The Secure Data Container: An Approach to Harmonize Data Sharing with Information Security

Christoph Stach[†] and Bernhard Mitschang
University of Stuttgart, Institute for Parallel and Distributed Systems
Universitätsstraße 38, 70569 Stuttgart, Germany
Email: Christoph.Stach@ipvs.uni-stuttgart.de, Bernhard.Mitschang@ipvs.uni-stuttgart.de

***Abstract*—Smart devices became Marc Weiser's *Computer of the 21st Century*. Due to their versatility a lot of private data enriched by context data are stored on them. Even the health industry utilizes smart devices as portable health monitors and enablers for telediagnosis. So they represent a severe risk for information security. Yet the platform providers' countermeasures to these threats are by no means sufficient. In this paper we describe how information security can be improved. Therefore, we postulate requirements towards a secure handling of data. Based on this requirements specification, we introduce a secure data container as an extension for the Privacy Management Platform. Since a complete isolation of an app is usually not practicable, our approach also provides secure data sharing features. Finally, we evaluate our approach from a technical point of view as well as a security point of view and show its applicability in an eHealth scenario.**

***Index Terms*—smart devices; information security; data sharing.**

I. Introduction

Modern smart devices not only become more and more powerful concerning computation power, stand-by times, and built-in sensors but they are also able to interconnect with each other. Thus, each device becomes information sink and data source at the same time. By combining all available data these devices are able to draw accurate conclusions about the user's current situation, so that apps can optimally adapt their behavior to the user's needs.

However, by sharing these information with any given app on the device and other devices, this technological progress can be the source of tremendous good but also the root of potentially dreadful evil: By using smart devices for almost any purpose, contact information, calendar entries, or even banking information can be found on these devices. Due to the large volume of sensitive data generated and stored on smart devices, these devices are a prime target for hackers and data thieves. The best way to prevent data thefts is to get the users on board by winning their awareness for the threats. However, none of the prevailing smart device platforms provides an adequate solution for this problem and the user becomes apathetic for security issues [1].

Hence, Wang et al. [2] claim the separation of sensitive from nonsensitive data and a smart encryption of the data in terms of acceptable battery consumption. We examine how these two goals can be accomplished for the data storage within an eHealth scenario, i. e., a use-case where a lot of really sensitive data accrues [3]. In addition, a solution for this information security problem also has to enable interoperability in terms of regulated inter-app data sharing [4]. Our approach is based on Android, yet the results are applicable to any application platform. In this paper, we discuss the following three issues:

(I) Since information security is an important and heavily demanded yet not technically realized property, we postulate a requirements catalog towards a data storage system. For this purpose, we deduce 7 protective goals from literature and discuss, how these goals can be archived.

(II) We analyze how data storage as well as inter-app data sharing can be realized in an Android app. In addition to this standard Android approach, we introduce novel approaches, realized as extensions for the **P**rivacy **M**anagement **P**latform (**PMP**) [5], [6]. The initial approach implements a simple PMP data container that enables the user to share data with other apps. In a second step, we enhance this basic container by encryption features, fine-grained sharing techniques, and data distribution. The result is called **S**ecured **D**ata **C**ontainer (**SDC**). Moreover, by identifying different users the SDC depicts a reliable protection against unwanted physical access to the smart device. The fundamental data model is essentially generic but can be tailored to any use-case.

(III) Finally, we evaluate the approaches in a benchmark tailored to an eHealth scenario. We are interested in performance metrics and the impact on information security.

The remainder of this paper is as follows: In Section II we discuss key protective goals concerning information security. The data storage approaches are introduced in Section III and evaluated from a technical point of view and a security point of view in Section IV. Finally, we give an overview of information security approaches for mobile platforms in Section V, before Section VI sums up this paper.

II. Protective Goals

Since apps often deal with sensitive data, there is a high degree of information security required. Data has to be protected against malware and the user has to be able to control how data is processed and with whom it is shared. While the legal

[†] This work was supported by a Google Research Award.

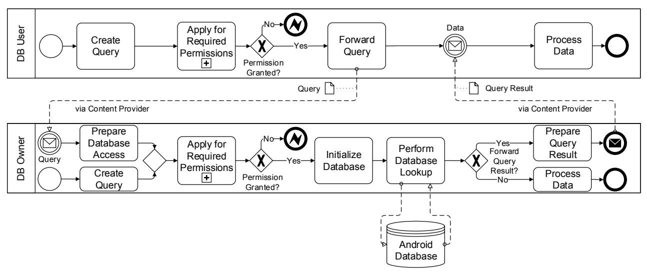

Figure 1. Data Access in the Android Data Container Approach

guidelines are pretty explicit, a general technical realization is still missing. Dhillon and Backhouse [7] identify the three traditional information security principals: **Confidentiality** ensures that only authorized entities get access to information. **Integrity** ensures that the information cannot be compromised by third-parties. **Availability** ensures that the information is accessible at anytime.

However, they point out that these three principles are not sufficient. Cherdantseva and Hilton [8] add four more principles which are especially relevant in the domain of mobile systems: **Auditability** ensures that any information access is monitored. **Authenticity** verifies the identity of an entity requesting access to information. **Non-repudiation** attests unchallengeable whether an entity processed the information. **Privacy** gives users full control over their data.

After identifying the relevant information security principals, we introduce the overall architecture of our approach (see Section III-B and Section III-C) and verify whether it complies with these principals (see Section IV-B).

III. Data Storage Approaches

In this section, we describe three different approaches how apps can store, manage, and share their data, namely a native Android database approach, a PMP Resource implementing a simple data container, and the Secured Data Container.

A. Databases in Android

Basically, every Android app has its own private storage area for its data. In this private storage SQLite databases can be created. However, each app has to reimplement the database integration from scratch. Only the owner of the database—that is the app which created the database—is able to use the database. In contrast to other mobile OS, Android enables app developers to share the stored data with other apps. To this end, the database owner has to implement a Content Provider that defines an URI via which other apps can access the data. The Content Provider has an interface which processes SQL-like statements. An individual Content Provider is required for each table which should be accessible for other apps. Yet, the data owner cannot restrict the access to the data for certain apps. Subsequently, any app knowing the URI can use the Content Provider and therefore gets reading and writing access to the corresponding data.

Figure 1 depicts the procedure of how data can be shared in this approach. The user formulates a query and applies for all permissions which are required in order to access the database. If the permission is not granted, the calling app crashes since in Android all permissions have to be granted when the app is installed and a subsequent withdrawal of permissions is not allowed. Thus usually the permission is granted and the query is sent to the URI, i. e., the database owner. The owner then prepares the database, which means the owner also needs to request for permissions (e. g., to read and write the storage content). Again, it can be assumed that the permissions is granted. Then the database is initialized in order to perform the lookup. The results are automatically sent back to the inquiring app for further processing.

Apart from performance and information security issues (see Section IV) this approach requires a lot of Content Providers and thus URIs which have to be known by any involved app. Moreover, the approach requires adaptations in every app which uses the Content Providers when changes are made to the owner app—not to mention that all of these apps can be used only as long as the owner app is installed.

B. PMP Data Container Resource

The PMP is an intermediate layer between apps and the OS (in the context of this work the PMP can be seen as a part of the OS). In the following, we give a very brief overview of the PMP. For more information please refer to the literature [5], [6], [9], [10]. With the PMP the user is able to specify fine-grained and context-aware privacy policy rules. The underlying privacy policy model is based on the idea that an *app* requires its permissions not all the time, but only for certain features. These so-called *Service Features* cannot access private data directly but have to query information brokers, the so-called *Resources*. The user defines which Service Feature gets data by which Resource.

A simple PMP-based data container enhances the data sharing ability and reusability of Android-based data bases. To this end, it provides a database accessible via an interface similar to the SQLite interface. The data model of the database is generic (key-value pairs) in order to be applicable to any app and can be tailored to any use-case. A Resource can be used by an arbitrary number of apps and an app can use an arbitrary number of Resources. Therefore several of these data container Resources can coexist each with a specialized data model for a certain kind of data. The user specifies via the PMP which app is allowed to read from and write to which data container.

The data access procedure is as follows: Initially, each app registers to the PMP at installation time. Thereafter the app is able to request data from Resources. The PMP verifies whether this request is conform to the policy. If this is not the case, the user gets a feedback message telling him or her which app requests data from which Resource and s/he decides how to deal with the request. If the access is allowed the PMP forwards the query to the Resource where it is executed. The result is sent back to the inquiring app.

This approach provides several advantages over the Android-based approach. The **data sharing among apps is simplified**, since the data ownership is transferred to the Resource and any app can access it via the PMP. The **reusability** is high, since any app can use the Resource out of the box, i. e., the app developer is no longer responsible for the implementation of databases and Content Providers. Additionally, a **use-case driven tailoring** of the basic data model is feasible and variants of the data container can be operated in parallel. This is also a step towards **data separation**, since for each variant different policy rules can be applied. The applicability of the PMP policy to the data container is also a huge step towards **information security**.

C. The Secure Data Container

Based on this initial approach, the SDC focuses on information security features. Therefore, the SDC completely encrypts its database. Any data is only decrypted as long as necessary. The key is randomly generated and only known to the SDC. Thus, the data is secured against data thefts. Access privileges can be applied to single data tuples. To achieve this, the SDC generates two maintenance tables, one storing all registered app ids and one specifying which app is allowed to access which data tuple. The SDC hosts several database instances and the stored data is divided among theses partitions. Thereby, only the relevant parts of the data container have to be encrypted and decrypted. As a smart device can be used by several people, the SDC introduces a user identification feature. This provides not only a better protection for the stored data, but also enables a secured sharing of data not only among apps but even among users.

The data access procedure is shown in Figure 2: The initial steps are similar to the access procedure of the simple approach. As soon as the PMP allows the access to the SDC, the two procedures differ, since the PMP adds conditions to the query before forwarding it to the SDC. These conditions comprise lookups in the maintenance tables to ensure that only data tuples are processed which are shared with the inquiring app. Then, the SDC requests the user to authenticate via the PMP—no login data is shared with the app. The data partitions containing the relevant data are decrypted, the SDC executes the modified query, and the partitions are encrypted again. Subsequently, the modified query result—i. e., only data which is shared with the inquiring app—is sent back.

In addition to the advantages of the basic PMP data container approach, the SDC provides advantages concerning **information security**. The **data encryption** of the database is necessary in order to prevent information leakage as well as unauthorized modifications [11]. Due to the **smart encryption** (i. e., the data distribution and partition-based encryption) its performance is very promising, as the evaluation results show (see Section IV-A). Apart from the reduction of encryption costs, thereby a distinctive **separation of sensitive from nonsensitive data** is realizable.

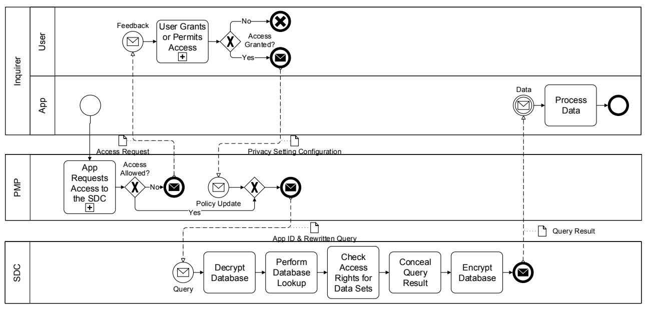

Figure 2. Data Access in the SDC Approach (Simplified)

IV. Evaluation

In the following, we compare a conventional Android database with a PMP-based data container and the Secure Data Container. For our evaluation we consider technical issues as well as security issues.

A. Technical Analysis

To carry out our measurements, we used a Motorola Moto E Phone (2nd generation) which represents a lower middle class Smartphone of the current generation. Thereby, the evaluation results are representative for real world purposes.

The benchmark consisted of two separated steps: A writing and a reading step. Both steps were executed with two datasets, differing in their size. The small dataset (labeled as S) consisted of 500 different health records while the big dataset (labeled as B) consisted of $16,000$ health records. We chose these sizes based on McObject's TestIndex benchmark for Android[1]. The model of a health record is shown in Figure 3. It consists of patient data, health values, sensor data, and answers to a questionnaire. For the SDC, the five tables of the star schema were stored in independent partitions.

Initially, the writing benchmark generated 500 or $16,000$ random health records and stored them in the data store which was evaluated, namely an Android-based database (*Android*), a PMP-based data container (*PMP*), and the Secure Data Container (*SDC*). For the SDC we used an AES-256 encryption. Subsequently, after the writing benchmark the reading benchmark accessed each health records in a random order. Both benchmarks were repeated for 10 times. After each run the database was deleted and the cache was cleared in order to prevent influences by warm caches. The benchmark results are shown in Table I. The values are the medians of the ten runs in order to eliminate outliers.

The benchmark results provide the following findings: (**T1**) Concerning the **overall runtime** of the writing benchmark, it is almost indifferent which approach is used. The SDC's AES-256 encryption causes a minimal overhead of approximately 7%. This means, the processing of a single dataset is prolonged by just about 5 milliseconds. When looking at the results of the reading benchmark, the shared databases of the PMP Resource and the SDC demonstrate their strengths. The use of Content

[1]see http://www.mcobject.com/android

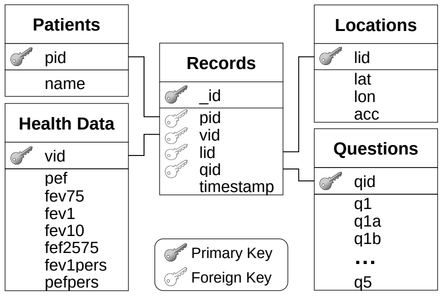

Figure 3. Exemplary Star Schema of a Health Record

Providers is not only insecure (e. g., the data owner—i. e., the app offering the Content Providers—can hardly restrict the set of accessing apps [12]) but also very costly. The client app has to contact the app that owns the data. The therein specified Content Provider automatically handles the inter-process communication. The readings show, that the PMP with its shared database outperforms Android's Content Provider approach by more than 50 %. Despite the cost for the encryption, the SDC outperforms the PMP, as the data is partitioned in several databases whereby the access cost is reduced. In summary it can be said, that the bigger the dataset gets, the more the encryption costs can be neglected.

(**T2**) The **average CPU usage** is similar on all approaches for the writing benchmark (all processes associated with the tests are taken into account). For the reading benchmark, the costs for the Android Content Providers and the shared PMP-based data container are at a similar level with a slight advantage for the PMP. The SDC encrypts and decrypts the database for each access. Thanks to the data distribution strategy of the SDC the amount of data which has to be encrypted is reasonable. As a result, the CPU workload is reduced compared to the high amount of inter-process communication used by the Content Providers.

(**T3**) The **total battery drain** is correlated with the runtime of an app and its accruing CPU workload. Thus, the three approaches are dead even in this category for the writing benchmark while for the reading benchmark the Android approach is clearly inferior. The SDC's battery drain is considerably less than the half of the Android's battery drain. This means an app using the SDC not only benefits from its enhanced security features but also saves energy.

(**T4**) The **maximum memory usage** considers only the peaks that are reached while running the benchmarks. We regard *private dirty usage* which counts the amount of RAM inside the process that is not shared with any other processes. For the PMP and the SDC the cumulative costs are taken into account, i. e., the total memory usage of the benchmark app, the PMP, and the required Resources. Overall the results for the approaches are pretty close together with a slight advantage for Android in the reading benchmark. However, a single PMP instance can manage any number of apps in parallel. So the memory usage of the managing tasks of the PMP (which is more than 50 % of the total memory usage) is divided equally among all apps using the PMP. Thus, these numbers would be in favor for the PMP-based approaches (including the SDC) if several instances run in parallel.

(**T5**) By using the SDC the **database size** increases due to the required auxiliary tables for maintenance and the additional columns in the data table. This administrative overhead for the SDC is approximately 70 % for the small dataset and almost 100 % for big dataset due to its data distribution and the required meta data (e. g., authorization information, access rights, or indices). Compared to the performance gain and the security features, this is indeed a small price, especially since the storage space of smart devices permanently increases with each new model.

B. Security Analysis

A security analysis based on the 7 protective goals is done in the following. The findings are summarized in Table II.

(**S1**) The **confidentiality** is breached when an app gets access to data which is not intended for it. Android ensures this by separating apps from the underlying OS layers which manage the data. However, when using Content Providers for data sharing, any app knowing the right URI gets full access to the provided data. By using the PMP, the user controls which app gets access to the shared data. With the SDC the data is encrypted and as a consequence the confidentiality is even guaranteed when there is a breach in the security mechanisms of the OS. Moreover, the SDC can even define access rules for each database entry and it fully guarantees confidentiality by adding user accounts, i. e., it can define access rules for different apps and users.

(**S2**) Due to the strictly separated OS layers, the **integrity** is guaranteed in Android (and therewith in the PMP) as long as there is no security breach. By encryption the SDC ensures that the stored data cannot be compromised by such an attack.

(**S3**) **Availability** is guaranteed since all approaches base on mature database technologies (e. g., transactional features and recovery techniques). However, the Android reading app requires certain Content Providers which are only available as long as the writing app is installed on the device.

(**S4**) Android checks which permissions an app has. However, this cannot document which actions an app actually performs. The PMP guarantees **auditability** by logging any access to PMP Resources. With the encryption, the SDC even prevents illegal data access and therewith guarantees that the PMP's log is complete.

(**S5**) Android only checks whether an app has all permissions required for its actions. The PMP and the SDC identify and register all apps. The SDC even identifies the users. Thereby the **authenticity** is fully guaranteed.

(**S6**) For the **non-repudiation** applies the same predicates as for the auditability (F4). However, since Android does not track which actions an app actually executes, the non-repudiation is not guaranteed by Android at all.

(**S7**) Everybody has a different idea of the right degree of **privacy**. Thereby, it is inevitably that the user is in full control over any data which is shared with an app. Android simply

Table I. Benchmark Results

		Writing Benchmark			**Reading Benchmark**		
		Android	**PMP**	**SDC**	**Android**	**PMP**	**SDC**
Runtime (in sec)	**S**	34.5	35.8	38.4	62.0	30.2	29.3
	B	1,051.5	1,095.1	1,121.2	2,011.6	972.7	936.3
CPU Workload (in %)	**S**	32.9	31.0	30.6	34.4	31.9	18.2
	B	35.2	32.7	32.6	34.9	29.9	17.9
Battery Drain (in mAh)	**S**	2.55	2.87	2.80	4.59	2.41	2.06
	B	78.19	86.62	80.84	160.76	78.35	70.49
Memory Usage (in MB)	**S**	20.90	16.05	16.81	16.84	16.96	17.64
	B	23.36	18.45	21.20	16.87	19.57	18.03
DB Size (in kB)	**S**	120	120	204	120	120	204
	B	2,444	2,444	4,800	2,444	2,444	4,800

informs him or her which permissions are requested by an app and s/he has to accepts all of them. In the PMP, the user is able to restrict any data access on a fine-grained level. In addition to it, s/he can control the data access for each database entry individually with the SDC.

From a *technical point of view*, the use of the SDC is not only suitable when an app processes confidential data, but also when some datasets have to be shared with other apps. In this case, a tremendous advantage in terms of runtime, CPU workload, battery drain, and memory consumption is generated. In addition, less CPU-intensive cryptographic algorithms for data which requires less protection—i. e., by introducing different protection levels—could improve its performance even further. However, the additional computational effort for the encrypting and decrypting of the database is also reduced for every new smart device generation as earlier evaluations results with a NexusOne attest. From a *security point of view*, the SDC fulfills all of the requirements towards a secure data container. Moreover, the usage of the SDC is virtually transparent to the user, i. e., s/he is not overburdened by it.

V. Related Work

There is currently no work fully comparable to the SDC in terms of regulated inter-app data sharing for smart devices. Several approaches cover certain aspects of our work:

Unlike other mobile OS, Android already supports inter-app data sharing via its Content Providers. As soon as an app defines a Content Provider for its data, any other app can use the Content Provider's SQL-like interface to access the data. However, the data owner is able to regulate the data access only in terms of granting reading and writing permissions which are applied for any app [13]. The *TISSA* system [14] enables the user to decide for any app using Content Providers individually, how to deal with its data requests, i. e., whether the data is returned accurately, anonymized, falsified, or empty. Nevertheless, these settings apply to any data provided by the source app and cannot be limited to certain datasets. Moreover, data encryption is not applicable to this approach.

The encryption of the home partition is introduced in Android 4.0 and extended to full disc encryption in Android 5.0. However, this concept secures the data only until the user unlocks the smart device; afterwards, all data is unencrypted. This protects against attacks on a locked device but not against attacks during operation [15]. Each app has to secure its data during operation individually, e. g., by using third-party libraries such as *SQLCipher* [16]. This limits the interoperability significantly and if the data is shared with an app which does not provide a sufficient data encryption, the information security is at stake.

Concerning the separation of sensitive from nonsensitive data, *Samsung KNOX* [17] introduces a secured and an unsecured domain on Android-based devices. In the secured domain only accredited apps can be executed and their data must not be shared with any unaccredited app. However, only selected apps benefit from this protection and also the interoperability suffers from the strict data separation.

A lot of work is done dealing with privacy issues by introducing enhanced permission systems similar to the PMP. Backes et al. [18] introduce such a system for Android called *AppGuard*. AppGuard implants a security monitor into apps which supervises every activity. When an operation defies AppGuard's policy it is not executed and is replaced by alternative code. Yet, in-vivo bytecode instrumentation is dangerous since it can have unforeseen side effects, e. g., app crashes. Moreover, allowing a user to manipulate the bytecode of any app constitutes a copyright violation. *Aurasium* [19] follows a different strategy, as it does not manipulate the app but its runtime environment by introducing secure sandboxes. These sandboxes monitor every app and interfere into the program flow if necessary. Conti et al. [20] add context-sensitive policy rules to the current Android permission system. I. e., in *CRêPE* the user can define a situation under which a certain rule should be applied (e. g., the phone should only ring after work). Another inline reference monitoring approach is *Dr. Android & Mr. Hide* [21] which additionally extends the Android permission management by context-aware and fine-grained rules. As Android is a single user OS, its permission system is not designed for different user. Whoever gets access to an unlocked device has full control over the stored data.

Table II. Feature Comparison (the filling degree of the circle indicates how well the respective feature is realized)

	Android	**PMP**	**SDC**
Confidentiality	◔	◑	●
Integrity	◔	◔	●
Availability	◑	●	●
Auditability	◔	◑	●
Authenticity	○	◕	●
Non-repudiation	○	◑	●
Privacy	○	◑	●

Rohrer et al. [22] introduced *DR BACA*, which adds user roles to permissions. Thus, a single device can be used by different users with individual permissions. Yet, all of these approaches aim for preexisting data providers and not for the restriction of inter-app data sharing.

A different approach to address the problem of information security is introduced by Neisse et al. [23]. Their framework enforces usage control even for data which has already left the smart device. However such an approach is out of this work's scope. Nevertheless, an integration of such a framework in the PMP and the SDC is worth further researches.

VI. Conclusion

Alarmed by an increasing number of data thefts, smart device users are becoming more and more aware of the need for comprehensive information security mechanisms which do not constrain the apps' usability. The mobile platform vendors face these threats with non-transparent app checks and elaborated permission systems. Thus, the users are either totally incapacitated or completely overwhelmed. Both preventing a satisfying protection for sensitive data.

For this reason we pursue three key objectives with our work: (I) We postulate a requirements catalog towards a data storage system containing 7 vital protective goals. (II) As these goals are not covered by the existing information security measures, we introduce the SDC as extensions for the PMP. (III) Finally, we show that the SDC provides more than acceptable results with respect to its performance and security features. Although the presented prototypes of the PMP and the SDC are implemented for Android, the underlying concept can be applied to any application platform such as the *Facebook Platform* or *Chrome OS*.

Acknowledgments

The PMP results from a close collaboration with Google Munich office. Hence, we would like to thank Google for their support and their suggestions for improvements.

This work was partially funded by the PATRON project of Baden-Württemberg Stiftung gGmbH.

References

[1] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android Permissions: User Attention, Comprehension, and Behavior," in *SOUPS '12*, 2012.

[2] Y. Wang, K. Streff, and S. Raman, "Smartphone Security Challenges," *Computer*, vol. 45, no. 12, pp. 52–58, 2012.

[3] R. S. H. Istepanian, S. Laxminarayan, and C. S. Pattichis, Eds., *M-Health — Emerging Mobile Health Systems*. Springer US, 2006.

[4] M. Chan, D. Estève, J.-Y. Fourniols, C. Escriba, and E. Campo, "Smart Wearable Systems: Current Status and Future Challenges," *Artif. Intell. Med.*, vol. 56, no. 3, pp. 137–156, 2012.

[5] C. Stach and B. Mitschang, "Privacy Management for Mobile Platforms — A Review of Concepts and Approaches," in *MDM '13*, 2013.

[6] C. Stach, "How to Assure Privacy on Android Phones and Devices?" In *MDM '13*, 2013.

[7] G. Dhillon and J. Backhouse, "Technical Opinion: Information System Security Management in the New Millennium," *Commun. ACM*, vol. 43, no. 7, pp. 125–128, 2000.

[8] Y. Cherdantseva and J. Hilton, "A Reference Model of Information Assurance & Security," in *ARES '13*, 2013.

[9] C. Stach and B. Mitschang, "Design and Implementation of the Privacy Management Platform," in *MDM '14*, 2014.

[10] C. Stach, "How to Deal with Third Party Apps in a Privacy System — The PMP Gatekeeper," in *MDM '15*, 2015.

[11] E. Shmueli, R. Vaisenberg, Y. Elovici, and C. Glezer, "Database Encryption: An Overview of Contemporary Challenges and Design Considerations," *ACM SIGMOD Record*, vol. 38, no. 3, pp. 29–34, 2010.

[12] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner, "Analyzing Inter-application Communication in Android," in *MobiSys '11*, 2011.

[13] W. Enck, M. Ongtang, and P. McDaniel, "Understanding Android Security," *IEEE Security and Privacy*, vol. 7, no. 1, pp. 50–57, 2009.

[14] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, "Taming Information-stealing Smartphone Applications (on Android)," in *TRUST '11*, 2011.

[15] J. Götzfried and T. Müller, "Analysing Android's Full Disk Encryption Feature," *JoWUA*, vol. 5, no. 1, pp. 84–100, 2014.

[16] Zetetic LLC, *Full Database Encryption for SQLite*, 2015. [Online]. Available: https://www.zetetic.net/sqlcipher/.

[17] Samsung Electronics, "Meet evolving enterprise mobility challenges with Samsung KNOX," Samsung Electronics, White Paper, 2014.

[18] M. Backes, S. Gerling, C. Hammer, M. Maffei, and P. von Styp-Rekowsky, "AppGuard - Enforcing User Requirements on Android Apps," in *TACAS '13*, 2013.

[19] R. Xu, H. Saïdi, and R. Anderson, "Aurasium: Practical Policy Enforcement for Android Applications," in *Security '12*, 2012.

[20] M. Conti, B. Crispo, E. Fernandes, and Y. Zhauniarovich, "CRêPE: A System for Enforcing Fine-Grained Context-Related Policies on Android," *Trans. Info. For. Sec.*, vol. 7, no. 5, pp. 1426–1438, 2012.

[21] J. Jeon, K. K. Micinski, J. A. Vaughan, A. Fogel, N. Reddy, J. S. Foster, and T. Millstein, "Dr. Android and Mr. Hide: Fine-grained Permissions in Android Applications," in *SPSM '12*, 2012.

[22] F. Rohrer, Y. Zhang, L. Chitkushev, and T. Zlateva, "DR BACA: Dynamic Role Based Access Control for Android," in *ACSAC '13*, 2013.

[23] R. Neisse, A. Pretschner, and V. Di Giacomo, "A Trustworthy Usage Control Enforcement Framework," *Int. J. Mob. Comput. Multimed. Commun.*, vol. 5, no. 3, pp. 34–49, 2013.