Cryptographic Service Providers in Current Device Landscapes: An Inconvenient Truth

Florian Reimair[1], Johannes Feichtner[2], Dominik Ziegler[2], Sandra Kreuzhuber[3] and Thomas Zefferer[4]
[1]*Secure Information Technology Center - Austria (A-SIT), Graz, Austria*
[2]*Institute for Applied Information Processing and Communications, Graz University of Technology, Austria*
[3]*PrimeSign GmbH, Graz, Austria*
[4]*A-SIT Plus GmbH, Vienna, Austria*
florian.reimair@a-sit.at, {johannes.feichtner, dominik.ziegler}@iaik.tugraz.at, sandra.kreuzhuber@prime-sign.com, thomas.zefferer@a-sit.at

Keywords: Applied Cryptography, Cryptographic Service Providers, Cloud.

Abstract: Current application and device landscapes became a harsh environment for data security. Multi-device users enjoy the convenience and efficiency of modern distributed applications in a highly heterogeneous device landscape. However, today's data protection mechanisms fell behind in taking care of some current use cases and application scenarios. We perform a case study and an in-depth security analysis and risk assessment on a simplified set of three different cryptographic service provider types; software, hardware, and remote. Our case study shows that different provider types can change application characteristics considerably. Our security analysis and risk assessment shows how different provider types can influence the security properties of a set of use cases. We found that no one provider can excel for every cryptographic task. Based on these findings we formulate a list of features which we believe are crucial to get the data protection mechanisms up to speed again so that everyone can again benefit from data security even in a world of highly distributed applications and data.

1 INTRODUCTION

The current heterogeneous device landscape became a harsh environment for privacy and data security. Applications need to run on a variety of different devices, data is stored in the cloud, and dynamic distributed computing in personal area networks is within reach (Reiter and Zefferer, 2015). The need for data security i.e. integrity, confidentiality, availability of data becomes more apparent as the attack surface grows with the number of devices involved.

However, today's data security does not scale well with the current evolution of computing environments. Personal mobile devices (smart phones, tablets, smart watches, etc.) have limited resources in terms of computing power and battery life, browsers hardly have access to hardware-assisted key storage for handling sensitive data, and, among others, implementations of the required cryptographic methods may not be available for the device at hand. Beside the limitations brought up by devices, applications and use cases have evolved as well. Examples are having data available on every device of a modern (multi-device) user. Cryptography is required on almost every device in almost every situation, even when cryptographic keys cannot be appropriately protected by the device at hand.

Industry and researchers already took on the challenge of tackling shortcomings in protecting data in current device and application environments (Reimair, 2014). Different cryptographic APIs exist, for local or remote use, having different ways of persisting cryptographic keys. Authentication needs of remote APIs are often handled in a nonstandard manner and hence, hinder flexibility and interoperability. Hardware tokens are available which on the one hand can reasonably protect cryptographic primitives, but, on the other hand, often rely on interfaces that are not available on every device. These approaches and solutions mostly only target narrow use cases, may only work safely given a set of prerequisites, and/or do not even use cryptography in order to reach their goals.

In this work, we elaborate on the challenges and issues raised against cryptographic service providers by current applications. First, we reduce available technology to three types of cryptographic ser-

Reimair, F., Feichtner, J., Ziegler, D., Kreuzhuber, S. and Zefferer, T.
Cryptographic Service Providers in Current Device Landscapes: An Inconvenient Truth.
In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT*, pages 367-374
ISBN: 978-989-758-259-2
Copyright © 2017 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

vice providers namely software providers, hardware providers and remote cryptographic service providers. We then discuss how different provider types can change the characteristics of a single application. We highlight that current application environments cannot guarantee a certain type of cryptographic service provider and an application might change its characteristics without the developer's or the user's consent. Last but not least, we provide an in-depth security analysis and risk assessment of the three provider types against the use cases specified for the W3C Web Cryptography API (Halpin, 2014a).

We found that in today's heterogeneous device landscape none of the evaluated cryptographic service provider types can excel in every use case. We list features, shortcomings and issues with cryptographic service provider types available today and discuss candidates which, after receiving some work, might be capable of bringing data security back to the modern user. We conclude that applications and devices need means of using the most suitable cryptographic service provider type and implementation for a task at hand. Only then can applied cryptography catch and keep up with the fast evolution of applications and device environments in terms of data security.

The remainder of his work is structured as follows. Section 2 starts by discussing related work. Section 3 reduces existing solutions to three different cryptographic service provider types, discusses the capabilities and features of each type and how they can influence the characteristics of an application. Section 4 evaluate the representative set of three different provider concepts after their impact on the security of modern use cases. Section 5 discusses missing features, candidates and the ultimate goal. Section 6 concludes the work.

2 RELATED WORK

Although the topic at hand seems to be more important then ever, not much has been done to address the struggle of cryptographic service providers in current applications. There are, however, evaluations and comparisons of specific platforms or specific types of services and work has also been done on the topic of usability.

A prevalent example of device-specific evaluations are mobile devices. Researchers have created powerful frameworks for analysing mobile applications in order to expose their implementation flaws or their malicious purpose (Baumgärtner et al., 2015; Backes et al., 2016). Studies about misuse of cryptography by design have been created (Egele et al., 2013), mobile operating systems have been scanned for features and shortcomings in providing security to the user (Mohamed and Patel, 2015). Other work presents concepts of leveraging hardware-assisted security modules (Santos et al., 2014).

Cloud security is a well-covered topic as well. Surveys list issues and solutions (Fernandes et al., 2014; Ren et al., 2012) while innovative uses of cloud environments for better security are proposed as well (Varadharajan and Tupakula, 2014). Furthermore, frameworks have been created to foster secure cloud computing (Chang et al., 2016; Chang and Ramachandran, 2016).

Work has also been done in less popular areas. Hardware security modules (HSMs) have been compared and evaluated (Reimair, 2011), cloud signature services have been discussed (Reimair, 2014). Even topics like usability of cryptography (Gutmann and Grigg, 2005) and studies about why cryptographic software fails (Lazar et al., 2014) have been created.

However, to the best of our knowledge, no comparison of cryptographic service provider types with respect to security and applicability for certain use cases has been done so far. We believe that only so we can address the struggle of cryptographic service providers in current applications.

3 CRYPTOGRAPHIC SERVICE PROVIDERS ARE DIVERSE

In this section, we discuss the capabilities and features of cryptographic service providers and how they can influence the characteristics of an application. First, we give a basic definition of what a cryptographic service provider is in this context. Next, we classify providers into three different types in order to evaluate and compare them. Based on this simplification, we discuss on how one application can alter its characteristics according to the cryptographic service provider being used.

3.1 Definition

A cryptographic service provider provides a ready to use implementation of cryptographic methodologies i.e. algorithms, protocols, etc. Some providers can do only basic cryptographic operations like encrypting and decrypting data, others provide protocol implementations and/or can generate cryptographic keys, some can persist cryptographic keys.

Cryptographic service providers come in different shapes and security levels. FIPS140-2 (National Institute of Standards and Technology, 2001) provides

some basic reference. Providers can be software-only i.e. they are a piece of software, a library, that provides cryptographic methods. Hardware providers, also known as hardware security modules (HSMs), are, in general, distinct device which interfaces with PCs or alike via common interfaces like USB or Ethernet. Some HSMs feature strong protection against attacks like erasing the persisted cryptographic keys when under physical attack. The level of security a cryptographic service provider can achieve depends on how easy it is to retrieve cryptographic keys, inject faults, or abuse keys.

3.2 Three Types

For this work, we consider three different types of cryptographic service providers – software, hardware, and remote providers. Software and hardware providers represent traditional implementations and are aligned with the FIPS140-2 standard.

Software-only providers represent the cheapest but also least secure variant of providers. They are broadly available, are easy to deploy and easy to use. *hardware-assisted* providers are capable of providing significantly more security, are traditionally much more expensive and require more efforts to setup and operate. For our work, we consider hardware providers to offer the highest level of security. Additionally, we add *remote* cryptographic service providers to the set. We consider remote cryptographic service providers to be cryptographic service providers which can be accessed via the Internet. Consequently, these providers can be used by multiple devices, therefore allow for a sort of key mobility and key sharing and therefore, enable new use cases especially in current application landscapes. For this work we assume that remote providers offer the highest security level and are under the sole control of the user.

3.3 Showcase

Our showcase application offers basic cryptographic functionality, i. e. encryption and decryption. A user can encrypt or decrypt data using standardised encryption schemes. After a successful encryption or decryption process, the application allows for storing the encrypted/decrypted data in the cloud.

An architectural overview of the application is given in Figure 1. The *user* interacts with the *application*. The *application* code handles the *data* by loading and storing it. Furthermore, the *application* makes use of a *cryptographic service provider* to actually encrypt/decrypt the *data*.

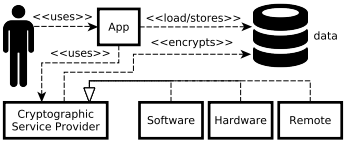

Figure 1: Demonstrator Architecture.

The application is therefore able to perform three use cases by simply changing the cryptography provider. First, the application succeeds in doing encrypted local and remote storage in a single-device and single-user environment (Figure 2). The user

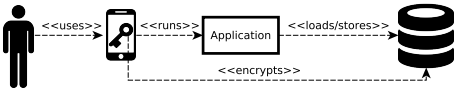

Figure 2: Use case: local key.

launches the application on his smart phone and is able to encrypt the data prior to storing it to disk or to a cloud storage provider. The user can use the hardware cryptographic provider of his modern smart phone. The same use case can be accomplished with a software provider yielding a lower security level.

Next, the application succeeds in adding multiple devices to the single-user use case discussed above (Figure 3). Without changing the application, the application enables a cloud storage use case where the user can read and modify data from every device of his while maintaining data security. That feature becomes available by only enable the user to select a remote cryptographic service provider.

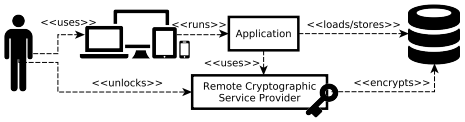

Figure 3: Use case: remote key.

And last but not least, the application can utilise key sharing capabilities of a remote cryptography service provider to enable the use case of data sharing for multiple users and multiple devices while keeping the data secure and confidential.

The case study indicates that different cryptographic service providers do change the characteristics of an application considerably. Therefore, interchangeable cryptographic service providers can lead to new features and more secure computing.

3.4 Headache!

What seems like a feature swiftly becomes bad news as soon as we look at the heterogeneity of current

devices and their provider implementations. In this section we discuss two examples of how unforeseen provider types can compromise security.

Java's Cryptography Extension (JCE) features a provider factory where the developer can either pick a provider or let the factory choose one depending on system configuration. The issue therefore already existed back then. However, hardware providers for PCs and servers need special treatment and corresponding applications anyhow pick their providers accordingly. Furthermore, current use cases outgrew the interfaces provided by the JCE and therefore, applications had to be customised for special providers and their special capabilities.

Current applications, however, face a highly heterogeneous device landscape where the issue becomes very real. A representing example is the W3C Web Cryptography API for browsers (Halpin, 2014a; Halpin, 2014b). The issue of unforeseen cryptographic service provider types/implementations in web applications is very visible because current web applications run on a multitude of different devices and browsers. Different devices likely have different cryptographic service providers available and a sensitive cryptographic key might be properly protected on one device but not on another.

All in all, each security provider implementation has its unique features. In the past this did hardly pose a problem as users mainly used a single PC and servers had their physical place in the world. In the modern world, however, the device landscape grew more heterogeneous and users started to use a multitude of devices. We believe that today one can hardly anticipate which security provider implementation will be available on the deployment site.

4 SECURITY ANALYSIS AND RISK ASSESSMENT

The discussion of the aforementioned representative case study showed that – albeit using the same cryptographic methodologies – different implementation concepts (i. e. software, hardware, and remote) cause vastly different application characteristics in terms of data security but also achievable functionality. Inspired by these findings, we implemented and evaluated the representative set of three different provider concepts after their impact on the security of an application.

We will derive assets and respective operations from the use cases defined by the Web Cryptography API. Next, we define threats against these assets. We then discuss residual risks. Some general assumptions limit the scope of the evaluation. We then draw conclusions on the results of our security analysis.

4.1 Demonstrator Implementation

Our proof-of-concept web application utilises W3C's Web Cryptography API. The enclosed provider factory architecture is realised as a JavaScript library and offers three providers which conform to W3C's Web Cryptography API implemented using the JavaScript technology as well. These providers are the browser implementation itself (representing software cryptographic service providers), a plugin for Apache Cordova[1] (representing hardware providers), and the CrySIL infrastructure (Reimair et al., 2016) (representing remote providers).

4.2 Assumptions

We assume, that the provider architecture code is correct and correctly delivered. The same applies to the provider implementations. Client platforms are considered as honest but curious.

4.3 Use cases

The W3C has carefully selected a set of cryptographic operations and algorithms that should be supported by implementations of the Web Cryptography API. These operations build around use cases found in web apps:

Multi-factor Authentication (UC1). Web applications may require the user to prove that she has access to some cryptographic primitive. Usually, a challenge has to be decrypted during authentication followed by signing the decrypted challenge.

Protected Document Exchange (UC2). Web applications that enable users to exchange data may offer mechanisms for ensuring that only legitimate receivers gain access to the data. Wrapping content encryption keys using the public key of the receiver (a. k. a. hybrid encryption) is often used.

Cloud Storage (UC3). An increasing number of users and services store their data within the cloud. In order to protect the confidentiality of their data, web applications require methods for encrypting data.

Document Signing (UC4). Web applications may allow users to electronically sign a document in order to express consent.

Data Integrity Protection (UC5). New storage mechanisms introduced in HTML5 allow web applications to cache data locally for later use. Therefore,

[1] https://cordova.apache.org

web applications may require mechanisms for ensuring that data has not been modified.

Secure Messaging (UC6). The increasing use of technologies that enable direct browser-to-browser communication yields to the need of protect exchanged messages. Therefore, web applications require methods for negotiating shared encryption keys and means for detecting modifications on the transmitted data.

JavaScript Object Signing and Encryption (UC7). The IETF JavaScript Object Signing and Encryption (JOSE) Working Group provides specifications for signing and encrypting data using the JSON data structure.

4.4 Assets and Operations

In order to derive assets and subsequent operations, we discuss each use case. Aside from the classic single-device user environment and subsequent usage scenarios, we discuss the modern multi-device user environment as well.

The multi-factor authentication use case (UC1) requires the user to decrypt and/or sign data in order to substantiate his own identity. As soon as the required cryptographic primitive is compromised, the authentication is rendered useless. This use case therefore depends on a signing key asset (**A1**) and probably on a decryption key asset (**A2**). The subsequent operations are signature creation (**OP1**) and decryption (**OP2**).

Given the data is protected by hybrid encryption schemes for the protected document exchange use case (UC2), the use case depends on the following assets: the payload data itself (**A3**), a content encryption key (**A2**) for bulk encryption, and key wrapping key (**A4**). The subsequent operations are encrypt/decrypt (**OP3**/OP2) and key wrapping (**OP4**).

To secure the data by the means of cryptography for the cloud storage use case (UC3), the data has to be encrypted before it is sent to the cloud. Therefore, the data asset (A3) is accompanied again by a content encryption key (A2). The subsequent operations are encrypt/decrypt (OP3/OP2).

Both use cases, cloud storage as well as protected document exchange, are more complex in the multi-device user environment. The user wants to access her data from every one of her devices. Therefore, we add key mobility (**OP5**) to the lists of operations. Recent advances in the field suggest that sharing a key might in some cases be better than managing keys for every individual of a system. Hence, we add key sharing (**OP6**) to the list of operations.

The document signing use case (UC5) obviously requires for a signing key (A1) with the subsequent operation of signature creation (OP1). The data integrity protection use case (UC5) adds the signature verification operation (**OP7**) to the list of operations.

Last but not least, the secure messaging use case (UC6) is about session key agreement and therefore requires signing keys (A1) and the subsequent signing operation (OP1) as well as the verify operation (OP7). A user may want to maintain his identity across his different devices and therefore require for the key mobility operation (OP5) as well.

The JavaScript Object Signing and Encryption use case (UC7) requires signature creation/verification (OP1/OP7) as well as data encryption and decryption (OP3 and OP2) with hybrid encryption schemes and thus, key wrapping (OP4).

We consider operations which use these assets as derived assets. Common operations are the basic cryptographic operations like encryption and decryption (OP3 and OP2) and signature creation and verification (OP1 and OP7). Signature operations do need data fingerprinting i. e. hashes (**OP8**). Advanced operations are key wrapping operations (OP4) as well as key mobility and key sharing (OP5 and OP6).

4.5 Threats

The assets and subsequent operations above lead to five distinct threats and result in a total of three different risks.

The first two threats are loss of data/keys (**T1/T2**) either through physical loss or loss of integrity. Loss of data and/or data integrity and loss of key and/or key integrity renders the data corrupt. The user therefore suffers from a denial of service (DoS) situation, the attacker, however, cannot abuse the data. Therefore, we rate the severity of the DoS risk (**R1**) as low.

The second two threads concern data or key theft. While data or key theft might not be noticed by the user at all, the attacker can use the obtained information for her cause. For the data theft threat (**T3**), we rate the severity as medium because even as there is a breach of security and privacy, only a specific set of data is disclosed (**R2**). For key theft (**T4**), there are two options. First, given the compromised key is a CEK (A2), the severity of the data disclosure risk (R2) is high, because the key could be used for multiple sets of data. Second, given the compromised key is a singing key (A1), an attacker can use the key to impersonate the key owner. We rate the impersonation risk (**R3**) as high as well.

The last threat is about key authenticity (**T5**). When an attacker can swap a key for a key of her own, he can perform T1 to T4 without being noticed.

4.6 Evaluating the Software Provider

Browsers, in general, are pieces of software and software has never been particularly good in keeping secrets. We evaluate the threats against a browser implementation of the W3C Web Cryptography API, namely the Firefox web browser, version 39.

The Web Cryptography API suggests that the HTML5 Indexed Database is the place to persist freshly created key data. However, at least with Firefox 39, the Indexed Database is stored in a file on disk and we succeeded in extracting sensitive key data. Literature lists attacks on the Indexed Database (Kimak et al., 2014; Kimak et al., 2012) as well.

Therefore, we rate any cryptography in need for sensitive key data as being easily attacked and the corresponding risks to be high. Only the hashing operation (O8) can be assumed to be secure. The key loss and therefore data loss risks (R1) are likely to occur because clearing the browser cache removes the cryptographic keys for good. This is particularly cumbersome with the use cases cloud storage (UC3) or signature creation (UC4). The risk of impersonation (R3) is high due to the fact that keys are not stored securely and might be tampered with. For a practicability rating, i. e. performance and operational overhead, all operations except for key mobility and key sharing (OP5 and OP6) can be implemented very efficiently.

4.7 Evaluating the Hardware Provider

A hardware-assisted cryptographic service provider boosts the security of sensitive data considerably. Our Cordova provider uses the key chain implementations of current smart phones. These implementations of current phones tend to rely on secure elements. They are designed to keep the sensitive data save in case of malware and even if the device is stolen.

These features render key and data disclosure risks (R2) as well as key impersonation risks (R3) almost impossible. The risk of losing keys and data (R1) is low as well. Since the key data is held locally, the practicability ratings are similar to the software implementation. All operations, except for the key mobility and key sharing operations (OP5 and OP6), can be implemented efficiently.

4.8 Evaluating the Remote Provider

CrySIL represents off-device cryptographic services providers that operate as web-accessible services with key usage constraints based on authentication methodologies. We assume that the crypto service of CrySIL operates on a HSM and therefore assume any operation to be hard to attack in general. However, we consider the risk of losing the key (R1) as the weakest point of the system because of DoS attacks. An unreachable service results in an unusable key and thus leads to undecryptable data.

As for practicability, we consider a remote webservice to be less practical as a local solution. That is simply because sending commands and data via the web takes its time. We consider cases of bulk data processing, as it is done by content encryption (OP3) and hashing (OP8) as hardly feasible. However, remote service do have their strength when it comes to key mobility (O5) and key sharing (O6).

4.9 Summary

A tabular overview of all operations and the likelihood of the respective risks along with a practicability rating is given in

Table 1: Risk assessment.

	Browser				Cordova				CrySIL			
	R1	R2	R3	p	R1	R2	R3	p	R1	R2	R3	p
OP1		-	-	+		+	+	+		+	+	o
OP2	-			+	+			+	o			o
OP3		-		+		+		+		+		-
OP4	-	-		+	+	+		+	o	+		o
OP5		-	-	-		+	+	-		+	+	+
OP6		-	-	-		+	+	-		+	+	+
OP7				+				+				o
OP8		+		+		+		+		+		-

Table 1. The likelihood rating is given as $Rx \in \{-, o, +\}$ corresponding to low, medium, and high likelihood of having to deal with the risk. The practicability rating p follows the values $p \in \{-, o, +\}$ as bad, medium, and good practicability, respectively.

Our evaluation highlights that no provider is perfect. Each of the evaluated providers can perform different operations with different practicability and different security levels.In detail, the software security provider is particularly good in processing bulk data as required by the hashing (O8) and bulk encryption (O3) operations. Protecting sensitive key data is done best by a hardware-assisted cryptographic service provider. And finally, whenever key mobility (O5) and key sharing (O6) is required, a remote cryptographic service provider is the natural way.

5 MIND THE GAP!

The evaluations made in the previous sections make clear that no one distinct cryptographic service provider implementation can excel in every cryptographic task and use case. Furthermore, unforeseen provider implementations/types can change features and security of an application considerably. In other words, there is a gap between what current methodologies of applied cryptography can cope with and what is needed to properly serve the modern user's data protection needs.

5.1 Missing Features

First and foremost, an application needs to use the most suitable cryptographic service provider type and implementation for a task at hand. In short, key protection capabilities of different platforms and devices vary and subsequently make some devices more suitable for handling sensitive keys than others. We believe that a modular interoperability layer which is flexible in terms of software technology, operating system, connectivity and of course, cryptographic features can close this gap.

Second, modern multi-device users need to access their cryptographic keys wherever they need them—be it on a smart phone or on a desktop PC. Only so can users still benefit from the convenience of current applications and service while maintaining proper data security. However, current technology hardly allows for this kind of flexibility.

Third, we believe that the user deserves to decide which cryptographic service provider she wants to use. With such a feature in place, a user can use her remote keys in an arbitrary application and thus, push her data security level. She can also decide to spend more time doing cryptography rather than offload her data to an untrusted service to just save a few seconds. Users themselves might not (jet) use the feature, however, once the feature is there, a policy system can be established. Developers can then install policies to cope with selecting the most suitable cryptographic service provider for a task at hand while leaving the user the chance to change the selection. Such functionality would benefit the user by being able to rely on developer defaults as we do today but change the selection if necessary.

Last but not least, it is crucial for applied cryptography to be easy to use for users and developers. For the user, a solution needs to be completely transparent during everyday use. Interaction should only be necessary either for authentication purposes or for changing settings. For developers, we need means of transparently handling multiple different cryptographic providers without the the need for changing existing cryptographic APIs. Creating a new API might just lead to yet-another crypto-API which hinders broad acceptance.

We believe that these four features would greatly support applied cryptography in its attempt to keep up with current heterogeneous device landscapes, distributed applications, and multi-device users.

5.2 Candidates

First and foremost, the Java Cryptographic Extension (JCE) offers provider-based cryptographic services since its introduction with Java 1.1. The extension enables the use of separate providers for different tasks and has been successfully used for hardware security modules and a variety of software providers. However, when it comes to remote cryptographic providers, JCE falls short in terms of API features.

The World Wide Web Consortium (W3C) set out to unify cryptography for browsers by proposing their Web Cryptography API (Sleevi and Watson, 2014), an attempt to provide better access to cryptography to web applications. The API is designed to use the cryptographic features of the underlying operating system. All in all, W3C's Web Cryptography API is without doubt a promising concept but quickly reaches it limits when used in current device and application landscapes.

The Cryptographic Service Interoperability Layer (CrySIL) concept (Reimair et al., 2016) has been created to satisfy current requirements. It offers transparent use of cryptographic service providers by hiding the interoperability layer behind common cryptographic APIs like JCE, PKCS#11, or Microsoft's CNG. The concept can offer local providers and off-device cloud key services as well. The framework even handles basic authentication tasks out of the box. All in all, as CrySIL can provide local software cryptography providers, hardware backed providers, as well as remote providers without a change in interface, CrySIL is, as of today, the most promising approach to tackle current challenges.

6 CONCLUSIONS

New technologies in computing are without doubt enablers on new applications and use cases. However, applied cryptography as of today cannot always keep up with the fast-paced evolution of device environments and applications.

In this work, we reduce cryptographic service providers to three different types namely software, hardware, and remote providers. Our case study showed that current applications cannot assume to use one specific provider even on the same device class. Furthermore, we found that the characteristics of an application can change considerable by changing the provider type. Not knowing which provider is going to be used can therefore compromise security of an application. And finally, our security analysis showed that no single provider can excel in every use case.

Based on these findings we create a list of features that we believe would answer the challenges given:

- an application has to use the cryptographic service provider which is most suitable for a task at hand,
- modern multi-device users need to access their cryptographic keys wherever and whenever in need,
- the user deserves to decide which cryptographic service provider she wants to use, and
- it is crucial for applied cryptography to be easy to use for users and developers.

These features are not yet available in applied cryptography and we believe that providing these features will get personal data security up to speed again.

REFERENCES

Backes, M., Bugiel, S., Derr, E., Gerling, S., and Hammer, C. (2016). R-droid: Leveraging android app analysis with static slice optimization. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '16, pages 129–140, New York, NY, USA. ACM.

Baumgärtner, L., Graubner, P., Schmidt, N., and Freisleben, B. (2015). Andro lyze: A distributed framework for efficient android app analysis. In *2015 IEEE International Conference on Mobile Services*, pages 73–80.

Chang, V., Kuo, Y.-H., and Ramachandran, M. (2016). Cloud computing adoption framework: A security framework for business clouds. *Future Generation Computer Systems*, 57:24 – 41.

Chang, V. and Ramachandran, M. (2016). Towards achieving data security with the cloud computing adoption framework. *IEEE Transactions on Services Computing*, 9(1):138–151.

Egele, M., Brumley, D., Fratantonio, Y., and Kruegel, C. (2013). An empirical study of cryptographic misuse in android applications. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 73–84, New York, NY, USA. ACM.

Fernandes, D. A. B., Soares, L. F. B., Gomes, J. V., Freire, M. M., and Inácio, P. R. M. (2014). Security issues in cloud environments: a survey. *International Journal of Information Security*, 13(2):113–170.

Gutmann, P. and Grigg, I. (2005). Security usability. *IEEE Security & Privacy*, 3:56–58.

Halpin, H. (2014a). The W3C Web Cryptography API: Design and Issues. In *Proceedings of the 5th International Workshop on Web APIs and RESTful design (WS-REST)*, Seoul, Korea.

Halpin, H. (2014b). The W3C Web Cryptography API: Motivation and Overview. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 959–964. W3C.

Kimak, S., Ellman, J., and Laing, C. (2012). An investigation into possible attacks on HTML5 indexedDB and their prevention. In *The 13th Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking and Broadcasting (PGNet 2012)*, Liverpool, UK. Liverpool John Moores University.

Kimak, S., Ellman, J., and Laing, C. (2014). Some Potential Issues with the Security of HTML5 IndexedDB. *IET Conference Proceedings*, pages 2.2.2–2.2.2(1).

Lazar, D., Chen, H., Wang, X., and Zeldovich, N. (2014). Why does cryptographic software fail?: A case study and open problems. In *Proceedings of 5th Asia-Pacific Workshop on Systems*, APSys '14, pages 7:1–7:7, New York, NY, USA. ACM.

Mohamed, I. and Patel, D. (2015). Android vs ios security: A comparative study. In *2015 12th International Conference on Information Technology - New Generations*, pages 725–730.

National Institute of Standards and Technology (2001). *FIPS140-2: Security Requirements for Cryptographic Modules*.

Reimair, F. (2011). Trusted virtual security module.

Reimair, F. (2014). Cloud-based signature solutions: A survey. Technical report, Secure Information Technology Center - Austria.

Reimair, F., Teufl, P., and Zefferer, T. (2016). CrySIL: Bringing Crypto to the Modern User. In *Web Information Systems and Technologies*, volume 246 of *Lecture Notes in Business Information Processing*. Springer.

Reiter, A. and Zefferer, T. (2015). Power: A cloud-based mobile augmentation approach for web- and cross-platform applications. In *Cloud Networking*.

Ren, K., Wang, C., and Wang, Q. (2012). Security challenges for the public cloud. *IEEE Internet Computing*, 16(1):69–73.

Santos, N., Raj, H., Saroiu, S., and Wolman, A. (2014). Using arm trustzone to build a trusted language runtime for mobile applications. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 67–80, New York, NY, USA. ACM.

Sleevi, R. and Watson, M. (2014). W3C Candidate Recommendation: Web Cryptography API.

Varadharajan, V. and Tupakula, U. (2014). Security as a service model for cloud environment. *IEEE Transactions on Network and Service Management*, 11(1):60–75.