MoCrySIL – Carry Your Cryptographic Keys in Your Pocket

Florian Reimair, Peter Teufl, Christian Kollmann and Christoph Thaller
Graz University of Technology, Graz, Austria

Keywords: Cloud Security, Central Cryptographic Solutions, Advanced Cryptographic Protocols, Heterogeneous Applications, Mobile Devices.

Abstract: Today's applications need to share data and workload in heterogeneous device environments. Many of these handle sensitive data and need to make use of cryptography, which induces keys that have to be provisioned, stored and shared securely. Our Cryptographic Service Interoperability Layer (CrySIL) architecture addressed these challenges by storing the key material off-device in a central hardened service that provides cryptographic functions to arbitrary devices via standardised APIs. While CrySIL is typically deployed by a trusted entity utilising hardware-security-modules (HSMs), the setup of this central trusted instance might be too complex or not desired in SME/personal deployment scenarios. Therefore, we present MoCrySIL, an extension to CrySIL that omits the need for a thrusted third party by making use of hardware-backed key-storage facilities available in today's smart phones. We describe the MoCrySIL architectures and present a prototype that performs S/MIME based email encryption/signatures via a PKCS#11 library. We conduct a thorough security/risk analysis, and reflect on functional achievements and shortcomings.

1 INTRODUCTION

In recent years, a highly heterogeneous platform landscape has become the standard scenario for the development and deployment of applications. Especially, mobile device platforms introduced an unprecedented level of architectural heterogeneity that needs to be considered for all aspects of the application development process. In addition to the relatively new mobile operating systems and platforms, rich HTML5-based cross-platform and browser applications gain importance while classic desktop operating systems still stand their ground. Developing applications for these diverse environments boosts complexity of the processes significantly. Especially, security-related aspects – the deployment of cryptographic algorithms and protocols to offer confidentiality, integrity and authenticity for the processed data – suffer in several ways: **First**, managing cryptographic keys on multiple platforms is a significant challenge either due to the lack of required key storage facilities, or due to their strongly deviating levels of security. **Second,** low- and high-level cryptographic algorithm and protocol implementations are not available on every platform. **Third**, the high level of complexity is the likely cause for severe implementation mistakes that lead to significant security issues (e.g., (Egele et al., 2013; Fahl et al., 2012)). With this work, we follow the call for action issued by (Rocha and Correia, 2011) and the need for addressing important *personal* crypto use cases such as health data protection (Benaloh et al., 2009) and/or secure cloud storage for the *industry* as proposed in (Kamara and Lauter, 2010), among others.

Scientific research and the industry reacted to the situation and created a plethora of crypto platforms, communication protocols, and key management solutions. Most of these solutions, however, offer a very limited set of cryptographic functions, deploy (implicit) authentication mechanisms tailored to specific requirements, or are not flexible enough to be deployed in constantly changing environments.

We recently presented the Cryptographic Service Interoperability Layer (CrySIL) (Reimair et al., 2015) concept as our approach to complement existing approaches and stand the challenges of key management and utilising cryptographic functions within heterogeneous application deployment scenarios. CrySIL is all about getting access to the keys, everywhere and at any time. And it does so by offering a centralized key storage and crypto engine. In a typical scenario CrySIL is deployed in private cloud environments and utilises HSMs for protecting key material. While this setup is highly relevant for enterprise-based applica-

Reimair F., Teufl P., Kollmann C. and Thaller C..
MoCrySIL – Carry Your Cryptographic Keys in Your Pocket.
DOI: 10.5220/0005547902850292
In *Proceedings of the 12th International Conference on Security and Cryptography* (SECRYPT-2015), pages 285-292
ISBN: 978-989-758-117-5

tions, it might not be suitable for personal use cases or SMEs due to setup complexity and hardware costs (HSMs).

Thus, we extend the flexible CrySIL architecture to gain a mobile version (MoCrySIL) based on today's mobile hardware-backed key storage facilities. While foreclosing the need for yet another third party, it provides a personal crypto platform for one's pocket. The platform features direct key management and a crypto engine that can be used for VPN, email signature/encryption, and data encryption, among others. Finally, transferring the CrySIL architecture from the typical server/HSM based environment to the mobile scenario allowed us to evaluate and improve the flexibility and security of the original CrySIL architecture.

To evaluate the security of the MoCrySIL system, we present a prototypical Android-based MoCrySIL system that is used for S/MIME-based email signatures/encryption (via a CrySIL-enabled PKCS#11 API). The cryptographic keys are stored in the hardware-backed Android KeyStore. The communication between the PKCS#11 API on the client and the MoCrySIL app on the Android device is established via an external WebVPN component, which utilises push notification facilities and long lived WebSocket connections. The communication path is protected by an end-2-end TLS tunnel. We present a functional analysis as well as performance statements. Although standard protocols and components form the basis of the MoCrySIL architecture, their new combination, the mobile nature of the system and the requirements for protecting key material and cryptographic functions mandate a thorough security/risk analysis.

The results of our evaluation indicate a success in providing low-footprint access to cryptographic primitives from different devices, and, given internet access, anywhere and at any time. With MoCrySIL, we created a central key storage solution with most advantages of centralised key storage services while not introducing yet another trusted third party. The key material never leaves the central crypto engine and therefore does not have to be guarded at every client device. Access to the primitives is guarded by build-in strong authentication procedures. While MoCrySIL introduces latencies that renders it not suitable for bulk encryption, it succeeds in getting cryptography to places where crypto could not securely be used before (e. g. in browsers).

The subsequent work is structured as follows. Section 2 presents solutions which already try to stand the challenges but fall short in multiple aspects. The general CrySIL architecture is described in Section 3. The requirements, adaptations and the detailed workflow for MoCrySIL are introduced in Section 4. The system is then subject to a detailed risk/security analysis in Section 5. We conclude the work with a glimpse to the future.

2 RELATED WORK

A number of server-based cryptographic services have been implemented by the industry. This section gives an overview and functional evaluation of selected cryptographic services that can be integrated into cloud-based environments.

Dictao[1] and *Cryptomathic*[2] support digital signatures for transaction security and user authentication. Both services facilitate key access by authenticating clients using simple credentials, such as username/password schemes, or credentials that feature higher strength (eID cards, OTPs, mobile devices etc). While *SigningHub* and *Cryptomathic* are deployed as cloud services, *Dictao* requires integration into an enterprise IT infrastructure. Basically, the provided functionality is limited to user authentication and signature creation.

The so-called *Austrian Mobile Phone Signature*[3], one implementation of the Austrian citizen card concept (Leitold et al., 2002), is a cloud-based service and uses a hardware security module (HSM) to store the private signature keys of all Austrian citizens. Access to these keys is protected by a strong two-factor authentication mechanism, involving a password as well as an OTP being sent to the citizen's mobile phone. Applications can access the Austrian Mobile Phone Signature and its signature creation functionality through a well-defined XML interface. Although the Austrian Mobile Phone Signature meets the demands of the law, the currently deployed implementation fails to support use cases other than signature creation and user identification.

Scientists adopted the new environment and requirements as well. A very intuitive approach is to host the cryptographic keys centrally and hand copies to clients on request. The OASIS Key Management Interoperability Protocol (KMIP) (KMIP-v1.1, 2013) and the Cloud KMIP (CKMIP) (Lei et al., 2010) enable this kind of key distribution. While this kind of key distribution may be effective and spot-on for private cloud environments, the approach does not suffice for personal use cases and public cloud appliances.

[1] https://www.dictao.com

[2] http://www.cryptomathic.com

[3] https://www.handy-signatur.at

Trusted Computing, as proposed by the Trusted Computing Group (TCG), introduces a hardware security module (HSM), the trusted platform module (TPM) (Trusted Computing Group, 2011), into commodity off-the-shelf hardware for personal computers. This HSM allows for rather secure personal key storage. Together with virtualisation solutions, scientific research was successful in securely storing and using keys without distributing them to the client (Bleikertz et al., 2013; Butt et al., 2012; Toegl et al., 2013). However, these approaches introduce a trusted third party for key management which is not ideal for public cloud environments.

All in all, there are plenty of attempts to stand the challenges. Most of them fall short for the use case of easy-to-use, flexible, and portable *personal* key management for a multi-device user.

3 CrySIL ARCHITECTURE

The Crypto Service Interoperability Layer (CrySIL) concept (Reimair et al., 2015) has been created to meet today's heterogeneous device landscape and allows the user to use her keys within cryptographic operations regardless where the key resides and where it is needed. In a nutshell, the user takes one of her devices and launches an application to perform some cryptographic task. The application interfaces with the interoperability layer, CrySIL, which connects to another device. This other device has access to the actual cryptographic primitive, creates and validates authentication challenges if required and performs the requested operation. The result is returned to the interoperability layer and back to the application running on the device of the user. A graphical illustration of the workflow is given in Figure 1.

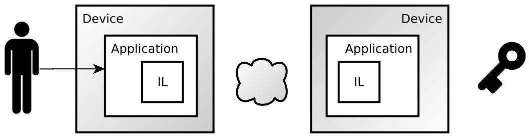

Figure 1: CrySIL's basic architecture.

CrySIL introduces great flexbility by breaking the classic cryptographic provider apart. The resulting parts – modules – have different jobs and work together to form the actual cryptographic service provider. *Receivers, actors*, a *router*, and other modules handle inter-node communication, protocol mappings, crypto, advanced crypto and authentication. A *receiver* for example acts as a protocol bridge to the interoperability layer with pre-build implementations for the Windows CNG, the W3C Web Cryptography API as well as a SOAP-based web interface. An *actor* makes the services of a crypto provider, e.g. a smart card or an HSM, available to the layer and collects authentication information as they are required. *Authentication* modules do handle the authentication challenges issued by *actors*. The *router* connects the modules together. Anyhow, all modules are considered as building blocks and are not restricted to any technology, platform, or programming language. Every configuration of a router and other modules is referred to as a *CrySIL node* and forms the heart of the concept. An illustration of modules and their interconnections is given in Figure 2.

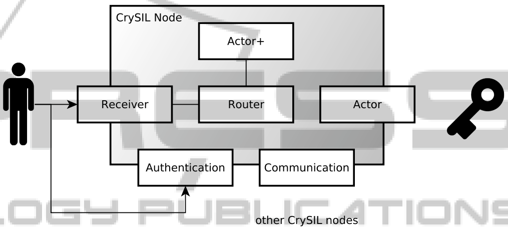

Figure 2: CrySIL node architecture overview.

Communication modules forward requests to other nodes and therefore renders off-device crypto completely transparent to the user, the developer, and to the application while maintaining a simple architecture. A user benefits from her ability to use a variety of keys within different cryptographic operations provided by different crypto providers from a variety of applications on different devices.

4 MoCrySIL – A CrySIL EXTENSION

Although today's mobile devices may not compete with the security levels of dedicated cryptographic smart cards or similar hardware, they do offer hardened operating systems and hardware-backed key storage solutions. Furthermore, since the mobile device is at the owner's fingertip, direct user interaction is available. For example, when the mobile device receives some request to perform a cryptographic operation with sensitive key data, the mobile device can ask the user (the key owner) for allowance.

Together with the rather novel deployment options provided by CrySIL, this allows for moving the central cloud-based key service to a user's mobile device. MoCrySIL extends the standard CrySIL architecture by nodes that can be deployed on mobile devices and an external WebVPN component that ensures that these nodes are reachable from the internet.

The WebVPN approach solves the reachability challenge for mobile devices (NAT, changing IP addresses etc.) by acting as relay service that uses standard push notification systems and long lived WebSocket connections. The communication paths via WebVPN are protected with end-2-end TLS tunnels, which eliminates the possibility for a possibly malicious WebVPN node to read or manipulate CrySIL messages. An additional *communications* module communicates with the WebVPN component, a capable *actor* connects the mobile phone key store to CrySIL.

For explaining the detailed application workflow, the following assumptions were taken: An Android device is used for the MoCrySIL application. The WebVPN component is hosted by a public cloud provider. The complete workflow can be categorised in two main steps – **(A)** the WebVPN enrolment process and **(B)** using the system. In this case "using the system" refers to handle signed and encrypted emails via a client that is S/MIME enabled and capable of using the CrySIL infrastructure via standardised APIs (e.g., PCKS#11). An illustration of the overall workflow is given in Figure 3.

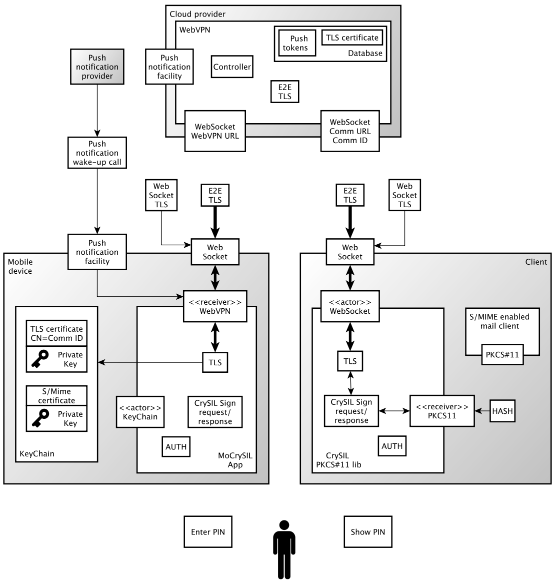

Figure 3: MoCrySIL architecture overview.

(A1) WebVPN Setup. The user starts the *MoCrySIL app* for the first time and initiates the enrolment process over a TLS tunnel. For setting up the *WebVPN* account the user needs to authenticate herself via a pre-defined method (depending on security requirements). The application then requests a unique *Comm ID*[4] used to setup the URL which acts as *CrySIL* interface for the *clients*. The *MoCrySIL app* then generates a *TLS certificate* that stores the *Comm ID* within the common name (CN). The corresponding cryptographic key is directly generated within the Android KeyChain and thus will never be stored/used in insecure environments. The generated certificate is sent to *WebVPN*, which verifies the correctness of the *Comm ID* (via the CN) and signs the certificate with its own private key. The signed certificate is then returned to the *MoCrySIL app* and stored in the *WebVPN* database. *MoCrySIL* now drops the initial TLS-based communication channel and establishes a client-TLS WebSocket based communication channel by using the just generated certificate. The *MoCrySIL app* initialises the push notification system (in this case *Google Cloud Messaging*). The required tokens are associated with the *Comm ID* and are stored in the *WebVPN* database.

(A2) Storing/Generating S/MIME Key Material. In this step certificates for processing S/MIME signed/encrypted emails are setup in the Android KeyChain of the *MoCrySIL* device. Certificates can be self signed or issued by a certification authoriy (CA). However, as there are no special requirements to the certificate issuing process it is considered out of scope for this work.

(A3) Setting up Client Communication. To allow for secure communication between the *client* and the *MoCrySIL app* the required trust for the *TLS certificate* needs to be established on the client. The *MoCrySIL* app displays the WebVPN endpoint URL (including the *Comm ID*) – further denoted as *Comm URL*, which is then configured on the client. Then the first connection is established between the *client* and the *MoCrySIL app* (more details in **B1**). To establish a trust relationship, the fingerprint of the *TLS certificate* is displayed on the client device[5] and the *MoCrySIL app*. When both fingerprints match, a possible MITM attack can be ruled out. The *TLS certificate* is stored on the client and used for protecting *CrySIL* protocol data.

(B1) Signing an Email. The plain email is handed over to the S/MIME signature component of the email client, which calculates the hash value and hands it over to the (examplary) PKCS#11 CrySIL API. The API then prepares a CrySIL signing request.

Before issuing the request to the *MoCrySIL* app, a secure TLS end-2-end channel (denoted as E2E TLS in Figure 3) is to be established. The TLS handshake

[4] Requirements for the communication ID in terms of length and randomness are similar to those of session identifiers.

[5] (e.g., by the CrySIL enabled PKCS#11 API)

data is encoded in CrySIL protocol packets and sent to the *Comm URL*. *WebVPN* uses the *Comm ID* to retrieve information about the respective *MoCrySIL app* and push-notification tokens from the database. In case no long lived WebSocket connection exists already, a “wake-up-call” is sent to the *MoCrySIL app* via the push notification system. The *MoCrySIL app* establishes a long lived TLS protected WebSocket connection to *WebVPN* by authenticating via the *TLS certificate*. The TLS handshake between the *client* and the *MoCrySIL app* can now be completed. The *client* verifies the validity of the *TLS certificate* by using the hash values that were accepted by the user during the registration process.

Now, the actual CrySIL request is sent to the MoCrySIL app. The *MoCrySIL* actor receives the incoming signing request and initiates the (current[6]) authentication process: **(a)** The *MoCrySIL app* generates a PIN code that is sent via the *CrySIL* authentication challenge to the *client*. **(b)** The *client* displays the PIN code to the user, who then **(c)** needs to provide the code to the *MoCrySIL app*. After a successful authentication process, the hash is signed and returned to the *client*. The client extracts the signature response and hands it over to the *email client*, which then finalises the S/MIME email and hands it over to the respective mail transfer agent. The communication channels between the different entities remain open for a pre-defined session time, which allows the efficient transmission of further requests/responses without the need to build up the TLS channel and send the required push notifications.

5 EVALUATION

With the MoCrySIL approach, we successfully transformed a mobile device into a portable cryptographic platform with a direct key management system. There is no need for a trusted third party, yet MoCrySIL implements a centralised solution. On top of that, the third parties that are required for waking the MoCrySIL application cannot read/manipulate data due to the end-to-end encryption done by CrySIL. Establishing trust relationships between consuming and serving CrySIL nodes does not require trusted third parties either. The underlying CrySIL technology depends on state-of-the-art technology and is designed to meet upcoming requirements. For evaluating the MoCrySIL approach we did a prototypical implementation. To understand consequences in relation to security in these system we present a detailed security/risk analysis of the system.

5.1 Security/Risk Analysis

For the conducted risk analysis we assume an application scenario that deploys the S/MIME standard for signing and encrypting emails. Thereby the MoCrySIL application is installed on a current Android 5.0 devicethat offers a hardware-based Key-Chain for storing the asymmetric key data. As client a desktop based mail client is used that supports S/MIME and is capable of using cryptographic functions via the standardised *PKCS#11* API.

5.1.1 Entities and Assumptions

The ***client***[7] is the device running the application, which utilises the cryptographic functions offered by the CrySIL infrastructure. For the most part it is considered to be out of scope for the conducted security/risk analysis. The ***mobile device*** hosts the *CrySIL* instance and stores and handles the asymmetric cryptographic key data used for S/MIME based encryption/signature processes. The ***user*** operates the mail application on the respective ***client***, handles the *MoCrySIL app* on the Android device, registers the *MoCrySIL* app with *WebVPN* and the ***client***, and uses the *MoCrySIL* infrastructure by signing/decrypting mails on the ***client***. *WebVPN* is hosted on a public ***cloud provider***[8] that is assumed **(1)** to be honest-but-curious. **(2)** The possibility of a successful attack by a third party needs to be considered. This party has the interest to manipulate *WebVPN* or listens to communication. The ***CrySIL provider*** develops and maintains the *CrySIL* components. These include the *MoCrySIL app*, the ***client*** PKCS#11 CrySIL library and *WebVPN*. The ***CrySIL provider*** is considered as trustworthy. However, similar to the ***cloud provider*** the components might be successfully attacked by a third party. Finally, the ***push notification provider*** is hosted by the respective mobile platform – in this case Google. Here, the same assumptions as for the ***cloud provider*** are taken.

5.1.2 Assets

The plain ***email data*** handled on the ***client*** is considered as the primary asset. The ***client’s*** crypto APIs are capable of handling bulk cryptographic operations (e.g. hashing and symmetric algorithms) as well as

[6] The current authentication method just represents a simple example and could be replaced by a wide variety of authentication methods tailored to the specific environments.

[7] Entities and assets are formatted in ***italic/bold.***

[8] such as Amazon EC2, or Jelastic

key wrapping and signature verification. Thus, the *MoCrySIL app* and the key material is only used for **(1)** signing the hash values and **(2)** for decrypting the symmetric content encryption keys. The ***cryptographic material*** (symmetric keys and hash values) needs to be transferred to the *MoCrySIL app*. Knowing the symmetric keys would allow an attacker to decrypt an encrypted email, whereas being able to manipulate the transferred hash values would allow an attacker to sign arbitrary documents in the name of the legitimate user. Another core security asset is represented by the ***asymmetric cryptographic keys*** stored in the Android KeyChain. An attacker who can extract or use these keys is capable of signing arbitrary content in the name of the legitimate user or decrypting her emails.

5.1.3 Threats, Countermeasures and Risks

By considering the assumptions, the assets, and the different players the proposed system can be analysed in respect to attacks (**A**), threats (**T**), countermeasures (**C**) and residual risks (**R**).

Cloud Provider. **(A1)** The honest-but-curios ***cloud provider*** might listen to the ongoing communication but does not attack the deployed applications. **(A2)** The ***cloud service provider*** is successfully attacked by a third party, who deliberately intends to gain access to or manipulate data transferred over *WebVPN*. In a similar way, a possibly malicious ***cloud provider*** needs to be considered.

(T1) Denial-Of-Service (relevant for **A1, A2).** The communication between the clients and the MoCrySIL device might be subject to a DOS attack, which prohibits the utilisation of the provided cryptographic services. The current system is not capable of addressing this threat due to the reliance on the WebVPN component. As a consequence the residual risk of a DOS-attack remains (**R1 – DOS CrySIL communication**).

(T2) Sniffing Communication Meta Data (relevant for **A1, A2).** The attacker is capable of sniffing communication meta data of the connections between clients and MoCrySIL devices. This meta-data includes IP addresses, information that can be extracted from the TLS handshake (certificates, cipher-suites) for the end-2-end security tunnel or usage data (time, amount of traffic). Capturing this data cannot be mitigated by the current system (**no countermeasures**) and thus needs to be considered as residual risk (**R2 – Gaining communication meta data**)[9].

(T3) Sniffing/Manipulating Cryptographic Material (relevant for **A2,** partly **A1**)**.** A successful attacker might try to sniff (**A1, A2**) or manipulate transferred cryptographic material (cryptographic hashes or symmetric keys) (**A2**). Manipulating this data would allow an attacker to sign arbitrary documents (by exchanging the transferred hash values) or decrypt arbitrary symmetric keys (see **T5** for details).

(T4): Re-routing Packets (relevant for **A2).** The attacker re-routes **(1)** CrySIL protocol packets from a legitimate client to other MoCrySIL devices, or **(2)** from a malicious client to a specific MoCrySIL device: The latter has a high impact on security, because it could enable an attacker to sign arbitrary documents in the name of the attacked MoCrySIL user. Also, documents belonging to that user could be decrypted by sending malicious decryption requests to the CrySIL infrastructure (see **T5** for details).

(T5): Man-In-The-Middle Attack. T3 and **T4** are mitigated by the end-2-end encryption TLS tunnel (**C1 – TLS tunnel**). However, the system is somewhat susceptible to MITM attacks. During the initial communication handshake the user currently needs to verify the validity of the certificate by comparing certificate hash values that are displayed on the MoCrySIL app and the client. Failing to do so would allow an attacker (**A2**) to carry out an MITM attack and read and manipulate the CrySIL protocol packets. Therefore, the residual risk of a MITM attack – with the consequences described in **T3** and **T4** – needs to be considered (**R3 – MITM attack**).

Push Notification Provider. Attack vectors on the push notification provider are highly similar to those on the cloud service provider: **(A3)** Honest-but-curious **Push Notification Provider** (similar to **A1**) and **(A4)** External attack on **Push Notification Service** (similar to **A2**). However due to "wake-up" only nature of the push notifications the consequences differ from those of **A1** and **A2**.

(T6) Denial of Service (relevant for **A4).** The push notification service of WebVPN could be subject to a DOS attack. In this case, the "wake-up-call" to initiate the WebSocket channel would be lost. This can simply be mitigated by allowing the user to manually start the MoCrySIL app (**C2 – Manual establishment of communication channel**). Therefore, the residual risk of this DOS attack has a very low impact (**R4 – DOS Push Notification System**).

(T7) Sniffing/Manipulation of the Push Notification Messages (relevant for **A4**, partly **A3**)**.** In this case the attacker listens to push notification communication (**A3**, **A4**), or manipulates (**A4**) push notifi-

[9] Deploying the WebVPN components within a private cloud can address this residual risk. However, this scenario is not considered in the assumptions.

cation tokens or sends arbitrary tokens (**A4**) to the MoCrySIL app. Due to the "wake-up-call" nature of the push notifications, there is no residual risk in learning their content.

Mobile Device and User. The attacks are mainly related to attacking the mobile device either by gaining physical access or by injecting malicious software. **(A5)** refers to stealing the device with the intention to extract/or use the key material stored on the mobile device. **(A6)** refers to an attack carried out by a malicious application that does not/cannot gain root access. **(A7)** refers to malware that is capable of gaining root access. (**A8**) refers to external attacks originating from an outside location by exploiting weaknesses of the system or the user.

(T8) Extracting Cryptographic Keys (relevant for **A5, A7**). By extracting the asymmetric keys, the attacker can arbitrarily use them to create signatures or to decrypt symmetric key material. The impact is different when **(1)** the user is aware of attack (e.g., due to stealing the device) and **(2)** the user is not aware of the key extraction process (e.g. due to malware). The latter allows an attacker to use the keys for a long time before countermeasures are taken. The most effective countermeasure against the extraction of private key material is the utilisation of the hardware-backed key storage (**C3 – Hardware-backed key storage**). There remains a very low risk that the key material could be extracted via a side-channel attack. However, an application scenario in which an attacker is willing carry out this complex attack must never be considered for a mobile use case.

(T9) Using Cryptographic Keys (relevant for **A5, A7**). This threat needs to be considered in detail, since it can only be partly mitigated by the employed hardware element. An attacker who gains access to the CrySIL interface offered by the MoCrySIL app and either knows the authentication credentials or circumvents the authentication system can arbitrarily sign and decrypt data. Similar to **T8** the impact is considered to be higher when the user is not aware of the ongoing misuse. All in all, the residual risk and countermeasures are strongly tied to mobile device security (**R5/C5 – Security of mobile device**).

(T10) Stealing Credentials (relevant for **A7** and especially for **A6**). Due to the nature of the MoCrySIL authentication process, this threat does not apply to this type of credentials. However, credentials, that are used to protect the Android KeyChain and the mobile device could be subject of an attack. Such an attack could rely on malware that either directly captures the entered credentials (**A7**), lures the user into entering them (phishing) (**A6**). Phished credentials could then be used when the device is stolen at a later time. Especially, the malware problem can be mitigated by the same countermeasures as described in **T9**. The risk of gaining access to the credentials by shoulder surfing needs to be considered as residual risk (**R6 – Stealing credentials**).

(T11) Accepting a Malicious Request (relevant for **A8**). An attacker who gets to know the *Comm ID* could send a CrySIL request to the CrySIL URL. However, the authentication process requires the user to enter a PIN code that is only shown by the client's CrySIL library. Thus, the user is not able to enter the correct PIN code, even if she would interpret the malicious request as valid (**C6 – CrySIL authentication system**). However, an attacker could easily mount a denial of service attack by sending arbitrary requests to the communication end point. This threat cannot be addressed by the current CrySIL authentication features and thus manifests in a residual risk (**R7 – DOS by an external attacker**).

(T12) Impersonating Another Device (relevant for **A8**). An attacker might try to impersonate a target device when connecting to the WebVPN communication end point. Even if the attacker learns the *comm ID*, he still has no access to the *TLS certificate* used by the device for WebVPN authentication (**C7 – WebVPN authentication**).

5.1.4 Discussion on Risks

The residuals risks are mainly associated with DOS attacks (**R1**, **R4**, **R7**), gaining meta-information (**R2**), deploying malware on the device (**R5**, **R6**) and attacking the user (**R6**, **R3**). The attacks related to DOS and extracting meta-information do not have a significant security-related impact on the MoCrySIL architecture. However, the remaining risks have severe consequences and need to be mitigated by managing the device and implementing tight security policies (device encryption, password locks etc.).

With having these policies in place, MoCrySIL's security services offer a solid level of security.

5.2 Performance

As for performance, relying on a third party for communication takes time. Yet, our use of long-lived Websocket connections reduces the amount of time lost in creating a connection significantly.

As for the integration efforts, whenever an application utilises well-known crypto APIs, CrySIL can be integrated with a minimal effort. As of today, we work on receivers for PKCS11, JCE, MS CNG and W3C Crypto API. One can easily implement such a module if needed. For Java, our JCE receiver module for example is implemented using only 1000 lines of prototypical code, router and sending communications modules do have 80 loc each with a common protocol definition of 1500 loc. The code of imported libraries is not included in the numbers given.

6 FUTURE WORK AND CONCLUSIONS

With MoCrySIL, we present our solution of a secure, flexible and portable personal key store service one can carry around in his pocket. MoCrySIL removes the need for a trusted third party and therefore complements our CrySIL solution well.

Our prototype implementation highlights the flexibility of the CrySIL architecture, shows its potential and affirms the ease of use for developers and end users. The security analysis of MoCrySIL indicates that using hardware-backed key storage facilities on mobile devices allows to reach a significant level of security. Due to the uncontrolled environment a mobile device is used in, however, the residual risks for certain threats is higher than in classic deployment scenarios. Therefore, MoCrySIL focuses on standard applications where the hardware-backed key storage facilities on mobile devices already provide a significant improvement of the security when compared to standard software-based key storage solutions.

Future work will extend CrySIL's feature set and client APIs and heavily focus on extending the flexibility of the authentication system to allow for an even wider range of use cases.

REFERENCES

Benaloh, J., Chase, M., Horvitz, E., and Lauter, K. (2009). Patient controlled encryption: Ensuring privacy of electronic medical records. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, CCSW '09, pages 103–114, New York, NY, USA. ACM.

Bleikertz, S., Bugiel, S., Ideler, H., Nürnberger, S., and Sadeghi, A.-R. (2013). Client-controlled cryptography-as-a-service in the cloud. In *Applied Cryptography and Network Security*, volume 7954 of *Lecture Notes in Computer Science*, pages 19–36. Springer Berlin Heidelberg.

Butt, S., Lagar-Cavilla, H. A., Srivastava, A., and Ganapathy, V. (2012). Self-service cloud computing. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 253–264, New York, NY, USA. ACM.

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, New York, USA. ACM Press.

Fahl, S., Harbach, M., Muders, T., Smith, M., Baumgärtner, L., and Freisleben, B. (2012). Why eve and mallory love android. In *Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12*, page 50, New York, New York, USA. ACM Press.

Kamara, S. and Lauter, K. (2010). Cryptographic cloud storage. In *Lecture Notes in Computer Science*, volume 6054, pages 136–149.

KMIP-v1.1 (2013). Key Management Interoperability Protocol Specification Version 1.1. OASIS Standard.

Lei, S., Zsihan, D., and Jindi, G. (2010). Research on Key Management Infrastructure in Cloud Computing Environments. In *9th International Conference on Grid and Cooperative Computing (GCC)*, pages 404 – 407, Nanjing. IEEE.

Leitold, H., Hollosi, A., and Posch, R. (2002). Security architecture of the Austrian citizen card concept. *18th Annual Computer Security Applications Conference, 2002. Proceedings.*

Reimair, F., Teufl, P., and Zefferer, T. (2015). WebCrySIL - Web Cryptographic Service Interoperability Layer. In *Web Information Systems and Technologies*.

Rocha, F. and Correia, M. P. (2011). Lucy in the sky without diamonds: Stealing confidential data in the cloud. In *41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 129 – 134, Hong Kong. IEEE.

Toegl, R., , Reimair, F., and Pirker, M. (2013). Waltzing the Bear, or: A trusted virtual security module. volume 7868 of *Lecture Notes in Computer Science*, pages 145–160. Springer Berlin Heidelberg.

Trusted Computing Group (2011). TCG TPM specification version 1.2 revision 116. http://www.trustedcomputinggroup.org/resources/ tpm_main_specification. last visited on January 29, 2013.