Injecting Trust to Cryptographic Key Management

Gökhan Bal
Goethe-University
Frankfurt am Main, Germany
bal@cs.uni-frankfurt.de

Andreas U. Schmidt
Fraunhofer Institute for Secure
Information Technology
Darmstadt, Germany
Andreas.U.Schmidt@sit.fraunhofer.de

Nicolai Kuntze
Fraunhofer Institute for Secure
Information Technology
Darmstadt, Germany
Nicolai.Kuntze@sit.fraunhofer.de

***Abstract* — Notwithstanding the adoption of strong cryptographic mechanisms, the anticipated degree of security in the protection and management of privacy sensitive data can only be achieved, if secret keys can be shielded adequately. In practice, most implementations are based on software tokens that shall guard the keys against eavesdropping. This fact alleviates the hardness of circumvention of used cryptographic protocols and with this the disclosure of secret keys. In this paper we propose a key management architecture which – based on the capabilities of Trusted Computing Technology – will provide a higher level of security.**

***Keywords* — Key Management, Browser Security, Trusted Computing.**

1. Introduction

Many applications, whether enterprise or private, have to deal with sensitive data. One major example are web browsers, as they got more and more features by time and have to manage a lot of privacy sensitive data like passwords, cookies, certificates etc. As the misuse of these data could lead to a serious threat to security and privacy, they have to be protected in an appropriate manner. Beneath proprietary protection mechanisms, there exist approaches to centrally manage user credentials with specialized software services. Examples are gnome-keyring [13], KWallet [14] or Apple's Keychain. Aside from these user centric use cases, there also exist application scenarios in the embedded domain, e.g. as studied in the nanodatacenters research project [16]. The purpose of all these solutions is the storage and management of sensitive data like passwords by encrypting them. Since trust to these implementations has its seeds in software modules, the protection mechanisms are still annullable by malware, irrespective to the strength of used cryptographic schemes. This emphasizes the necessity of new approaches to fill this gap. In this paper we will introduce such an approach.

The paper is organized as follows. Section 2 gives a short overview of state-of-the-art protection mechanisms for user credentials. In section 3 the Trusted Computing (TC) Technology specified by the Trusted Computing Group is presented. Section 4 forms the main part, in which our proposed trusted key management architecture is introduced. In section 5 we discuss the architecture in terms of security. Section 6 presents a use case scenario. It follows a short overview of related works in section 7 before we then conclude in section 8.

2. Traditional Approaches

A widespread type of implementation for applications whose actual profession are not privacy and security tasks, introduce proprietary security components to protect sensitive data. A more secure way to achieve this is the engagement of specialized software libraries that implement basic cryptographic functionalities and provide these to application programmers over interfaces. Microsoft adopts the **Cryptographic Service Provider (CSP)** in his applications, which implements cryptographic standards and algorithms. Through the **Cryptography API (CAPI)** these are provided to application programmers.

A widely adopted standard is **PKCS#11**, first published in 1995 by RSA Laboratories [10]. This standard describes an application programming interface, also called **cryptoki** (cryptographic token interface), which provides access to any kind of token that implements cryptographic functionality. Thus, cryptoki abstracts the concrete type of employed token. In that way, different types of cryptographic hardware or smart cards from different vendors can be accessed by application programmers over a single interface, irregardless to the actual interfaces of the different tokens.

In fact, most implementations that use cryptoki utilize a 'virtual' token. In other words, the employed cryptographic token is a software library itself. The obvious reason for this is that smart card equipment or other cryptographic hardware is far from being ubiquitous, especially in private use. Using smart cards in personal computer environments is intricate and special crypto hardware has been costly. Anyway, the advantages of using hardware tokens with protected capabilities is dissolved by using software tokens, because secret encryption keys are not kept in protected capabilities.

3. Trusted Computing

Formed in 2003 as the successor of the abortive Trusted Computing Platform Alliance (TCPA), the **Trusted Computing Group (TCG)** is an organization aiming the establishment of new standards in information security in disparate computing platforms [6]. To this date the TCG has published a series of specifications, which describe the architectures of **Trusted Computing Systems (TCS)**. TC Systems as defined by TCG are computer systems extended by additional components which shall bring **trust** to the computing environment. Trust, in terms of TC, means that components of the system always work as expected. To achieve this goal, the TCG – subdivided into different working groups – has published and is still working on specifications describing architectures, affecting system components at any level from hardware to the operating system.

One of these specifications describes the **Trusted Platform Module (TPM)**. This module, ideally realized as hardware chip hard-wired to the computer platform, plays a key role in building a TCS. It implements basic cryptographic functionality like SHA-1 calculation, message digest creation, random number generation, creation of 2048 bit RSA key pairs and a RSA engine for encryption and signing purposes. Realized as independent hardware module, it can provide protected capabilities in which a shielding of secret data can be done efficiently. Further the TCG defines three different **roots of trust**. These are components of a TCS on which the trust to the whole system is built on. That means that a dysfunction of one of these components would break the entire TCS, because any trust is built upon these components.

One of the roots of trust, the **Root of Trust for Measurement (RTM)** is implemented by the **Core Root of Trust for Measurement (CRTM)**. As an extension of the BIOS, its duty is to perform measurements of system components involved to the booting process. Measured components then can perform measurements of components involved to the next stage of booting. Through this principle of transitive trust, trust can be passed on to the operating system and the software executed in user space. All measured system components together form a so called **Trusted Building Block (TBB)**. Through this architecture it shall be guaranteed that a computer system always starts in an authenticated state that can be verified by an external entity and therefore to spur the establishment of trust.

The second root of trust is the **Root of Trust for Reporting (RTR)**. One of the aims of TC is to enable computer systems to proof other platforms that it is in a trusted state. Therefore the results of measurements of system components have to be presented to the remote platform. To guarantee the genuineness of these data, they have to be signed. For this purpose every TPM contains a 2048 bit RSA key pair, the **Endorsement Key (EK)**, which is generated before shipping. The EK, together with an **EK Credential**, represents the identity of the platform. Representatives, so called **Attestation Identity Keys (AIK)** are used for signatures, for example of measurement results used by the remote party to verify the correctness of the desired state.

The third root of trust is the **Root of Trust for Storage (RTS)**. The intended purpose of the RTS is to establish a secure storage for cryptographic keys or other sensitive data. The RTS is implemented by introducing the **Storage Root Key (SRK)**, a second 2048 bit RSA key pair stored in the non-volatile memory of the TPM. Just like the EK, the Storage Root Key never leaves the shielded location of the TPM. That allows building a hierarchy of keys, with the SRK as the root, in which direct successors are protected by encryption with the SRK. These keys on their part can protect any number of other keys. Thus, trust is bequeathed from the SRK to all successor keys. Any key, following up the SRK can be stored off-chip, not least because memory in TPM is limited, but the number of possible TPM-generated keys is not. Keys never leave the TPM in clear; they are always encrypted by parent keys. The benefit from this is the possibility to work with encryption keys, which in the end are under protection of a hardware module and with this the possibility to encrypt data based on a hardware module.

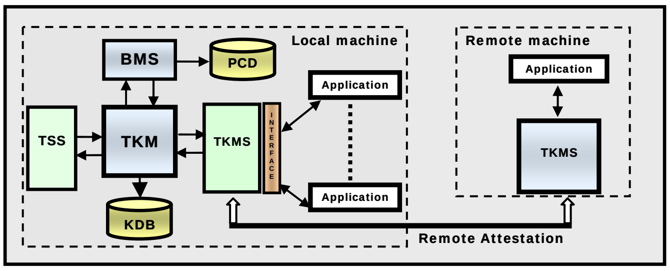

Figure 1. Architecture Overview

To provide a standardized application programming interface to TPM functionalities, the TCG has published the specification for a **Trusted Software Stack (TSS).** At its highest level the **TCG Service Provider Interface (TSPI)** contains all interfaces to enable the utilization of TPM key functionalities.

4. Trusted Key Management Architecture

Considering the weakness and potential vulnerabilities of current solutions for the protection of sensitive data, TC with its roots of trust seems to be a promising approach for designing improved architectures based on a hardware module. This has been an uncomfortable and expensive solution so far because of the circumstances that it comes with.

In this paper we introduce a key management architecture based on the possibilities given by TC Platforms, which extends current solutions by trust and robustness, because of the recoupment of security brought by hardware crypto tokens.

This generic architecture mainly will base on the Trusted Software Stack to implement its functionality. To achieve a maximum level of universality, the services provided by this architecture will cover the needs of all applications dealing with privacy sensitive data. An application programming interface will facilitate the utilization of the services.

Figure 1 gives an overview over the architecture, which consists of several parts. The **Trusted Key Management Service (TKMS)** forms the API to the actual service providers. One of them is the **Trusted Key Manager (TKM)**, which is the main component of the architecture. All essential tasks are performed here. Actually the TKM will call on the Trusted Software Stack to implement its functionality. Another component is the **Backup & Migration Service (BMS)**. Its duty is to perform all the tasks needed to permit protected data to be transferred to another platform or just to be backed up.

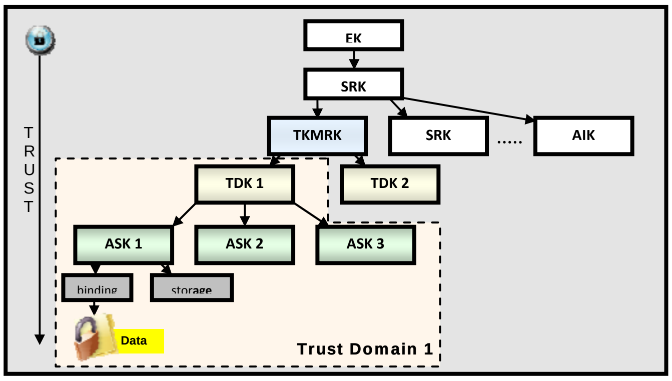

Figure 2. TKM Key Hierarchy

Having in mind that TPM protected keys are bound to a specific TPM and with this to a specific platform, this procedure is not trivial. Remote usage of the TKM is possible by establishing a trusted connection first. Therefore attestation and reporting of systems states of critical components is preconditioned. In the following we describe the single parts in detail.

Trusted Key Management Service

The Trusted Key Management Service can be regarded as a broker, which arranges the communication with applications requesting services. It waits for and accepts incoming requests. Through its interface, applications can request several services to implement their own security features. These include requests for new key pairs, TPM-based encryption and decryption of data and request for a migration or backup of TPM protected data. The crux of the matter is that keys are never passed to the applications. That is concordant to the principle of trusted storage, where any action including a TPM-key is performed inside the protected capabilities of the TPM; the TKMS never catch a sight of the keys.

Trusted Key Manager

The Trusted Key Manager (TKM) is where the actual tasks of the service are performed. Any request the TKMS gets is passed to the TKM in order to execute the needed actions by employing the TSS.

The universality of the service implies that keys and other data of different applications have to be managed. To enable a clear management, the TKM internally handles the **Key Database (KDB)**, in which all data together with additional information about the applications is stored.

To be in line with the key hierarchy built by the Root of Trust for Storage, the TKM actually possesses an own storage key positioned as a direct child of the Storage Root Key (SRK) in the TPM. Based on this key as the root, there will be established a new key hierarchy. The root key, called the **TKM Root Key (TKMRK)**, owns one child storage key per user of the platform to establish separated and independent **Trust Domains (TD)** for each user. At the next level there is one storage key per application that calls on the service. As a result, the TPM key hierarchy is consequently extended as shown in **figure 2**. The TKMRK as the master key of the service protects the keys generated for any user. The user specific key, called the **Trust Domain Key (TDK)** will in the same way protect the **Application Storage Key (ASK)**. The levels beneath the ASK are then affected by the applications themselves. As they are allowed to request as many new key pairs as they want, the hierarchy can arbitrarily grow. The TKMRK, the TDK's and the ASK's are storage keys as defined by the TCG. Keys that follow up the ASK can also be binding keys.

Since the TKMRK is protected by the Storage Root Key of the TPM, all keys employed by the TKMRK are never forwarded to the applications in clear. Thus, the keys are always protected by the TPM and in that way trust comes up to the applications.

Backup & Migration Service

The Backup and Migration Services' (BMS) justification of existence results from the fact, TPM generated keys being bound to that specific platform and migration of keys cannot be handled by just copying them to the destination platform. The Root of Trust for Storages' existence depends on keys never leaving the TPM unprotected. To make migration possible anyway, the TCG has published specifications and protocols that describe how to go about this matter.

In our architecture the BMS is actually an implementation of a TCG compliant migration service for keys and data managed by the Trusted Key Manager. The BMS is responsible for preparing all required steps to execute a migration or backup request. Therefore it generates **Migration Packets (MP)**, which contain all the keys and data which shall be transferred to another (migration) or the same platform (backup). To keep trust while transporting the packet, it has to be wrapped by the destination TKMS' root key before it leaves the host platform. To make this possible, the BMS needs information about the destination platform before it can create the migration packet. This information is kept in the **Platform Credential Database (PCD)**. There, all needed data can be checked up by the BMS. This requires the platform user to enter data about platforms to which migration shall be made possible; otherwise a migration cannot be performed. Migration packets can contain any branch of the key hierarchy within the users' trust domain. In this manner a user has the ability to export his whole private data covering any application or just data of a specific application. With a restore request the MP, respectively the branch of keys can then be integrated to the key hierarchy at the destination platform.

Remote Usage

The TKMS is not limited to be operated in a scenario where client and server run on the same platform. Furthermore the TKMS as the service provider can handle remote requests. In such a scenario it is crucial to establish a secure channel between the participants. We envision that there should be established trust by performing a remote attestation. Both, client and server then can have trust in the integrity and correctness of the partner. Once a connection has been established, the client can make use of the service just like in a local scenario.

5 Discussion

Security implementations can only be accepted if they do not constrain the usability of the system. Our requirement to the introduced architecture was to achieve a higher level of security on the one hand and a minimum of complexity in usage on the other hand. In the following we show that these requirements are fulfilled

Usability

The final level of usability depends on how application programmers implement the user interfaces. The fact is that the proposed architecture allows a higher level of security without making any fundamental changes in how the security is brought to the user. Adoptions have only to be made by application programmers. They have to leave any key management to the Trusted Key Manager.

Security

Since our architecture bases on TC, we first examine general security brought by it, before we then analyze security of our approach in detail.

Trusted Computing Security. Regardless of the public discussion, where TC mainly is presented as an accelerator for better Digital Rights Management (DRM), it shows great promise for the achievement of stronger security in information systems. From the security point of view the shifting of the security basis from software tokens to a hardware module with protected capabilities is an enhancement of robustness, because of its making complicate an attack to the source of protection. The trusted booting architecture including the Core Root of Trust for Measurement ensures that trust is established at an early stage of system execution. Through continuing measurements and the principle of transitive trust, applications can be included to a Trusted Building Block. In client/server scenarios remote attestation enables communication partners to have information about system states of each other. Through evaluation of obtained measurement results, a server can decide whether to trust a client or deny access.

TC is still in an early stage of development. To become widely accepted there first has to be implementations of all components needed to form a TC System. TrustedGRUB [8], the extended version of the GRUB boot loader is a working example of a TCS component. IBM's Integrity Measurement Architecture (IMA) [9] is an implementation of the integrity measurement architecture, in which the integrity measurement of system components can be measured. It is crucial that a trusted operating system exists, which evaluates the measured data and performs all needed tasks to react to the detection of any abnormality in system state. Otherwise the measurements would have no meaning. EMSCB [11] is a first approach to create a trusted OS, but the prototypes are still not ready for use.

Security of the Trusted Key Management Architecture

Assuming the existence of a TC system, the Trusted Key Management Service should be involved to the integrity measurement and with this, included to a Trusted Building Block. In that way trust will reach the Trusted Key Management Service. This trust must not be lost in the run-time of the services. To show that this will not be the case we examine any possibility where a trusted state could potentially be lost.

Trusted Execution. At first step the TKMS will be measured by a trusted operating system and its IMA. Such a system would then deny the execution if any abnormality would be detected. In that way, we guarantee that the service will always start in a trusted state.

Trusted Management of Keys. All keys that will be used by the TKM will be TPM-generated keys. A TCG compliant system will never leave these keys unprotected externally. The key database will only hold encrypted keys. In this way, all keys that will be stored externally are protected by the Storage Root Key of the TPM at last. Operations using these keys will always be performed by the TPM. Any usage of a key has to be authorized by the user (e.g. by providing a password) so that misuse is made difficult. Passwords therefore should never be stored and always be entered by a user to start a session.

Application Security and Privacy. The applications that make use of the service have the ability to encrypt sensitive data without any concerns about how to protect the used keys. The TKM will unload them on this. Applications would have a hardware-based alternative to the widely used software token, which aren't an effective engagement of the PKCS#11 standard.

Remote Attestation. In the client/server scenario a remote attestation has to be performed before a session can start. The Trusted Network Connection specification (TNC) [17] shows how a trusted relationship could be established in such a scenario.

6 Use Case

There are a lot of application scenarios which come into consideration to benefit from the Trusted Key Management architecture. Examples could be encryption software, e-mail clients, home banking software or office applications. As model case we want to demonstrate how security and privacy mechanisms in web browsers can be strengthened by resorting to the TKM.

Modern web browser functionality exceeds the simple displaying of static web content by now. In times of social networking it has to fulfil a series of other functionality that shall relieve the user from caring about the management of a set of personalized data. These data comprise login credentials, cookies, certificates, browser history, form cache and so on. In the majority of cases these data are managed by the browser itself, including the database of secret keys, which are used to protect sensitive data. Mostly the previously mentioned PKCS#11 standard is used by introducing a software token. Offline attacks like explained in [12] show that these protection mechanisms are not adequate to guarantee a high level of security. When seen as an alternative to the software based cryptoki, the TKMS could play the role of a crypto token, which protects and manages secret keys. A web browser could request one key per data class. The browser could then request the encryption of the affected data. In this case the TKMS could be regarded as a digital wallet for private data. When the whole user profile shall be exported to another platform, the browser could utilize the BMS functionality described in section 4. In the users point of view the whole procedures involved in these activities could be made transparent to him. By just adopting the application, the familiar user interfaces and procedures could be kept.

7 Related Work

There have been several works similar to ours that concentrate on the issue of protection of sensitive data or key management. In [2] the authors propose an architecture, in which secret keys of users are protected by describing new architectural and software enhancements for general-purpose processors. They also introduce a Concealed Execution Mode (CEM) and a trusted software library, the crypto operations library (COL) to make use of the crypto processor. In contrast to those the advantage of our proposal is it being based on the Trusted Platform Module (TPM), which on the one hand is already widely distributed and on the other hand enables the measurement architecture, which ensures trust being established at an early stage of system execution. In [3] the authors propose an architecture for a cryptographic server, basing on a special hardware crypto module. Compared to such approaches, TPM-based architectures have low hardware equipment costs. Hardware crypto modules are useless if the overlaying software libraries can't be trusted. So another advantage of our system is that it is included to the integrity measurement of a TC System. In [5] the author shows why smart cards and the Trusted Platform Module have complementary roles and how these two can be combined.

There are also initiatives that deal with architectures for an efficient key management. The **P1619.3 Key Management** working group of IEEE SISWG is currently working out a specification which describes an architecture for the key management infrastructure for cryptographic protection of stored data, describing interfaces, methods and algorithms [7]. The specification focuses on the description of key objects and how they can be managed through their whole life-cycle. Up to date the specification is still in process. The second initiative is the **Trusted Storage Key Management Services Subgroup (KMSS)** of the TCG. Its mission is to develop a uniform approach to manage keys across a variety of storage devices [15]. To avoid overlap the two working groups have decided to cooperate.

8 Conclusion

The issue of protection of privacy sensitive data is more important than ever, since fatal security gaps led to the massive loss of personal data in the last years. But for all of that the protection mechanisms used by the most of the systems still base on software libraries which are very vulnerable to malware, so that integrity of these mechanisms cannot be guaranteed. In this paper we proposed an architecture to overcome this issue. It bases on the TC Technology, which due to the spread of the TPM promises to play a vital role in future computing systems. To make full use of solutions like ours that base on TC, there first have to be fulfilled the preconditions of a Trusted Computing Base including a trusted operating system. Works on these are still in progress. Because of its basis on the TPM, this architecture promises a higher level of security to applications as it is the case in currently used software token approaches like the usage of the PKCS#11 standards. Being application independent, the service can be used by any kind of application with security and privacy needs so that applications need not to deal with the protection and management of secret keys and data.

Further research in this area will be centered in applying TC protection for credentials in embedded systems. There various stakeholders are operating based on one device physically installed in potential hostile environments. Thus, the protection of the credentials is vital for the belief of the stakeholders in the respective solution to put trust into it. This work will be part of the afore mentioned NanoDataCenters project.

References

[1] Jackson, C., Bortz, A., Boneh, D., and Mitchell, J., "Protecting browser state from web privacy attacks", WWW'06: Proceedings of the 15th international conference on World Wide Web, ACM, 2006, pp 737-744

[2] McGregor, J.P., R.B. Lee, "Protecting Cryptographic Keys and Computations via Virtual Secure Coprocessing", SIGARCH Comput. Archit. News, ACM, 2005, 33, pp.16-26.

[3] Rong, X., X. Gao, R. Su, and L. Zhou, "Design and Implementation of a Parallel Crypto Server", Lecture Notes in Computer Science – Computational Intelligence and Security, Springer Berlin/Heidelberg, 2005, 3802/2005, pp. 398-406.

[4] Sánchez, D.D., A.M. Lopez, and F.A. Mendoza, "A Smart Card Solution for Access Control and Trust Management for Nomadic Users", Lecture Notes in Computer Science – Smart Card Research and Advanced Applications, Springer Berlin/Heidelberg, 2006, 3928/2006, pp. 62-77.

[5] Aussel, J.D., "Smart Cards and Digital Security", Computer Network Security, Springer Berlin Heidelberg, 2007, 1, pp. 42-56.

[6] Trusted Computing Group, TCG Architecture Overview, Version 1.4, 2007

[7] IEEE SISWG, P1619.3/D5 Draft Standard for Key Management Infrastructure for Cryptographic Protection of Stored Data, 2008

[8] TrustedGRUB. `http://trustedgrub.sf.net`

[9] Sailer, R., X. Zhang, T. Jaeger, and L. van Doorn, "Designing and Implementation of a TCG-based Integrity Measurement Architecture", Proceedings of the 13th USENIX Security Symposium, San Diego, California, August, 2004, pp. 223-238.

[10] RSA Laboratories, PKCS#11 v2.20: Cryptographic Token Interface Standard, 2004

[11] European Multilaterally Secure Computing Base. `http://www.emscb.com`

[12] Felker, M., Password Management Concerns with IE and Firefox. `http://securityfocus.com/infocus/1882, 2006`

[13] GNOME Keyring. `http://live.gnome.org/GnomeKeyring`

[14] KWallet. `http://docs.kde.org/stable/en/kdeutils/kwallet/index.html`

[15] Trusted Computing Group,Trusted Storage Key Management Services Subgroup FAQ, 2007

[16] Nanodatacenters EU FP7 research project. `http://www.nanodatacenters.eu`

[17] Trusted Computing Group, TCG TNC Architecture for Interoperability, Version 1.3, 2008