons for not using encryption [16].

III. APPROACH

The challenge in developing a suitable method for encryption is security and also usability. To achieve high acceptance it is necessary to design the application for encryption as transparent and as user friendly as possible. Furthermore, compatibility with third party applications must be ensured. In conclusion, we claim that there are three primary "quality" criteria which must be fulfilled by our solution:

- Comfort
- Transparency
- Security

Comfort means that the use of cloud services is still possible even when the data is stored in encrypted form. That must be possible without or rather with little migration effort. In addition to this, the user should be able to use encrypted data like unencrypted data. A password recovery concept is also important. Another aspect is multi-platform support. The user should not be limited, and therefore, the encryption solution should work on the most common platforms.

Third party applications must still be able to access contact data if this is allowed by the user. Hence, the encryption mechanisms must be implemented transparently so that there is no need to adapt these third party applications. They should be able to access data when needed in the same way even though the data is stored in encrypted and decrypted form. This is important for software components providing contact data synchronization.

Of course, data must be encrypted securely. That means an up-to-date algorithm for encryption must be used. Fig. 3 outlines the common data flow of contact data in combination with cloud services for synchronization.

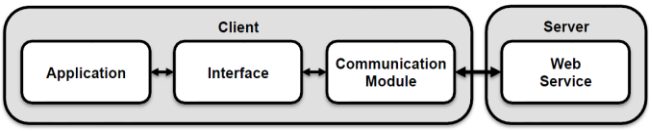

Figure 3. General contact data flow

Data is used and managed by the client application, which usually provides a GUI enabling the user to read or change contact data. These applications retrieve data from a (central) interface, where sometimes data persistence is also implemented. To synchronize data with cloud services a communication module is necessary.

Depending on the platform some of the described modules are possibly bundled in one component. However,

 ISBN: 978-1-61208-347-6

as a general rule data is processed according to the following steps:

- Representation and user access (application)
- Data management (interface and persistence layer)
- Server communication (through a communication module)

These steps are potential levels where en- or decryption can take place. According to the three mentioned quality criteria, it is inappropriate to implement encryption at the application level because that would require adaptations of every used application.

Fig. 4 shows the discussed data flow specifically for Android. There are three applications (People, Facebook and Email) that are able to access contact data. The central Android interface for accessing contacts is called "Contacts Provider". All requests are handled by this interface, which also implements data persistence. "Sync Adapters" are responsible for synchronizing data with cloud services and resolving conflicts.

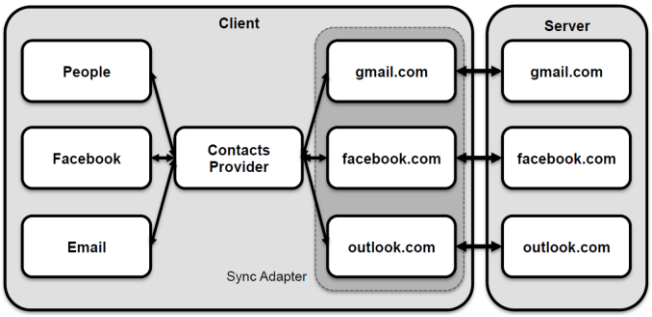

Figure 4. Android contact data flow

Further, it is also improper to implement encryption at the level of the communication module. This would also require adapting every used Sync Adapter in the case of Android. Therefore, we chose the central access interface (Contacts Provider) for our proof-of-concept to implement encryption. This ensures that all quality criteria are fulfilled, which is further discussed in Section V.

For this implementation to work seamlessly the internal data structure of the contacts has to remain unchanged and only the value of the individual fields can be encrypted so applications and backend services can still process the data. The central data interface (Contacts Provider) provides three methods with the following functionalities:

- "query": Retrieve one or more contacts
- "update": Modify or delete an existing contact
- "insert": Create a new contact

A hook on every one of the three mentioned methods is an easy and effective way to intercept all requests according to contact data operations.

IV. IMPLEMENTATION

Before implementing the proposed architecture for Android, it was necessary to determine if it is possible to set hooks on method calls of other processes and applications. Our research shows that there is no official interface for this purpose due to Android's security concepts, but as Android is an open source platform, this functionality has been added by external developers. One such implementation is provided by the "Xposed Framework" which we used to implement a module according to the proposed architecture.

Installing this framework requires so-called "root access" as it needs privileged access rights. The process of attaining this privileged access varies from device to device and can lead to a partial or complete loss of warranty. We are aware that this can pose a significant obstacle, especially in regards to ease of use, but at the time of writing we are not aware of any alternative. This is a direct result of Android's security concepts not providing access control mechanisms for third party applications. Conventional server and desktop platforms provide various ways of managing permissions, while on Android the user typically has no way of gaining privileged access rights. The idea behind this limited privileges concept is to prevent applications from damaging the system. This concept, however, neglects the need for privileged access for security applications like anti-virus software or firewalls.

Android applications are executed in a virtual machine, called Dalvik, which is a specialized implementation of the Java VM. The Xposed Framework loads additional classes to every instance of a Dalvik-VM allowing the implementation of hooks for every method call inside such an instance. A hook is a method that is called before or after a specified method and allows modifications to a method's parameters or its return value.

The prototype that was developed to prove the viability of the proposed architecture was implemented as a module for the Xposed Framework. This module defines hooks on the query, insert and update methods to Android's internal Contacts Manager to implement encryption and decryption of all contacts handled by this central manager.

Fig. 5 shows a schematic representation of a query method call to obtain contact details from the Contacts Provider. The highlighted modules have been implemented by us.

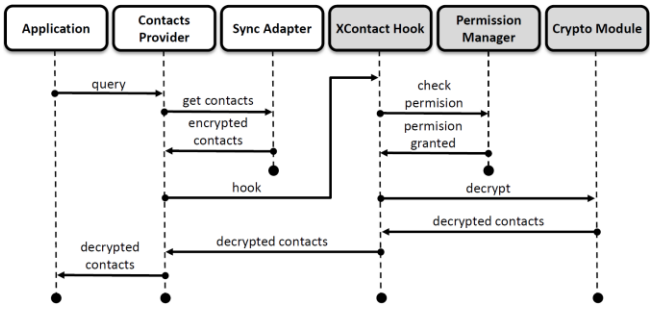

Figure 5. Sequence of a query action

A. *XContact Hook*

The module "XContact Hook" provides the necessary hooks on the three methods specified before to intercept calls of all applications to the Contacts Provider. It defines hooks that are executed before calls to the insert and update methods of the Contacts Provider. The new or updated contact data is encrypted in these hooks and then passed to

the insert or update methods. It also defines a hook that is executed after the query method collected the contacts specified by its parameters. These are then decrypted and passed to the calling methods if the application has the permissions to get the decrypted contacts. In both cases, the data structure of the contact data is kept intact, and internal ID fields are not encrypted to keep compatibility with applications and sync adapters.

B. *Permission Manager*

This module provides an interface for managing the permission on an application level. It enables the user to enable or disable the cryptographic module for specific applications, therefore granting only trusted applications access to unencrypted contact data.

C. *Crypto Module*

The cryptographic module provides routines for encrypting and decrypting contact data while maintaining interface compatibility by preserving the structure of the encrypted data. For encryption the Advanced Encryption Standard (AES) [17] algorithm is used as it is implemented on Android and at the time of writing is sufficiently secure [18]. Initially, a key length of 256 bit was used but with a large number of contacts the additional processing required for encryption and decryption was perceptible by the user.

Since only field data is encrypted, a new random initialization vector is used for each field to mitigate frequency analysis and known plaintext attacks.

V. RESULTS

The major result of our investigations is an architecture concept for end-to-end encryption and a proof-of-concept for Android. Furthermore, we analyzed the performance impact of encryption key lengths and discussed how the proposed architecture could be applied on other platforms. The defined three quality criteria are fulfilled by the proof-of-concept.

Comfort: Although the contact data is encrypted the user is able to use it as usual. The only difference is that users are able to manage permissions with the "XContact Permission Manager" (see Fig. 6). We also discussed potential ways for recovering the password. Furthermore, the user is not forced to use a specific platform because the concept is multi-platform capable.

Transparency: All cloud services for remote data storage can be used with our proof-of-concept because the structure of the data is not changed. Only the data fields are encrypted. Also third party applications (apps) are able to access contact data as usual. There are no specific changes for them.

Security: AES, which was published by the National Institute of Standards and Technology as a standard for data encryption in 2000, is a secure encryption algorithm. There are no known relevant attacks and even a successful brute-force attack is hardly possible [19]. The private key for encryption is only stored on the user's device. So, the cloud service provider is unable to decrypt the users' data. In case of any errors, the calling action is not executed. So, there is no chance of an unencrypted data leak.

A. *Proof-of-concept (prototype)*

In this section, we present our proof-of-concept for end-to-end contact data encryption on Android. The main components of "XContact" are:

- Xposed Hook
- XContact Permission Manager

Xposed Hook works completely in background and is used to hook every relevant method call on the Android Contacts Provider. XContact Permission Manager is an app that allows the user to control which applications are able to access unencrypted contact data.

For installation it is necessary to install the Xposed Framework. This is done in four steps, and, as already mentioned, requires root-permissions. After that our prototype can be installed with an APK file. To use the described features it is necessary to active the XContact module (see Fig. 6). So, users can select those applications they would like to grant access to unencrypted contact data.

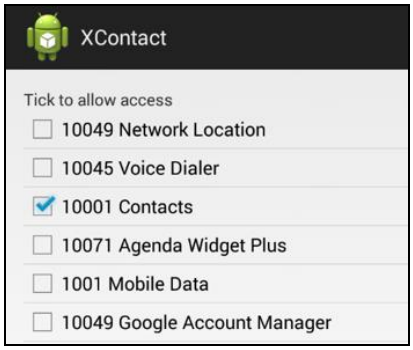

Figure 6. XContact Permission Manager GUI

The XContact Permission Manager shows all installed applications that have requested permission to access contact data at installation time. If the checkbox is not ticked, the application will receive an empty list when it calls the "query" method, and new contacts will not be encrypted. Since XContact encrypts data transparently, the procedures to retrieve, modify or delete contact data are not affected. Therefore, encrypted data can be automatically synchronized with cloud services like Google Contacts.

B. *Performance impact measurements*

In this section we discuss the performance impact of encryption. Our test device is a Nexus 7 (2013) tablet running Android 4.3.1 and Xposed 2.4.1. For our measurements the time at the beginning and the end of an en- or decryption hook have been monitored. The difference of these values is the additional amount of time consumed by the encryption algorithms.

For every test we used the same contact with three fields (name, phone number and e-mail). Fig. 7 describes the arithmetic mean of 30 measurements.

 ISBN: 978-1-61208-347-6

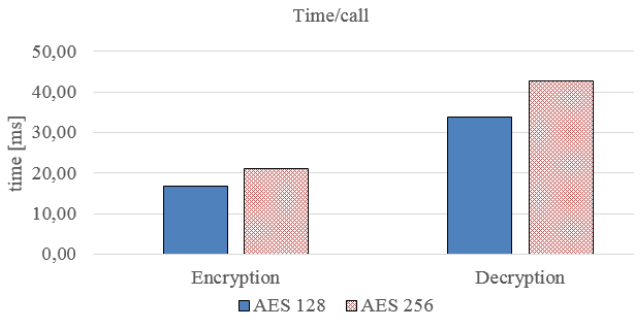

Figure 7. Performance Measurements

The comparison of AES-128 and AES-256 shows that the latter encryption algorithm is about 25% slower. AES-128 is state-of-the-art and generally considered as secure [19]. A key length of 128 bit for symmetric encryption is recommended for long-term data protection [18].

The general performance impact depends on many different factors (e.g., used CPU, other running processes, number of different address books, number of contact groups and single contacts). With our test device and one contacts application we were not able to visually perceive or recognize any delays for different use cases.

VI. PASSWORD AND SYNCHRONIZATION

The nature of the proposed architecture requires a different approach to certain basic features. These features are typically implemented server-side, however, our approach is located only on the client. It, therefore, is essential to minimize the risk for the users losing access to their data due to a forgotten password. Typically, services provide mechanisms for resetting passwords or asking secret questions, but this would require storing the users' password on the device or a server-side implementation.

An alternative method for solving the password problem is to securely store passwords with a password manager, so the user only has to remember one password to access all other passwords. The passwords for multiple services are stored in an encrypted container that can be backed up using cloud services or other backup solutions as it follows the same end-to-end encryption principles as our approach.

While this method cannot strictly be labeled, we believe that password recovery offers the best trade-off between comfort and security. Furthermore, it can be implemented without the need for a server or service provider and, therefore, is best suited for the proposed architecture.

A stated goal of this paper is to utilize data encryption as user-friendly as possible. The platform independent aspect suggests that users may use the application on multiple devices and, therefore, have to enter the password more than once. Also, in the event of changing the password, the new password has to be propagated to all devices using the encryption module requiring a synchronization mechanism. The client-side implementation requires alternatives to usual cloud-based synchronization methods.

We propose the use of a peer-to-peer approach for synchronizing the password across multiple devices and platforms. A discovery protocol identifies relevant nodes on a local network, while a specification of the transmission protocol is needed for transmitting the data.

Based on such a protocol it is possible to synchronize passwords across multiple devices on a local network. If the user installs the encryption module on a new device, it can detect already configured devices on the same network and request the password from them. The user is notified about this request and can either permit or deny it. It is also possible to propagate password changes to all relevant devices on a local network.

VII. CONCEPTS FOR ALTERNATIVE PLATFORMS

Enabling transparent end-to-end encryption for platforms other than Android-based operating systems means implementing an application for each targeted platform. While the proposed architecture was designed with minimal implementation overhead in mind, the nature of end-to-end encryption makes this impossible to avoid.

Platforms can be divided into two main categories with different requirements for implementation.

A. *Desktop Platforms*

A variety of operating systems and applications exist for desktop computers and workstations that require contact data. Therefore, we focus our research on web-browsers as they are available on multiple platforms and are a common way to access and manage contact data, largely because service providers often provide web applications for doing so. The authors of [12] show that there are no technical restrictions for implementing the cryptographic module as an extension for common web-browsers.

Contact data is an essential part of e-mail applications. At the time of writing the most commonly used e-mail application on desktop platforms is Microsoft Outlook with a market share of 20.14% [20]. This application can be extended with so-called Add-Ins similar to browser extensions, thus facilitating the implementation of an XContact extension for this application.

B. *Mobile Platforms*

The mobile operating system iOS by Apple is similar to Android as it has a UNIX core and therefore similar (unofficial) methods to acquire privileged access. A framework similar to the Xposed Framework on Android is called "Cydia Substrate" and provides the APIs for defining hooks on methods of other applications. Installing Cydia Substrate on an Apple iOS device requires superuser privileges that can be acquired by "jailbreaking" the device. On Apple iOS contacts are managed by a central address book, providing the basic prerequisites for implementing the XContact module for iOS.

For "Windows Phone" there is no official way of acquiring privileged access and at the time of writing also no unofficial way similar to iOS or Android. It is possible to register a Windows Phone device as a developer device but this is not feasible for end-users and the acquired privileges are insufficient for implementing applications with the proposed functionality.

 ISBN: 978-1-61208-347-6

VIII. CONCLUSION

Privacy and information security in respect to distributed data management and mobile data availability are on the rise and are not solely discussed in scientific communities but have also drawn public interest.

One effective measure for protecting user data is end-to-end encryption, a well-established security paradigm already applied to a series of scenarios concerning remote data management, but especially for exchanging sensitive data between two or more participants. The major drawback of this technique is the administrative effort for creating and securely exchanging keys, thus preventing it from being widely used, e.g., for secure mail transfer.

In terms of synchronizing one's own contact data through cloud services, exchanging keys is irrelevant, for one user is administering one key to be used for various clients. A prototype implementation for end-to-end encryption of contact data for the Android platform has proven the applicability of such a security feature for mobile devices with minimal impact in terms of convenient, transparent and seamless usage. However, (still) off-the-record techniques had to be applied (the device had to be rooted) in order to practically implement this encryption model.

Nevertheless, we are convinced that developers will be able to access critical system functions of mobile platforms needed for special security operations in the near future, as there are upcoming issues concerning malware that need to be counteracted [21]. Appropriate third party partner programs, for example, might be a way to enable only licensed vendors in developing security-related software.

Independent from this vision, security issues for mobile platforms will increasingly become a topic of research [22]. The security concept presented in this paper is an attempt to protect personal contact data stored on cloud servers, the architecture of which is adequately applicable for encrypting calendar dates or private photos, and all this for various platforms.

REFERENCES

[1] V. Tchifilionova, "Security and Privacy Implications of Cloud Computing – Lost in the Cloud," in *Open Research Problems in Network Security SE - 14*, vol. 6555, J. Camenisch, V. Kisimov, and M. Dubovitskaya, Eds. Springer Berlin Heidelberg, 2011, pp. 149–158.

[2] A. Greenberg, "Cloud Computing's Stormy Side - Forbes," *Forbes*, 2008. [Online]. Available: http://www.forbes.com/2008/02/17/web-application-cloud-tech-intel-cx_ag_0219cloud.html. [Accessed: 09-May-2014].

[3] V. Strauss, "Student privacy concerns grow over 'data in a cloud,'" *The Washington Post*, 2014. [Online]. Available: http://www.washingtonpost.com/blogs/answer-sheet/wp/2014/01/03/student-privacy-concerns-grow-over-data-in-a-cloud/. [Accessed: 09-May-2014].

[4] ISACA, "Cloud Computing: Business Benefits With Security, Governance and Assurance Perspectives," 2009.

[5] K. P. N. Puttaswamy, C. Kruegel, and B. Y. Zhao, "Silverline: Toward Data Confidentiality in Storage-intensive Cloud Applications," in *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, 2011, pp. 10:1–10:13.

[6] V. P. Lijo and S. Kalady, "Cloud Computing Privacy Issues and User-Centric Solution," in *Computer Information Systems – Analysis and Technologies*, vol. 245, N. Chaki and A. Cortesi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 449–451.

[7] J. C. Muñoz, G. Tamura, N. M. Villegas, and H. A. Müller, "Surprise: User-controlled Granular Privacy and Security for Personal Data in SmarterContext," in *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research*, 2012, pp. 131–145.

[8] S. Ruj, M. Stojmenovic, and A. Nayak, "Privacy Preserving Access Control with Authentication for Securing Data in Clouds," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, 2012, pp. 556–563.

[9] A. Behl and K. Behl, "An analysis of cloud computing security issues," in *Information and Communication Technologies (WICT), 2012 World Congress on*, 2012, pp. 109–114.

[10] S. Pearson, M. C. Mont, L. Chen, and A. Reed, "End-to-End Policy-Based Encryption and Management of Data in the Cloud," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 2011, pp. 764–771.

[11] R. Karim, C. Ding, and A. Miri, "An End-to-End QoS Mapping Approach for Cloud Service Selection," in *Services (SERVICES), 2013 IEEE Ninth World Congress on*, 2013, pp. 341–348.

[12] R. Koch, D. Holzapfel, and G. Dreo Rodosek, "Data control in social networks," *2011 5th Int. Conf. Netw. Syst. Secur.*, pp. 274–279, Sep. 2011.

[13] Secomba GmbH, "Technical overview - How Boxcryptor works," 2014. [Online]. Available: https://www.boxcryptor.com/en/technical-overview. [Accessed: 09-May-2014].

[14] J. Xu, E.-C. Chang, and J. Zhou, "Weak Leakage-Resilient Client-side Deduplication of Encrypted Data in Cloud Storage Categories and Subject Descriptors," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security - ASIA CCS '13*, 2013, no. 2, pp. 195–206.

[15] A. Kapadia, "A Case (Study) For Usability in Secure Email Communication," *IEEE Secur. Priv. Mag.*, vol. 5, no. 2, pp. 80–84, Mar. 2007.

[16] BITKOM, "Mehr Sicherheit durch Verschlüsselung," *BITKOM Presseinfo Verschluesselung 19 12 2013 (German)*, 2013. [Online]. Available: http://www.bitkom.org/files/dochttp://www.bitkom.org/files/documents/Verschlsselung.jpg. [Accessed: 09-May-2014].

[17] H. C. A. van Tilborg and S. Jajodia, *Encyclopedia of Cryptography and Security*. Boston, MA: Springer US, 2011.

[18] N. P. (BRIS) Smart, "ECRYPT II Yearly Report on Algorithms and Keysizes (2011-2012)," 2012.

[19] A. Biryukov and J. Großschädl, "Cryptanalysis of the Full AES Using GPU-Like Special-Purpose Hardware," *Fundam. Informaticae*, vol. 114, no. 3–4, pp. 221–237, Aug. 2012.

[20] Campaign Monitor, "Email Client Popularity." [Online]. Available: http://www.campaignmonitor.com/resources/will-it-work/email-clients/. [Accessed: 09-May-2014].

[21] A. Gupta, S. Dutta, and V. Mangla, "Malware Attacks on Smartphones and Their Classification Based Detection," in *Contemporary Computing*, S. Aluru, S. Bandyopadhyay, U. V. Catalyurek, D. P. Dubhashi, P. H. Jones, M. Parashar, and B. Schmidt, Eds. Springer Berlin Heidelberg, 2011, pp. 242–253.

[22] M. La Polla, F. Martinelli, and D. Sgandurra, "A Survey on Security for Mobile Devices," *Commun. Surv. Tutorials, IEEE*, vol. 15, no. 1, pp. 446–471, 2013.

 ISBN: 978-1-61208-347-6