

D6.3.1 Advancement and integration of concepts for secure and dynamic creation of Mobile Services

Editors:	Marc-Michael Bergfeld (GD) Stephan Spitz (GD)
Reviewers:	Aleksandra Kuczerawy (KUL) Karel Wouters (KUL)
Identifier:	D6.3.1
Type:	Deliverable
Class:	Public
Date:	May 1 st , 2011

Abstract

This document presents the challenges of Identity Management and Services on Mobile Devices, especially with regard to open systems, collaboration and privacy between Front-end Mobile Devices and Back-end Servers. It looks into existing technologies for static and increasingly flexible Mobile Services and introduces future technology paths for highly dynamic creation of Mobile Services, bearing the increasingly important aspect of Security for such Mobile Services in mind.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 216483 for the project PrimeLife.

Members of the PrimeLife Consortium

1.	IBM Research GmbH	IBM	Switzerland
2.	Unabhängiges Landeszentrum für Datenschutz	ULD	Germany
3.	Technische Universität Dresden	TUD	Germany
4.	Karlstads Universitet	KAU	Sweden
5.	Università degli Studi di Milano	UNIMI	Italy
6.	Johann Wolfgang Goethe – Universität Frankfurt am Main	GUF	Germany
7.	Stichting Katholieke Universiteit Brabant	TILT	Netherlands
8.	GEIE ERCIM	W3C	France
9.	Katholieke Universiteit Leuven	K.U.Leuven	Belgium
10.	Università degli Studi di Bergamo	UNIBG	Italy
11.	Giesecke & Devrient GmbH	GD	Germany
12.	Center for Usability Research & Engineering	CURE	Austria
13.	Europäisches Microsoft Innovations Center GmbH	EMIC	Germany
14.	SAP AG	SAP	Germany
15.	Brown University	UBR	USA

Disclaimer: The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The above referenced consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law. Copyright 2011 by Unabhängiges Landeszentrum für Datenschutz, Giesecke & Devrient GmbH, Europäisches Microsoft Innovations Center GmbH, SAP AG.

List of Contributors

This deliverable has been jointly authored by multiple PrimeLife partner organisations. The following list presents the contributors for the individual parts of this deliverable.

Chapter	Author(s)
Executive Summary	Marc-Michael Bergfeld (GD), Stephan Spitz (GD)
First Chapter	Marc-Michael Bergfeld (GD), Stephan Spitz (GD)
Second Chapter	Marc-Michael Bergfeld (GD), Stephan Spitz (GD), Stuart Short (SAP), Ulrich Pinsdorf (Microsoft)
Third Chapter	4.1 Stuart Short (SAP), Ulrich Pinsdorf (Microsoft), 4.2 Marc-Michael Bergfeld (GD), Stephan Spitz (GD), 4.3 Harald Zwingelberg (ULD), Marit Hansen (ULD)
Fourth Chapter	Marc-Michael Bergfeld (GD), Stephan Spitz (GD)

Executive Summary

This paper elaborates upon the advancement and integration of concepts for secure and dynamic creation of Mobile Services in the course of the PrimeLife project.

It presents the current status and challenges for Secure and Dynamic Services on Mobile Devices by looking into the role of increasingly open systems, collaboration and privacy between Front-end Mobile Devices and Back-end servers (see section 1.1).

Herein, Secure Elements as technologies for the provisioning of secure, the management of various (partial) identities and the enhancement of privacy aspects are seen to be critical (see section 1.2).

These Secure Elements, however, will increasingly need to correspond to highly dynamic and flexible offerings that leverage both Mobile Front-end and Server Back-end infrastructures (see section 1.3).

Two emerging technologies are highly capable of corresponding to the requirement of ever more flexibility, without sacrificing security, identity and privacy: Secure Micro SD cards and Trusted Execution Environments (see section 1.4).

Both technologies have been developed further in the course of the PrimeLife project – for example in the eCV prototype which allows for flexible interaction between Front- and Back-end in the case when privacy-relevant policy mismatches occur in the service composition.

These technologies are explained in detail (see section 2.1) and particularly the capabilities of the Trusted Execution Environment are laid out in detail (see section 2.3). Technical specifications of the Trusted Execution Environments are attached in the appendix.

The integration of the Front- and Back-end is explained based on the PrimeLife eCV scenario and the results of the PrimeLife eCV demonstrator (see section 3.1 and 3.2). In this domain, the research of the PrimeLife project has already led to the creation of a unique solution in which the individual person can now reach into the service composition / service-oriented architecture of the Back-end in a secure, identity-aware and private manner ad hoc via the mobile Front-end.

Further, newly arising legal aspects of identity management and privacy enhancement in secure and dynamic Mobile Service are presented and put into perspective of future developments (see section 3.4).

A roadmap of solved and unsolved issues with regards to security, identity management and privacy enhancement in Mobile Services is presented (see section 4.1).

A conclusion is drawn – also reflecting upon the results of the PrimeLife project with regards to infrastructure for identity management and privacy enhancement and future directions of research are indicated (see section 4.2).

PrimeLife's funding supported G&D in its efforts by nurturing the development of the conceptual research regarding the topic, the development of a corresponding demonstrator (see PrimeLife Deliverable 6.2.2), the standardization of the findings in the Global Platform Consortium and the preparation of translating the research findings into a real-world product technology.

Contents

1. Introduction: Present status and challenges for Secure and Dynamic Services on Mobile Devices	7
1.1 Open systems, collaboration and privacy between Front-end Mobile Devices and Back-end servers	7
1.2 Present Secure Elements as Enablers of static and flexible Mobile Services	8
1.3 The present and future environment of Mobile Services: From static to flexible and highly dynamic solutions	10
1.4 Technologies for Secure and Dynamic Mobile Services and the privacy challenge in highly dynamic environments	11
2. Front- and Back-end Technologies for Dynamic Mobile Services	13
2.1 Front-end Technologies	13
2.2 Trusted Execution Environments as Next Generation Mobile Platforms and Service Enablers	15
2.3 TEEs, Open Interfaces and first APIs	16
3. Integration of Front- and Back-end for Secure creation of Dynamic Mobile Services	19
3.1 The collaboration of Front- and Back-end for secure and dynamic Mobile Services	19
3.2 The mobile Front-end as service delivering environment	21
3.3 The server Back-end as service processing counterpart	22
3.4 Future legal requirements for secure and Dynamic Mobile Services	23
3.4.1 Considering security and privacy protection goals	24
3.4.2 Other legal requirements.....	25
4. Outlook and Conclusions: A Roadmap for SE-based Privacy in Secure and Dynamic Mobile Services	27
4.1 A Roadmap of solved and open issues for SE-based privacy	27
4.2 Conclusion and outlook towards future research	28
References	30
A. TEE Client API Description	33
A.1 Overview of the TEE Client API as standardized in Global Platform.....	33
A.2 Scope of the standardization of the TEE Client API in Global Platform.....	33
A.3 TEE Client API Architecture	34
A.4 TEE Client API Principles and Concepts	35
A.4.1 TEE Client API Design Principles.....	35
A.4.2 TEE Client API Fundamental Concepts	36
A.4.3 TEE Client API Usage Concepts	42
A.4.4 Security	43

List of Figures

Figure 1: Various Secure Elements as “Private World” in Mobile Devices	9
Figure 2: Differentiating between static, flexible and highly dynamic Mobile Services.	10
Figure 3: Dynamics, Security and Privacy relating to the analyzed technologies	12
Figure 4: Exemplary Secure Element options for “Normal” and “Private Worlds”	14
Figure 5: Overview of the “Public” and “Private World” and the Interfaces in TEEs.....	15
Figure 6: Trusted Input + Output = Secure Private User Dialog.....	17
Figure 7: Interaction via the “Private World” of the Mobile Device: Example from the eCV Demonstrator.....	21
Figure 8: A Roadmap of Topics for secure, private and identity-related Mobile Services.	28
Figure 9: The full Integration of Front-end and Back-end for Secure Dynamic Mobile Services..	29
Figure 10: TEE Client API System Architecture as standardized in Global Platform.....	34
Figure 11: TEE Client API Shared Memory Buffer Lifetime.....	38
Figure 12: TEE Client API Memory Reference timing diagram	40

Chapter *1*

Introduction: Present status and challenges for Secure and Dynamic Services on Mobile Devices

1.1 Open systems, collaboration and privacy between Front-end Mobile Devices and Back-end servers

Open systems, mobile Applications and the increasing collaboration across individuals and groups, based on modern technology platforms and solutions are giving rise to the dynamic creation of new services, especially through the Application of Mobile Devices such as Mobile Phones, Netbooks, Tablet PCs or even cars and their interaction with Back-end Servers in Service-Oriented Architectures (SOA).

At present, these dynamics meet with an infrastructure that

“[...]was built without a way to know who and what you are connecting to. This limits what we can do with it and exposes us to growing dangers. If we do nothing, we will face rapidly proliferating episodes of theft and deception that will cumulatively erode public trust in the Internet.” [CK05].

Hence, the opportunities of open collaboration, such as ad hoc access and exchange of information and knowledge (e.g. the rapid sharing of data, usage of different identities across various peers) and the possibilities of combining Mobile Devices at the Front-end of customer interaction with the Backend-servers' (i.e. in the “Cloud”), are today interrelating with increasing challenges for security, privacy and identity management, because “The Web Means the End of Forgetting” [RJ10]. Examples of these challenges are:

- Providing Trusted Platforms for the execution of services between front- and back-end.
- Providing dedicated channels of communication, storage and interaction for partial identities of individuals.

- Securing the interactions against attacks and intervention.
- Providing solutions of anonymity where applicable without jeopardizing authentication.

This document elaborates on the recent conceptual and technological developments (e.g. in the PrimeLife project) and the hence arising opportunities for the dynamic creation of services with Front-end Mobile Devices and Back-end Servers, *including* the use of security, identity-management-enabled and privacy-enhancing technologies (in abbreviation: SPI technologies).

In this domain, the research of the PrimeLife project has already led to the creation of a unique solution in which the individual person can reach into the service composition / service-oriented architecture of the Back-end in a secure and private manner ad hoc via the mobile front-end (see section 4.2, also see Deliverable 6.2.2.). This is a highly novel Application of the technological concepts described in this deliverable.

This document has following structure:

- Chapter 1 details the present market and technology environment, and the privacy challenges in this context.
- Chapter 2 presents Front- and Back-end technologies for Secure and Dynamic Mobile Services.
- Chapter 3 elaborates on the integration of Front- and Back-end for Secure and Dynamic Mobile Services.
- Chapter 4 concludes the present situation and sketches paths for future research.

1.2 Present Secure Elements as Enablers of static and flexible Mobile Services

Secure Elements (SEs) are platforms, esp. for Mobile Devices, on which Applications can be installed, personalized and managed. Increasingly, this can be done over-the-air (OTA). With recent¹ technology developments, OTA provisioning of Applications is done via a Trusted Service Managers (TSM). This helps to adapt formerly static SEs more flexibly for new Mobile Services and Applications.²

Further, SEs are seen to potentially provide a “safe resort” for value-intensive, critical Applications which using significant professional and private data, especially as the environment for Mobile Devices and the services provided via these is increasingly challenged with risks of data theft [BM10], espionage [RC10], security breaches [WV10].

On a conceptual level, SEs can be categorized into three different areas:

- Removable SEs (e.g. Stickers, Secure Micro SD cards and UICCs³)
- Non-removable SEs (e.g. embedded SEs)
- SEs from a combination of software programs on dedicated hardware (e.g. Trusted Execution Environments).⁴

¹ Note: Recent referring to the introduction of such services from 2006/7 onwards.

² Note: For practical examples, see e.g. www.venyon.com or www.smartrust.com

³ Note: A UICC is a UMTS Integrated Circuit Card, i.e. a type of Subscriber Identification Module (SIM) used in 3G UMTS devices.

⁴ Note: In the case of the TEE, the SE consists of a physical module, e.g. a partition of the CPU and software embedded into this physical module (e.g. a secure Operating System). For a detailed elaboration on the different categories of SEs, see Mobey Forum (2005), p.4f.

The history of Secure Elements and the capabilities of Smart Cards and Tokens in the context of Secure Dynamic Mobile Services has already been analysed elsewhere [BHS08]. Further, the security in embedded systems and the different virtualization technologies have been analyzed and the usability aspects of Secure Environments have been commented upon, and the applicable cryptography have been revised [SSP08].

In essence, it has been shown that SIM cards, for example, have advantages for, operator-specific, static and highly secure identification tasks [BHS08]. Embedded security systems and virtualization technologies, in comparison, are more applicable for highly dynamic service provisioning [BHS08].

In short,

“Smart Cards and Tokens provide high security in a mobile and flexible manner. Embedded Security Mechanisms and Virtualization may provide significant processing power for security relevant Applications and that the subsequent combination of these still independent capabilities could be combined into “a system which can cover all levels of security, be static as well as flexible and highly performing. As an example, such systems could provide Smart Cards or Tokens for Mobile Devices which can store different identities and assist in using selected ones of these for different services such as payments, bookings or participation in online communities. The Embedded Security Mechanism would assist in decoding and processing the data stored on the Smart Card or Token, thus making the overall system secure and highly performing.” [BHS08]

In addition to the well established Smart Cards / SIMs / UICCs, some additional Secure Element technologies have emerged successfully in the Mobile Ecosystem, whilst others have not spread so widely. As a result, the present context of SE technologies for Mobile Devices looks as follows:

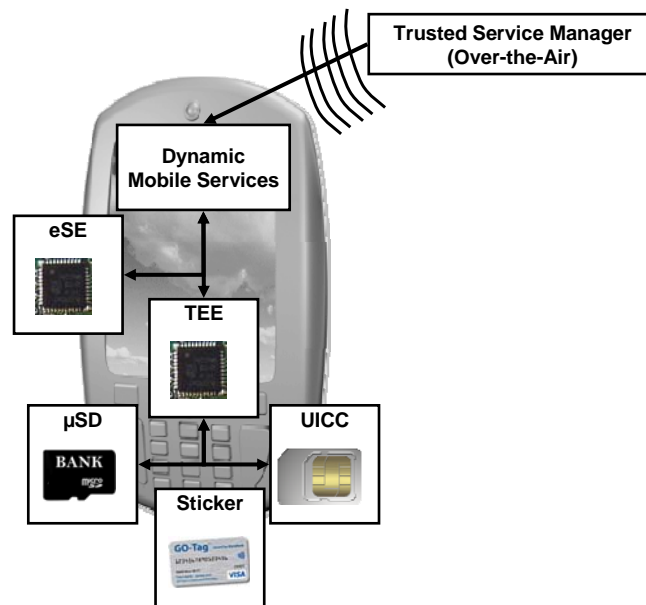


Figure 1: Various Secure Elements as “Private World” in Mobile Devices

A selection of the above mentioned SEs also supports increasingly Dynamic Mobile Services – without sacrificing security.

Complementary technologies such as Secure µSD cards, Stickers and selected embedded Secure Elements have seen rather wide uptake in the market, because they enabled new Mobile Services

in the Value Chain for established stakeholders and / or were accessible to new players for the establishment of new Mobile Service concepts.⁵ Secure Micro SD cards and the Trusted Execution Environment are of particular relevance as they combine increased security with increased flexibility. Because these two SEs can also be used as a storage and processing platform for the identification of individuals and their credentials, they are particularly relevant for privacy-enhanced and identity-management-enabled services that need to be highly secure and flexible. Further, they offer largely open interfaces within their architecture in order to promote a rapid uptake by existing and new stakeholders along the Mobile Services value chain.

On the other hand, initiatives to embed Trusted Platform Modules into Mobile Devices have not succeeded on a wide basis. Apparently, the economic incentive for the different stakeholders along the value chain of the Mobile Services industry remained unclear and fragmented business interest along the value chain were not orchestrated for a systemic solution [see MF10]. Neither has the potential option to integrate an additional Smart Card reader into Mobile Devices found wide acceptance.⁶

1.3 The present and future environment of Mobile Services: From static to flexible and highly dynamic solutions

The situation sketched in section 1.1 has led to a shift from technologies that relate to rather static Mobile Services towards those technologies that empower a highly dynamic, complex and rapidly changing environment of Mobile Services. Security, privacy and identity management solutions need to be tailored to this environment.

In Detail:

Initially, Mobile Services were embedded into *static* environments: Fixed and concrete client and server components, actors and scenarios constituted the service sphere: Fixed security requirements for given scenarios resulted. Time was sufficient to develop and modify client and server components when new security, privacy and identity challenges arose. Technologies for this scenario exist in the form of UICC cards and Stickers, for example.

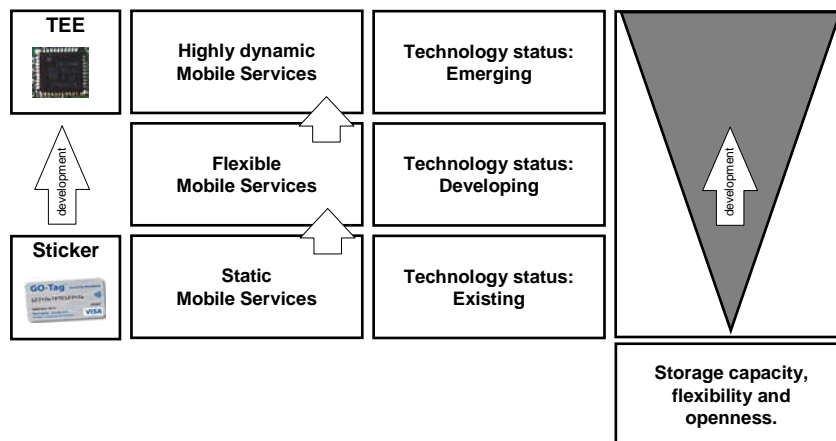


Figure 2: Differentiating between static, flexible and highly dynamic Mobile Services.

⁵ Note: Also see [MF10] for a detailed analysis of the Mobile Value Chain –with particular attention to Mobile Financial Services.

⁶ Note: The additional effort and costs for the handset manufacturer, who largely operates based on the requirements of the Mobile Network Operators (MNOs), were not consistently required and called for along the Mobile Services Value Chain. Similarly, a requirement for an additional Smart Card reader was not commonly agreed upon from all players in the Value Chain.

Subsequently, and at present, Mobile Services need to be *flexibly* embedded into increasingly changing environments: Heterogeneous scenarios needed to be addressed. Security rules are required in order to embed security, privacy and identity management aware behaviour in the equipment that can be context-aware and follow the overarching rules under different situations. Hence, the security, privacy and identity management “co-evolves with the isotropic and steadily changing context into which it is embedded” [BHS08]. Technologies for this scenario are developing, for example in the form of Secure μ SD Cards.

Increasingly at present, and even more so in the future, Mobile Services will need to be highly *adaptive* to ever changing, dynamic environments. “This will consider unknown equipment, actors, and heterogeneity of space. The definition of SoS will result in *security policies*. The client and the server will know the policies [..., and] take a “flexible and secure”, “pervasive and secure”, “resilient and secure”, “recoverable and secure” character, depending on the situation.” [BHS08]. Technologies for this scenario are emerging in the area of Trusted Execution Environments, for example.

Figure 1 exemplifies how the present, developing and emerging technologies fit to these different aspects of the Mobile Service environment. The below presented, existing technologies of, for example, Static Stickers and Secure μ SD cards both enable static and flexible Mobile Services respectively. The need for higher flexibility, capacity (i.e. processing speed and storage) and a predominantly open environment for Mobile Services, which can then empower security, privacy-enhancement and identity management, is largely expected to be answered by Trusted Execution Environments (TEEs) as emerging technology (see section 3).

Analyzed in detail, the following matching of privacy challenges and present, developing and emerging technologies can be summarized:

1.4 Technologies for Secure and Dynamic Mobile Services and the privacy challenge in highly dynamic environments

Figure 3 below exemplifies how the three technologies analysed in this paper correspond to the various building blocks of highly dynamic and secure Mobile Services, and where open issues for identity management and privacy enhancement remain. Exemplarily, figure 3 summarizes which subsections of secure, privacy-enhanced and identity-enabled Mobile Services are provided.

It becomes evident that static technologies such as the Passive Stickers provide a Secure Element for selected Mobile Services (e.g. NFC functionalities of Credit Card payments), but do not correspond to an environment that would call for highly flexible provisioning of partial identities. As Passive Stickers, they have one or a set of preinstalled identities (e.g., a credit card number), but cannot be provided with new, partial identities over-the-air in a flexible manner.

Further, privacy is only partially provided as the Passive Sticker would interact with a terminal which would then route the communication through an additional network. Hence, there is no “private” end-to-end communication between the individual and the recipient of the message (e.g. a payment via Credit Card), but processing networks are involved in “getting the message across”.

For Secure μ SD cards, flexibility is given and partial identities as well as privacy can be assured. Here, partial identities can be provided over-the-air for different relationships and audiences (e.g. a partial identity for travelling with frequent flyer programs, another partial identity for interaction with a bank, a third partial identity for customer loyalty programs, et cetera). Further, these partial identities can be combined with unique keys at both ends of the communication channel (i.e. VPN-like architectures), so that a communication channel which is linked to one of the partial identities remains “private” because it is encrypted and can only be read by the counterpart for this

partial identity and not the processing network in between (Nota bene: Also see the PrimeLife demonstrator with the PrimeLife Application running in a Privacy-PIN protected Private World on a μ SD Card, encrypting and decryption messages to a specific head hunter account). This could, for example, be used for the provisioning of “private” health information from an insurance company to a patient via mobile phones, which is impossible today in the U.S. because privacy cannot yet be assured.




	Sticker 	μSD 	TEE 
Highly dynamic	No	Partially	Yes
Trust: A Trusted Secure Element / Environment	Yes	Yes	Yes
Identity: A specific communication channel for the partial identity	No	Yes	Yes
Privacy: Secure communication, only for the individual	Partially	Yes	Yes
Anonymity: Unlinkability of the interaction to the individual	Possibly	Possibly	Possibly

Figure 3: Dynamics, Security and Privacy relating to the analyzed technologies

For TEEs, most categories are fulfilled in the same manner as for the μ SD cards. In addition, TEEs can provide a direct link to the hardware, e.g. a mobile phone and its keypad and display, and can thus assist in even making the input and output of information trustworthy, private and identity-related. For example, a secure User Interface (UI), which includes a secure display and keypad, will assure that the input, e.g., an amount for a money transfer or the need for a new medicine prescription, can only be read by the respective Application that is linked to the partial identity / e.g., a bank account), is then encrypted and wired through the network to the recipient in a privacy-enhanced manner.

For all technologies, however, anonymity as one additional building block for enhanced privacy remains an open issue: For example, Mobile Devices have unique identities in the networks over which they communicate, provided by the log-on of the SIM card (i.e. the subscriber identification module) to the networks, based on the corresponding Personal Identification Number. Hence, the network can identify the individual Mobile Device. Further, the usage profile correlated to the respective device provides further insight into the individual’s preferences and interests. Thus, although the above mentioned technologies can fully or partially provide trusted, identity-related and private means of communication and interaction, the individual device, and therefore also its user, are not anonymous.

The following chapter looks at the emerging technology environment for mobile devices and reflects upon their integration with back-end technologies in the context of identity management and privacy-enhancements.

Chapter 2

Front- and Back-end Technologies for Dynamic Mobile Services

2.1 Front-end Technologies

In order to comply with the above described dynamics in the market and technology environment for Mobile Service, Secure Elements need to be highly standardized, modularized and highly flexible / adaptive to ever changing requirements. These characteristics will drive their rapid and wide distribution.

In order to achieve such distribution and keep costs low, independently of the eventual usage of the Secure Elements, it will be necessary to embed them into the platforms of the above mentioned Mobile Devices ex ante. For this, technologies such as the ARM TrustZone®⁷ may be leveraged, because of their dominant design in the market place for Mobile Device platforms.

Further, the appeal of these Secure Elements will be particularly high to the respective Mobile Service providers if they rapidly, easily and seamlessly integrate with Applications provided by Third Parties in the market place. Quick diffusion can be expected, if the SEs in the Front-end enable these Third Parties to embed security, privacy and identity-management into their solutions ad hoc. Further, a pre-certification of the Secure Elements with regards to security, privacy and identity-management will additionally enhance market acceptance, because it would provide independent solution and Application providers with a “dock-on” method to security, privacy and identity-management. To achieve this goal, clear and open interfaces will be essential.

Most recent developments in the SE environment provide two solutions: Micro SD cards, as a physically removable option to provide “Private Worlds” on Mobile Devices and TEEs as built in – and thus widely diffused – architectures to provide “Private Worlds” too.

From a technology architecture point of view, both solutions are structured in a similar manner:

⁷ Note: ARM TrustZone® technology is a system-wide approach to security on high performance computing platforms for a huge array of Applications including secure payment, digital rights management (DRM), and web-based services.

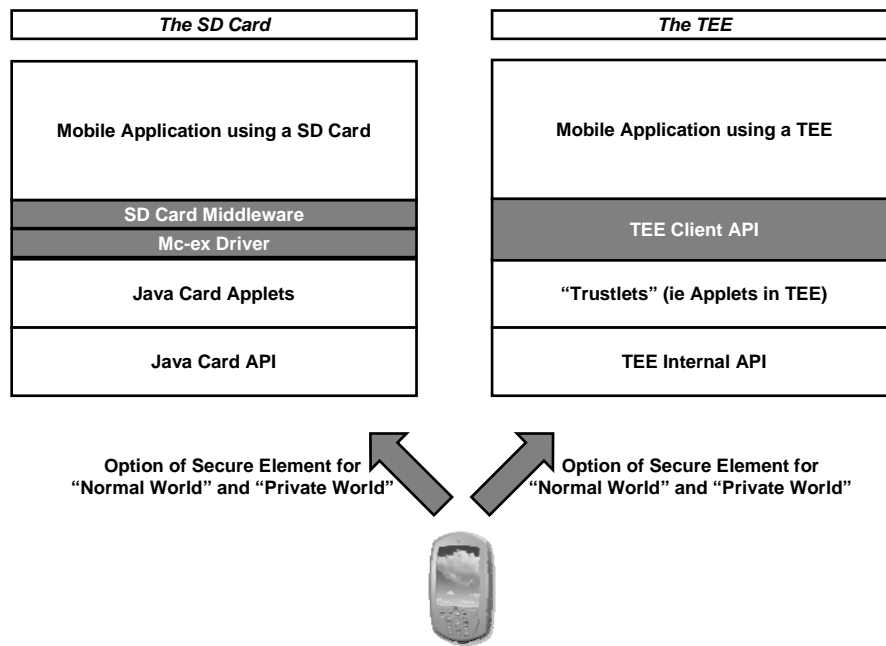


Figure 4: Exemplary Secure Element options for “Normal” and “Private Worlds”

In essence, the following characteristics apply:

- Mobile Applications using a SD Card leverage the SD Card Middleware and a Mc-ex Driver to connect to Java card Applets.
- Mobile Applications using a TEE leverage the TEE Client API to connect to “Trustlets”, i.e. Applets in the TEE (see Appendix for the full technical specification of the Client API).
- Both, Applets in the SD Card as well as Applets in the TEE leverage an internal API (Java Card API or TEE Internal API) to connect further.

Hence, μ SD cards as well as TEEs provide the necessary clear, open and scalable structure which is necessary to quickly and flexibly adapt to the highly dynamic market and technology environment of Mobile Service, as described above:

- Different Mobile Service providers can provide secure, identity-enabled and privacy-enhanced Mobile Applications to the μ SD or the TEE over-the-air.
- The interfaces of the μ SD or TEE can provide a “wall” behind which essential data, partial identities and access keys are protected in a “PrivateWorld”.
- The “Private World” can be certified according to security standards and requirements of different industries and legal environments (e.g. health, payment & banking, headhunting services as in the PrimeLife demonstrator, or loyalty programmes).

Based on this similar architectural structure, the technologies and demonstrators for one of these SE options can resemble and provide insights for the other Secure Elements as well. In the PrimeLife project, the “eCV scenario” (see [PU11]) has been implemented on a Secure μ SD Card. This demonstrator provided insights and a first example on how “Private World” Applications work, based on available, most recent technology.

In direct comparison, TEEs are expected to provide even higher flexibility, more storage, higher processing speed and smarter APIs than the presently existing Secure μ SD card solutions. In contrast though, TEEs will not be physically removable from the Mobile Devices, need integration

into the very beginning of the Mobile Services value chain (i.e. through the Silicon producer of the CPU), and their proliferation throughout the Mobile Device platforms will take additional time. Hence, it is expected that Secure μ SD cards and TEEs will coexist in the context of Mobile Services, but will both be capable of serving the increasingly dynamic market and technology context.⁸

For the sake of conceptual and technological advancement, the following sections will focus on the TEE and its integration into secure, identity-enabled and privacy-enhanced Mobile Services between Front- and Back-end:

2.2 Trusted Execution Environments as Next Generation Mobile Platforms and Service Enablers

As introduced above, TEEs provide security, privacy-enhancement and identity-management solutions that enable new types of services. TEEs address the need for flexible, powerful and efficient security solutions in various forms of Mobile Devices. Amongst others, TEEs can, for example, be based on ARM TrustZone enabled chipsets (i.e. SoCs). TEEs utilize ARM TrustZone’s division of the SoC into two distinct areas, so to speak a “Public World” and a “Private World”, as shown in the below figure. TEEs then provide open interfaces in order to enable the development of dedicated Applications with security, privacy and identity-management capabilities.

In this concept of “Public and Private Worlds”, the TEEs encapsulate security-, privacy and identity-management-relevant parts of an Application in the dedicated “Private World”. Those parts of the Application that are not security-, privacy- or identity-management-relevant remain in the “Public World”.

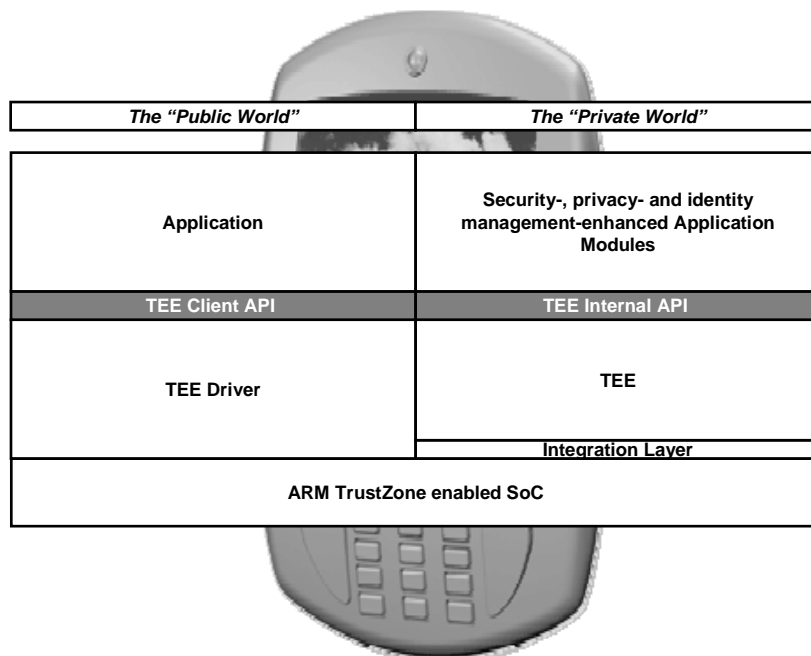


Figure 5: Overview of the “Public” and “Private World” and the Interfaces in TEEs

⁸ Note: TEEs can be set into perspective with Trusted Platform Modules (TPMs). In contrast to TPMs, though, TEEs do not add additional hardware costs.

Two clear and open interfaces between the “Public” and the “Private World” – the TEE Client Application Protocol Interface (API) and the TEE Internal API – enable Application providers to dock-on to the concept (see appendix for full technical specifications empowers them to offer secure services to the market without having to go into the details of security and privacy protection or the specifics of identity-management).

It is important to explain these two interfaces, because their characteristics have a direct effect on how the Back-end can relate to the Mobile Devices, delivering the Mobile Service to the end consumer.

2.3 TEEs, Open Interfaces and first APIs

The TEE Client API (see see appendix for full technical specifications) and the TEE Internal API essentially build the “wall” between the “Private” and the “Public” world.

Both follow a lightweight approach, meaning they are easy to use and easy to understand. Hence, developers can concentrate on the design of their business logics. TEEs are also integrated into different SoCs in order to diffuse quickly to the different Mobile Devices. In order to limit the complexity of security, privacy-enhancement and identity-management requirements for the developers, the TEEs’ two interfaces are also in the process of standardization in the Device Committee of the Global Platform initiative.⁹ In detail, they work as follows:

TEE Client API¹⁰

The TEE Client API enables the exchange of data between the rich OS Applications of Mobile Devices and the security- and privacy-enhanced part of the Application. Based on the standardized and open TEE Client API, developers are able to define which data shall be interchanged between the “Public” and the “Private World”, and in which manner. The interface is optimized for simplicity, runtime performance and ease of use.

TEE Internal API

The TEE Internal API offers Applications that leverage security-, privacy- and identity-management-related modules (i.e. “Trustlets”) the option to access the TEE itself. The TEE Internal API is very similar to industry-proven interfaces. This reduces the efforts for development and maintenance of the respective Applications.

In short, these two interfaces and the concept of TEEs offer a highly modularized, simple and easy-to-use Secure Element on Mobile Devices which empowers rapid deployment and constant adaptation of security-, privacy- and identity-management-enhanced solutions for e.g. Mobile Phones, Netbooks, Tablet PCs or even Cars.

From a security perspective, the following considerations have been taken into account in the development of TEEs:

TEE Security Considerations

In order to complement the existing Secure Elements in an adequate way, TEEs are characterized by the following points:

- High performance
- Low Footprint
- Provable Security

⁹ Note: For details, see: <http://www.globalplatform.org/aboutuscommitteesdevice.asp>

¹⁰ For technical description, see Appendix

- Certifiability
- High performance and low footprint

The exchange of data between the “Public” and the “Private World” is optimised to exchange large blocks of data and hereby create as little communication overhead as possible. Furthermore, highly performant TEEs are optimized to have very small footprints.¹¹

Provable security

The security level of TEEs is balanced to provide an ideal distribution of security, performance and flexibility. As most Applications on Mobile Devices do not need to be fully tamper resistant, e.g. against invasive attacks, TEEs are targeted at a mid-range security level (i.e. slightly lower than the static solutions of, e.g. Smart Cards). In return, however, TEEs offer high performance, flexibility and also storage capacity. In the future, TEEs are expected to be fully certified environments which provide provable security (also see the EU-funded Sepia Project, www.sepia-project.eu).

Certifiability

Being based on the above mentioned, standardized and clear interfaces, TEEs can be driven towards security and privacy certification more easily than other Secure Elements.

TEEs and Secure User Dialog Looking at modern Applications on Mobile Devices and their value for the individual User (e.g. mobile banking Applications, mobile social networking and mobile loyalty programs etc.), protecting these interactions and assuring the adequacy of the information exchanged via Mobile Devices becomes increasingly clear.

At present, two security gaps remain for Mobile Devices, especially Mobile Handsets: The input of data in a trusted manner (e.g. without interference between the act of typing on the keyboard or on the touch screen) and the output of data in a trusted manner (e.g. without the display of manipulated data over the screen of Mobile Devices):

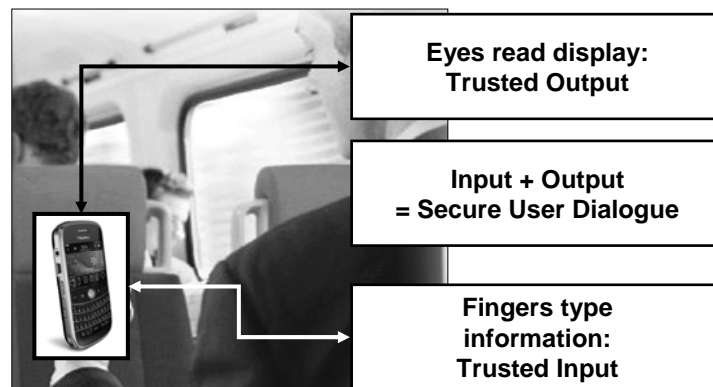


Figure 6: Trusted Input + Output = Secure Private User Dialog.

TEEs can provide a Secure User Interface / User Dialogue by assuring that any input will be transmitted in a secure way via the “Private World”. Also, TEEs can assure that the display is

¹¹ Note: The so called “memory footprint” refers to the amount of main memory that a program uses or references while running, including, for example, active memory regions like code, static data sections, all the stacks, plus the memory required to hold any additional data structures that the program ever needs while executing and will be loaded at least once during the entire run.

disconnected from the “Public World” while the individual is reading trusted information on the screen. However, the usage of the secure keypad functions does not yet prevent an attacker to write a “Public World” sniffer which could grab keypad data that is intended for the “Private World”. Therefore, a secure keypad Application needs to be combined with a secret that is stored in the “Private World”. Nevertheless, such solutions assist to provide the individual with input and output in an even more secure manner than all other presently existing SEs, and can add privacy- and identity-management-enhancement to these.

TEEs and Secure Storage of User Data

TEEs can provide convenient functions to encrypt/decrypt data without an additional need for implementing proprietary key handling. Applications in the “Private World” can leverage such specific keys which are derived from the SoC’s individual master key. These keys can then be used to protect secure objects that are maintained in the “Private World” and shall never be exposed to the “Public World”. These keys enable the protection of Application specific data and help to implement secure storage in the “Private World” for highly private and identity-management-relevant data sets. Hence, the TEE empowers the management of different, partial identities and privacy when communicating based on these.

Flexibility of the Front-end

In order to remain highly flexible and adaptive to changes in the environment of Mobile Services (see elaborations in the sections above), TEEs strive for independency from the Rich-Operating Systems (Rich-OS). This is particularly important as increasingly open Rich-OS systems diffuse in the Mobile Devices, e.g. Google’s Android.¹² Modern TEE approaches can be used on a wide range of TrustZone systems, especially if they are equipped with a clean and easy to understand integration interface to these. Here, reference drivers can be leveraged that help TEEs to integrate with specific Operating Systems, such as, for example, Google’s Android.

Based on the above described characteristics of TEEs, chapter 3 explains the interaction between Front- and Back-end:

¹² Note: For this reason, the PrimeLife demonstrator of the “eCV scenario“ was developed on the base of Google’s Nexus 1 Mobile Phone and Giesecke & Devrients Secure μ SD Card.

Chapter 3

Integration of Front- and Back-end for Secure creation of Dynamic Mobile Services

3.1 The collaboration of Front- and Back-end for secure and dynamic Mobile Services

To investigate how the Back-end of multi-domain web service provisioning can be identity- and privacy-enhanced, the integration of technologies needs to be looked into. For example, business software solution from market players such as SAP and Microsoft provide mechanisms that allow an administrator to control who should have access to a data and who not. When dealing with an environment where external online businesses (web / “Cloud”-based services, software-as-a service-offers) are consumed, access control techniques via the Mobile Devices in the Front-end may not be sufficient to specify how data should be handled. A user/administrator should be able to express this in the form of a rule or policy and should be reassured that future use does not in any way deviate from his/her original intention for the usage of the private data.

To facilitate such a system integration that respects the preferences/policies of users, a standard policy language and engine should be utilized. For this, an extension of existing programming practices and policy language such as the XACML standard applies. For example, the PPL (PrimeLife Policy Language, see [TS10]) tension that is currently being developed in the PrimeLife project may be leveraged.¹³

Herein, communication between the partners’ Applications in the Front- and Back-end adheres to the structure of the language and uses some features of the policy engine, such as obligations, i.e.

¹³ See www.primelife.eu

a function that obliges a data consumer to adhere to a policy such as deletion of data within a certain timeframe. The engine is also used to generate alerts to the user on policy conflicts with the latter having an option to permit access to data in spite of a difference in a data handling policy such as retention period. Interaction with the use can be enabled in a highly dynamic fashion, e.g. through interaction via the “Private World” of the Mobile Device in the Front-end.

A scenario may illustrate the above explained interaction between Front-and Back-end (also see [PU11]):

In the PrimeLife project, an “eCV-scenario” was designed. In this scenario, there are two sides representing the Front- and the Back-end:

- A user (i.e. a job applicant, Mobile Device, Front-end) creates an electronic CV (eCV) that contains up-to-date information on his personal details, work experience and academic qualifications. The personal information, such as the person’s gender, age or race, are entered by the user or provided by an official authority service. The other types of information may include university degrees, recommendation letters and previous or existing employer details, and they could be provided by the corresponding organization/data provider as a signed digital document or as a reference. For example, a university can certify qualifications attained and a recommendation is usually provided by an academic and/or employer. The user of the eCV portal has a Mobile Application on his Mobile Device. The Application is stored in the “Private World” and can only be accessed via entering the “Privacy PIN” of the individual person. The keys in the “Private World” de- and encrypt any communication the end user wants to have with the eCV portal and the policies managed in it.
- On the data consumer side (i.e. a headhunter, service side, Back-end), we have an eCV portal which executes a job broker service. This service collects offerings proposed by job providers or recruiter services and matches them with the data of the individual job applicant. The service contains clear rules on data usage, potentially under different conditions. For instance, the Back-end will not pass information to other parties, or will retain the data only for a pre-agreed period of time. If the interaction between a job applicant and a suitable job offer calls for exceptions regarding the pre-agreed data policy, the job portal can communicate with the job applicants Application on her / his private Mobile Device to ask for exceptions to the agreed data policy, based on specific circumstances. Hence, most of the data is stored in the Back-end and integration with the Front-end is possible via the eCV Application in the “Private World” (i.e. on the Secure μ SD card or the TEE).
- In this eCV scenario between the Front- and the Back-end, each contributory data provider (i.e. job applicant) could have a rule or sticky policy attached to the data that outlines how the data will have to be handled when used between the data producer (i.e. the job applicant), the data consumer (i.e. job broker leveraging the Back-end) or a third party. The parts of the eCV profile that contain the essential information about the privacy policy can only be altered by the job applicant himself. For this, the job applicant can log-in at the Back-end (e.g., an eCV portal) when she / he wants to change policy settings for the mid- to long-term.
- In cases where particularly interesting job opportunities arise, but the data needed to execute the matching between the job and the applicant exceeds the normal data policies stored in the Back-end, the eCV portal can make the applicant aware of the opportunity via the eCV Application in the “Private World” on the Mobile Device. For example, a policy will only allow using a recommendation letter for a certain period of time or it may be the case that the applicant does not allow a certain country, like the United Kingdom, to see his race. If this is a prerequisite to further process the matching of a particularly interesting opportunity, though, the applicant might want to make an exception and “overrule” his own policy for at least a limited amount of time. He can do so via the “Private World” by logging-in to the eCV Application on the Mobile Device, amending the policy preferences and thus lifting the “lock” on the Back-end process through his direct, secure, private and identity-related interaction with the Back-end.

- In essence, the electronic CV is composed of two parts, namely, the composition of data emanating from the different sources and the corresponding aggregated policies. The policy composition may contain conflicts, for example, the applicant may allow his personal contact details to be viewed by all services whereas the company he is working for states that it will not permit disclosure of where the employee works for security reasons. Conflicts can be resolved automatically or manually by the issuing of alerts to the user or by interference via the “Private World” of the Mobile Device of the job applicant.

On a general level, beyond the eCV scenario and its Application in the PrimeLife project, the interaction between Front- and Back-end has the following characteristics: A portal (i.e. a Back-end server) manages potential privacy- and identity management-related conflicts and sends requests that contain conflicts to the “Private World” of a user’s Mobile Device. The user then decides whether he / she wishes to use the requesting service or not, based on logging-in to her / his “Private World” (via the Privacy-PIN) in the Secure Element (e.g. Secure μ SD Card or TEE) on the Mobile Device.

3.2 The mobile Front-end as service delivering environment

In the above explained scenario of the eCV, the Mobile Device and the “Private World” in the SE on it allow for the following interactions which all together establish the Secure Dynamic Mobile Service:

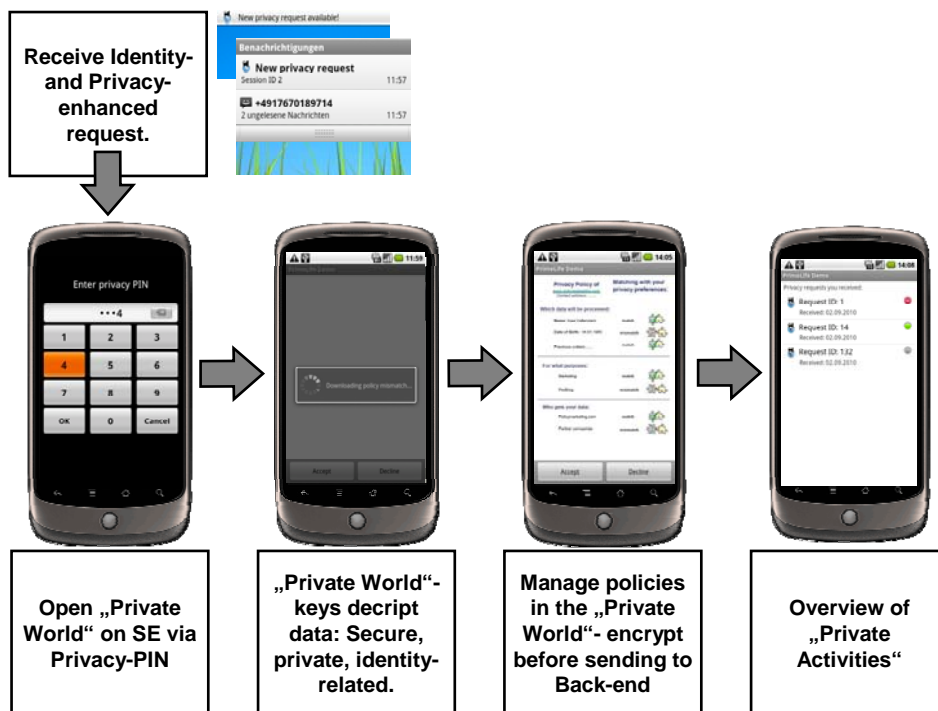


Figure 7: Interaction via the “Private World” of the Mobile Device: Example from the eCV Demonstrator.

- The “Private World” on the Mobile Device can “freeze” the Back-end if the end user / job applicant does not accept a policy mismatch.

- The “Private World” on the Mobile Device can deactivate the data set on the Back-end for a selected time period, e.g. if the job applicant does not want his private data to be visible for others online because he does not want to be approached by any job offering entity.
- The “Private World” on the Mobile Device can directly interact with the Back-end in a secure manner (e.g. via a Virtual Private Network or via encrypted communication) for data control in the future.
- The “Private World” on the Mobile Device holds the essential service keys for numerous privacy- and identity-management enhanced services and is therefore the privacy and identity-controlling device in the palm of each end consumer.
- The “Private World” on the Mobile Device provides a secure compartment / the TEE in which customizable services are empowered and in which additional data can be stored, e.g. additional certificates to enhance the eCV even more in selected cases.
- The “Private World” on the Mobile Device can “glue” other Secure Elements such as the SIM, the SD card and others together, if these are needed as sources of partial identities to provide more complete identity sets for particular services.

Hence, for example, an individual that does not agree with the way in which the Back-end requests to use private data (i.e. a policy mismatch in the Back-end because of which the server sends a privacy-enhanced request to the Front-end to ping the individual to decide how to solve the situation) can decline the request of the Back-end. The Mobile will then “freeze” any further data handling of the Back-end. Alternatively, the Front-end Mobile Device / the individual person could accept the request and “free” or even add further data for the subsequent processing via the Back-end. Also, one could overwrite policies that were initially stored in the Back-end, if, for example the individual user wants to change these policies because a policy mismatch has made him aware of a setting for which he has changed his preferences in the meantime. Also, the Mobile Device could trigger the complete deletion or temporary hiding of Back-end stored data if the individual decides that this data shall not be available to be seen by others via the Back-end. This interaction is secured and privacy is assured because the Front- and the Back-end need to hold the corresponding key. The interaction can also be encrypted through additional keys in order to assure that the channel between the Back-end and the Front-end is secured.

Further, the Front-end on the Mobile Device can be structured in such a manner that the TEE “glues” partial identities together. For example, one set of data (i.e. a partial identity) can be residing on the UICC whilst another partial identity is embedded on the SD card. Combined with the knowledge of the TEE about the existence of these two partial identities, the TEE can provide a combined identity to a Back-end service provider for highly security and privacy-relevant services.

3.3 The server Back-end as service processing counterpart

The Back-end for Secure Dynamic Mobile Services is embedded into a web-based platform (e.g. the Back-end of the eCV scenario). The server-side and the Mobile Device hold keys to empower the overall service. This enables a distribution of the private control of data between to two ends of the privacy and identity-management enabled solution, in a secure and highly dynamic manner. PrimeLife has specifically looked at this in a specific deliverable (see [PU11]). The Back-end for Secure Dynamic Mobile Services is embedded into a web-based platform (e.g. the Back-end of the eCV scenario). The server-side and the Mobile Device hold keys to empower the overall service. This enables a distribution of the private control of data between to two ends of the privacy and identity-management enabled solution, in a secure and highly dynamic manner.

On the Back-end, one junction point towards the Front-end on the Mobile Device is a job broker service that is called headhunter. The headhunter takes actions for seeking for appropriate

applicants on behalf of an employer. This is basically done by handing out the job offer including a service policy from the employer to a service that is called eCV portal.

The eCV portal let applicants register themselves and create profiles including their personal data, their CV, and their claims from former employers, universities, etc. Every user of the eCV portal is also kindly requested to define its privacy preferences, e.g., for what purpose its private data may be used and for what time period it may be stored by a third party like a headhunter or an employer.

The eCV portal, having a job offer and the related service policy from the employer, will match for suitable applications, taking not only the job offer's requirements and the applicant's skills into account but also considering the privacy policies from the user side and the service side, e.g. the employer. Only applications that are compliant in both terms are to be selected as suitable applications and are given to the headhunter as a response of its former job offer.

So far we left one participant of the eCV scenario's Back-end out: the domain expert. Domain experts are valuable experts on certain domains and offer assessing job applications regarding to their particular domain as a service to others. This service is to be used by the headhunter, if such a party is present like in our eCV scenario, as it would be too much overhead for an employer to interact with domain experts directly. In such a case where a domain expert is needed, the headhunter will also examine the policies of the available domain experts and will choose a compliant domain expert w.r.t. the application's sticky policy.

As the eCV portal as well as the headhunter are actively matching user policies against service policies, these are the junction points for the interaction with the "Private World" of the Mobile Device in case of a privacy policy mismatch. In such a case, these will trigger the "Private World" of the Mobile Device by sending it a text message which is consumed by the Front-end tool on that device only. Subsequently the Front-end tool will fetch necessary details about the mismatch using a secured web connection to the Back-end, and will show the user a notice about the mismatch occurrence. The user is now able to allow an exception of his privacy policy by "accepting" that mismatch or to freeze the application process by "rejecting" the mismatch.

3.4 Future legal requirements for secure and Dynamic Mobile Services

Within the envisioned scenarios of using TEEs¹⁴ in Front-end devices in coordination with larger Back-end mechanisms, a number of new legal challenges might arise. These challenges can only be estimated with a detailed look at the descriptions of the business models and the underlying technological descriptions, as a concrete legal evaluation is highly reliant on the specific use cases. In particular, detailed knowledge of these facts would be required for the evaluation whether the processing of certain personal data is necessary in the sense of Article 7 (a)-(f) of the Data Protection Directive (DPD)¹⁵ and the balancing test between the legitimate interests of a data controller and the data subjects rights and freedoms as required by Art. 7 (f) DPD.¹⁶

For weighing the interests of data subjects and data controllers, also other elements must be taken into account such as the type of data processed in particular when special categories of data are concerned, but also the context in which the personal data will be processed or might appear must be considered. On a more generic level it is possible to have a first view on the potential benefits

¹⁴ See chapter 3.1 above.

¹⁵ Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, available online: <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML>.

¹⁶ See Carey (2009), p. 71.

TEEs may have for privacy-enhancing technologies and how they could help to comply with existing legal requirements.

3.4.1 Considering security and privacy protection goals

Recent research¹⁷ in the area of privacy from a legal, sociological and technological perspective has shown that the traditional security protection goals¹⁸ of integrity, availability and confidentiality¹⁹ should be extended by specific privacy-related protection goals.

The security protection goals integrity, availability and confidentiality²⁰ share the commonality to address and ensure an appropriate level of data security primarily from the perspective of the organisation processing these data, e.g., service providers processing customer data, governmental authorities processing citizen data or employers processing employee data. While dealing with integrity, availability and confidentiality in a suitable way is a necessary prerequisite for effective privacy protection, these protection goals do not cover all aspects needed. To this end, additional complementary protection goals “intervenability”, “transparency” and “unlinkability” have been proposed that strengthen the data subject’s perspective on data processing.²¹ In Germany similar²² privacy protection goals have even been incorporated into the data protection law of the federal state of North Rhine-Westphalia, § 10 DSGVO NRW,²³ and intentions exist to incorporate such privacy protection goals in other German federal states as well.²⁴

Technologies such as the here presented Secure Elements, including the upcoming TEEs, may aid to implement these protection goals into current and future technology and thus help to provide enhanced privacy and user control as well as compliance to data protection law. The individual security and privacy protection goals are not independent from each other, and in each case it has to be analysed beforehand to which extent, on which layer and by which means the respective protection goal should be implemented into processes or IT systems. Possible tensions or even conflicts between protection goals have to be resolved: For instance, the security protection goal “integrity” aims at protecting data from manipulation. This could be understood in a way that prevents later deletion of records out of an existing database – all entries would be kept forever together with time stamps and information on their validity. However, this would not only contradict the data minimisation principle, but also it might be at conflict with the identified privacy protection goal “intervenability”. “Intervenability” means that it is possible to intervene in the data processing. This encompasses the possibility of redress or the data subject’s rights to rectify data or to revoke consent for processing with the consequence that data must be blocked for further processing or preferably deleted, cf. Art. 12 DPD.²⁵

¹⁷ See e.g. Rost/Pfutzmann (2009); Rost (2010); Konferenz der Datenschutzbeauftragten des Bundes und der Länder (2010)

¹⁸ Also called “control objectives”.

¹⁹ See chapter 2 above.

²⁰ See chapter 2 above.

²¹ Rost/Pfutzmann (2009); Rost (2010).

²² The privacy objectives in § 10 DSGVO NRW are partly different and comprise: Authenticity, requiring that information on the origin of personal Data must be available. Auditability requiring that it must be possible to assess who processed which personal data at which time and in which manner. Transparency requiring that procedures of data processing must be documented in a complete and up-to-date manner. (Translations by the author).

²³ Gesetz zum Schutz personenbezogener Daten (Datenschutzgesetz Nordrhein-Westfalen), available online: https://recht.nrw.de/lmi/owa/pl_text_anzeigen?v_id=3520071121100436275.

²⁴ Konferenz der Datenschutzbeauftragten des Bundes und der Länder, Ein modernes Datenschutzrecht für das 21. Jahrhundert, chapter 3. For Schleswig-Holstein available online: <https://www.datenschutzzentrum.de/sommerakademie/2010/sak10-guendermann-bedarf-novellierung-ldsg.pdf>.

²⁵ Note: See section 3.2 with the outlook on user scenarios and a potential “freeze” of data handled in the back-end via a Front-end device such as a mobile phone.

From the perspective of a data subject she / he should even be enabled to delete the data – after all they are her personal data – directly in the database of the organisation or respectively block the data when there are conflicting duties of the organisation to retain the data.²⁶ The organisation might want to restrict the direct access to the internal database to as few people as possible and thus may not be willing to provide data subjects this access possibility, but still support them in exercising the legally demanded data subject rights by offering comfortable ways to request changes or erasure of their data and inform them quickly on the outcome of their requests.

This illustrates that balances between conflicting protection goals have to be sought. In this context TEEs may be used to help implement “intervenability” while limiting the risk to the integrity of the data stored.²⁷ TEEs can be used to restrict access to the stored data strictly to the data subject entitled to these access rights, for example via a private Personal Identification Number (PIN) or code (e.g. biometrics such as voice or fingerprint recognition), or her designated delegates²⁸ including a secure and provable authentication of the person acting.²⁹

To proceed along this line of argumentation, further research is conducted at present in the legal and policy domain, including an ongoing discussion on the terminology of the privacy objectives.

3.4.2 Other legal requirements

Every processing of personal data must meet the legal requirements set forth in the DPD, the e-Privacy Directive³⁰ as well as national laws transposing these directives. Within PrimeLife a detailed analysis of the legal framework was done for data protection in service oriented architectures (SOAs).³¹ In general the requirements derived from the legal framework for SOAs apply for communications between Front- and Back-end in Mobile Services as well.

TEEs may reveal a special potential for enforcing compliance with policies on the Back-end respectively organisation part. Requirement 34 reads: “It should be possible to guarantee compliance with communicated policies.” This requirement may be enforced by Digital Rights Management (DRM) and also by Certification of the TEEs for specific Applications, e.g. for banking and payment services. Similar to the DRM technologies deployed by the music and film industry such technology might limit the processing of stored personal data to purposes and uses stipulated in the previously communicated privacy policy.³² While this may not make abuse impossible it may nevertheless hinder unintended and accidental disclosure of personal data. The administration of the rights and the control over data usage and exceptions may by these means be transferred to the user where the management tools could then reside well protected as part of the TEE.

²⁶ Note: See Chapter 4 with the outlook on user scenarios and a potential dynamic policy composition and also Front-end triggered deletion of back-end stored data via, e.g., a mobile phone.

²⁷ Note: TEEs as described in Chapter 2.

²⁸ As for requirements related to delegation of privacy related rights in general and data subjects rights in particular see: Hansen / Raguse / Storf / Zwingelberg (2010), p. 27 et seq.

²⁹ Note: From a business perspective, such interaction can be perceived as “Act of Will”. From a legal standpoint, however, there is a strong opinion amongst DPA and lawyers to strictly differentiate between authentication / identification on one hand, and “Acts of Will” on the other. According to that opinion, the latter may only be proven by digital signatures according to the signature directive / German SigG etc.

³⁰ Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications), available online: <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0058:EN:HTML>. See also Directive 2009/136/EC amending the e-Privacy Directive, available online: <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:337:0011:0036:En:PDF>. Directive 2009/136/EC must be transposed into national law by 25 May 2011 and may not be binding in all European jurisdictions until then.

³¹ Meissner / Schallaböck (2009).

³² See also Tóth (2004).

However, a set of requirements for a particular use case cannot be developed unless an exact use case description for the deployment of the SEs is available that explains in detail the types of data processed and data flows.

Chapter 4

Outlook and Conclusions: A Roadmap for SE-based Privacy in Secure and Dynamic Mobile Services

4.1 A Roadmap of solved and open issues for SE-based privacy

The above elaborated status quo of Secure and Dynamic Mobile Services with Secure Elements has drawn an up-to-date picture of the technical capabilities and the unprecedented integration of Front- and Back-end achieved in the PrimeLife project.

Based on the initial overview of security / trust, identity and privacy (including anonymity) capabilities of the various technologies, the following roadmap for further developments can be drawn:

- The requirement for highly dynamic adaptation to the ever changing market and technology environments can be tailored for through the different SE alternatives. The less dynamic the SEs need to be, the more one can expect that existing solutions will be leveraged. The more dynamic Mobile Services shall be designed, the more likely emerging SEs such as Secure μ SD cards and TEEs will be used.
- The Security / trust requirement is inherently solved by the usage of SE technologies.
- The requirement for numerous, partial identities is predominantly addressed by the emerging SEs, such as Secure μ SD cards and TEEs. It can be expected that Mobile Devices will leverage multiple SEs in the future (see figure 1), each hosting different partial identities.
- Privacy, as in the secure communication between predefined communication partners that relate to the respective partial identities (e.g., secure one-to-one communication between the end user and the eCV portal, between the end user and a banking / payment entity, or between the end user and a selected loyalty program provider such as an airline), can be assured through the emerging SE technologies.

- Anonymity, as in the unlinkability of the end user to its respective actions, remains an open issue. Here, technologies such as Identity Mixer and Direct Anonymous Attestation can prove very valuable if they would be combined with the above mentioned SEs. These could counteract the present unique identity of Mobile Devices (e.g. via their Subscriber Identity) and the linkability of this identity to the usage profile in Applications that still largely reside in the “Public” instead of the “Private” world.




	 Sticker	 µSD	 TEE	
Highly dynamic	No	Partially	Yes	Different SEs expected to co-exist, as existing and future applications will vary strongly.
Trust: A Trusted Secure Element / Environment	Yes	Yes	Yes	The SE concept inherently solves the trust issue.
Identity: A specific communication channel for the partial identity	No	Yes	Yes	Partial identities either pre installed or ad hoc, dynamic.
Privacy: Secure communication, only for the individual	Partially	Yes	Yes	New SEs expected to be preferred if privacy shall be stronger than in existing solutions.
Anonymity: Unlinkability of the interaction to the individual	Possibly	Possibly	Possibly	Next step beyond security, privacy and identity: Anonymity.

Figure 8: A Roadmap of Topics for secure, private and identity-related Mobile Services.

In essence, the here explained integration of Front- and Back-end technologies for Secure and Dynamic Mobile Service is a significant step into the direction of providing more (partial) identity-related and privacy-empowered Mobile Services. Nevertheless, open points such as anonymity remain to be solved.

4.2 Conclusion and outlook towards future research

The above presented findings draw attention to the following future directions of innovation in the area of Secure Dynamic Mobile Services, especially when distributed between Front- and Back-end domains.

For an overarching view, the eCV scenario (see [PU11]) of PrimeLife can assist to envision to collaboration between Front- and Back-end:

Conceptually, the following directions for future innovation in the field can be summarized (see figure 9):

Firstly, present Front-end technologies are adequate for static and increasingly flexible provisioning of privacy and identity management-enhanced Mobile Services. However, with the dynamics in the surrounding ecosystem of Mobile Service will increasingly call for even more highly adaptive solutions. Here, Trusted Execution Environments can provide highly dynamic and secure “Private Worlds”.

Secondly, Back-end systems need to be integrated with the upcoming capabilities in Front-end Mobile Devices. Here, interfaces, modularization, standardization and certification will be important stepping stones for consistent overall solutions. Taking the concept further that the Mobile Device could “freeze” an ongoing process in the Back-end, future research could be directed towards analyzing whether the Front-end Mobile Device could also be used as an essential “token” through which all online data and identity could be managed, encrypted if needed and deleted if wanted.

Thirdly, shared capabilities between Front- and Back-end with one overarching vision are necessary towards Secure Dynamic Mobile Services and a strict modularization / interface creation along the Value Chain to empower scalability.

Fourthly, further clarification of the legal implications at the interfaces between the individual user, the Mobile Device, the Back-end and any Third Parties wanting to access and leverage private data are needed.

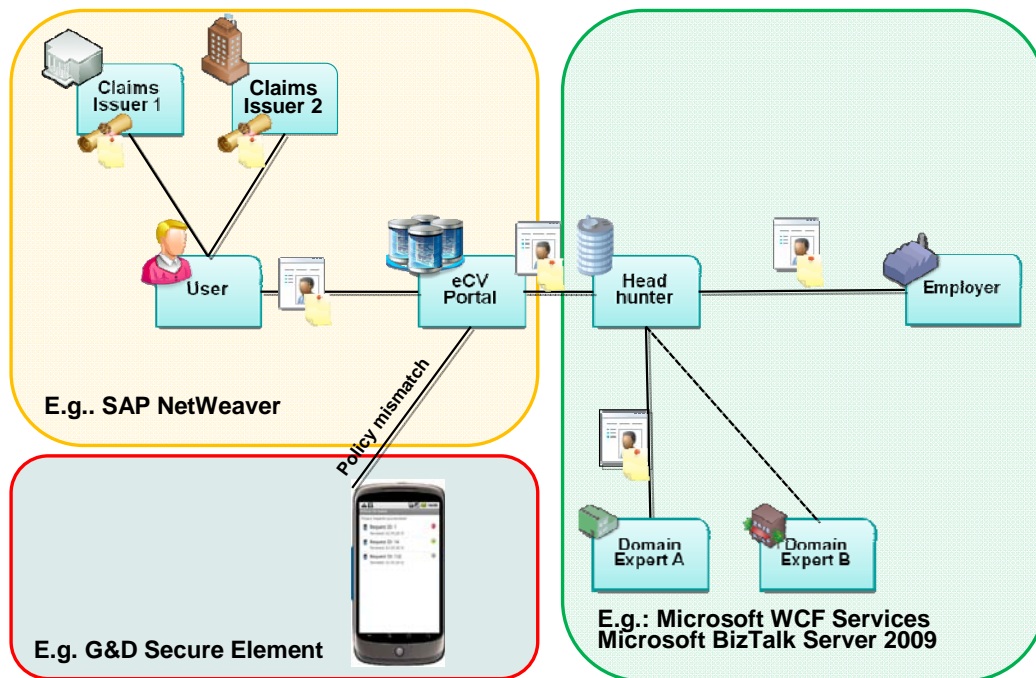


Figure 9: The full Integration of Front-end and Back-end for Secure Dynamic Mobile Services

Fifthly, the anonymity perspective of privacy is not yet solved by the latest technologies and the integration of Front- and Back-end technologies. Here, additional work needs to be done.

The perspectives laid out in this paper by referencing to the work of Giesecke & Devrient, Microsoft and SAP in the PrimeLife project can serve as a first indication for future research. The above mentioned points remain as future work.

References

- [BM09] Bergfeld, M.-M. H. (2009): Global Innovation Leadership: The strategic development of worldwide innovation competence. BOD, Norderstedt, 2009.
- [BM10] Bachfeld, D. and Mulliner, C. (2010): Risiko Smartphone: Spionageangriffe und Abzocke auf Android und iPhone. In: c't 2010, Heft 20, pp. 80-85.
- [BHS08] Bergfeld, M.-M. H., Hinz, W. and Spitz, S. (2008): Infrastructure for Trusted Content, Deliverable 6.2.1 of the PrimeLife Consortium, http://www.primelife.eu/images/stories/deliverables/d6.2.1-infrastructure_for_trusted_content-public.pdf, accessed: October 15th 2010.
- [CK05] Kim Cameron's Identity Blog (2005): URL: <http://www.identityblog.com/stories/2004/12/09/thelaws.html>, accessed: December 6th 2010.
- [ES06] Electronic Storage (2006): Removable memory cards: not just a flash in the pan, URL: http://www.oto-online.com/pdf/oto_download/2006/06/OTO_June_P5254_MemoryCards.pdf, accessed: December 6th 2009.
- [GA08] GSM Association (2008): GSMA calls for Pay-Buy-Mobile handsets, URL: <http://gsmworld.com/newsroom/press-releases/2008/2090.htm#nav-6>, accessed: January 12th 2010.
- [GP10] GlobalPlatform Device Technology TEE Client API Specification, Version 1.0, Public Release, July 2010, Document Reference: GPD_SPE_007.
- [HRSZ10] Hansen, M., Raguse, M., Storf, K. and Zwingelberg, H. (2010): Delegation for Privacy Management from Womb to Tomb – A European Perspective, in: Proceedings for IFIP / PrimeLife Summer School 2009 held in Nice, France, September 7-11 2009, pp. 18-33.
- [KD10] Konferenz der Datenschutzbeauftragten des Bundes und der Länder (2010): Ein modernes Datenschutzrecht für das 21. Jahrhundert - Eckpunkte, Stuttgart, <http://www.baden-wuerttemberg.datenschutz.de/service/gem-materialien/modernisierung.pdf>, accessed November 15th 2010.
- [MF09a] Mobey Forum (2009a): Global Overview of commercial implementations and pilots of NFC payments during 2009. Article for globalsmart.com – Smart Card Technology International, <http://mobeyforum.org/>, accessed: February 5th 2010.
- [MF09b] Mobey Forum (2009b): Why aren't banks rushing for NFC payments? Article for globalsmart.com – Smart Card Technology International, URL: <http://mobeyforum.org/>, accessed: February 5th 2010.
- [MF10] Mobey Forum (2010): Alternatives for Banks to offer Secure Mobile Payments. Whitepaper of the MobeyForum, <http://mobeyforum.org/Press->

Documents/Press-Releases/Alternatives-for-Banks-to-offer-Secure-Mobile-Payments, accessed: December 5th 2010.

- [IS08] iSuppli (2008): Mobile Handset Shipments with micro SD / Micro SDHC, in: Data Flash Market tracker Q3/2008, supplied via SD Association.
- [PU11] Pinsdorf, U. (2011): Infrastructure for Privacy for Life, http://www.primelife.eu/images/stories/deliverables/d6.3.2-infrastructure_for_privacy_for_life-public.pdf, accessed: April 15th 2011.
- [RC10] Rütten, C. (2010): Ausgespäht: Sicherheit von Apps für Android und iPhone. In: c't, Heft 20, pp. 86-91.
- [RJ10] Rosen, J. (2010): The Web Means the End of Forgetting, in: New York Times (2010), <http://www.nytimes.com/2010/07/25/magazine/25privacy-t2.html?pagewanted=1&r=4&hp>, accessed October 1st 2010.
- [RP09] Rost, M. and Pfitzmann, A. (2009): Datenschutz-Schutzziele – revisited, in: Datenschutz und Datensicherheit (DuD) 2009, pp. 353-358, http://www.maroki.de/pub/privacy/DuD0906_Schutzziele.pdf, accessed: October 20th 2010.
- [RM10] Rost, M. (2010): Datenschutzziele neu definiert, in Datenschutz-Berater (DSB) July/August 2010, pp. 22-25, http://www.datenschutz-berater.de/CONTENT/default.aspx?_p=1004211&_t=pdf&_s=361156, accessed: October 22nd 2010.
- [SD10] SD Association (2010): SD Association celebrates 10 years of innovation at CES, http://www.sdcard.org/press/SD_Celebrates_10_Years_of_Innovation_at_CES_2010.pdf, accessed: January 10th 2010.
- [SSP08] Swoboda, J., Spitz, S. and Pramateftakis, M. (2008): Kryptographie und IT-Sicherheit: Grundlagen und Anwendungen - eine Einführung, Vieweg & Teubner, Wiesbaden.
- [TG04] Tóth, G. (2004): DRM and privacy - friends or foes? - An introduction to Privacy Rights Management (PRM), online: http://www.indicare.org/tiki-read_article.php?articleId=45, accessed: October 22nd 2010.
- [TS10] Trabelsi, S. (2010): Second Release of the Policy Engine, http://www.primelife.eu/images/stories/deliverables/d5.3.2-second_release_of_the_policy_engine-public.pdf, accessed: March 5th 2011.
- [VL10] Van den Berg, B. and Leenes, R. (2010): Audience segregation in social network sites, in: Proceedings for SocialCom2010/PASSAT2010 (Second IEEE International Conference on Social Computing/Second IEEE International Conference on Privacy, Security, Risk and Trust), 20-22 August 2010 in Minneapolis (Minnesota, USA). IEEE, pp. 1111-1117.
- [WV10] Weber, V. (2010): Zwickmühle: Der Streit um Blackberry-Sicherheit. In: c't, Heft 20, pp. 144-161.

Appendix **A**

TEE Client API Description

Note: The below attached text is a direct quote from the text published by GlobalPlatform in [GP10]. G&D's activities in contributing to the below mentioned standardization results were coordinated with G&D's activities in PrimeLife. Parts of G&D's standardization efforts in the field were financed by PrimeLife.

[Quote]

A.1 Overview of the TEE Client API as standardized in Global Platform

This specification defines a communications API for connecting Client Applications running in a rich operating environment with security related Trusted Applications running inside a Trusted Execution Environment (TEE). For the purposes of this document a TEE is expected to be a trusted environment within the main device system-on-a-chip, which complements traditional security environments such as a UICC SIM card, although this is not a requirement of the API. A TEE provides an execution environment with security capabilities, which are either available to Trusted Applications running inside the TEE or exposed externally to Client Applications. A TEE may, for example, host a GPD/STIP runtime [5], but may also be based on other technologies such as a small operating system executing native code Applications.

See the Open Mobile Trusted Platform (OMTP) Advanced Trusted Environment TR1 specification [4] for a requirements analysis of Trusted Execution Environments in mobile devices.

A.2 Scope of the standardization of the TEE Client API in Global Platform

Instead of trying to standardize a single monolithic API which covers a significant proportion of the interactions between a Client Application and the TEE-hosted functionality, the approach of the Global Platform standardization effort is modular. The TEE Client API covered by this specification concentrates on the interface to enable efficient communications between a Client Application and a Trusted Application running inside the TEE. Higher level standards and

protocol layers may be built on top of the foundation provided by the TEE Client API – for example, to cover common tasks such as secure storage, cryptography, and run-time installation of new Trusted Applications – but these interfaces are outside of the scope of this specification.

A.3 TEE Client API Architecture

The relationship between the major system components described in this specification are outlined in the block architecture below.

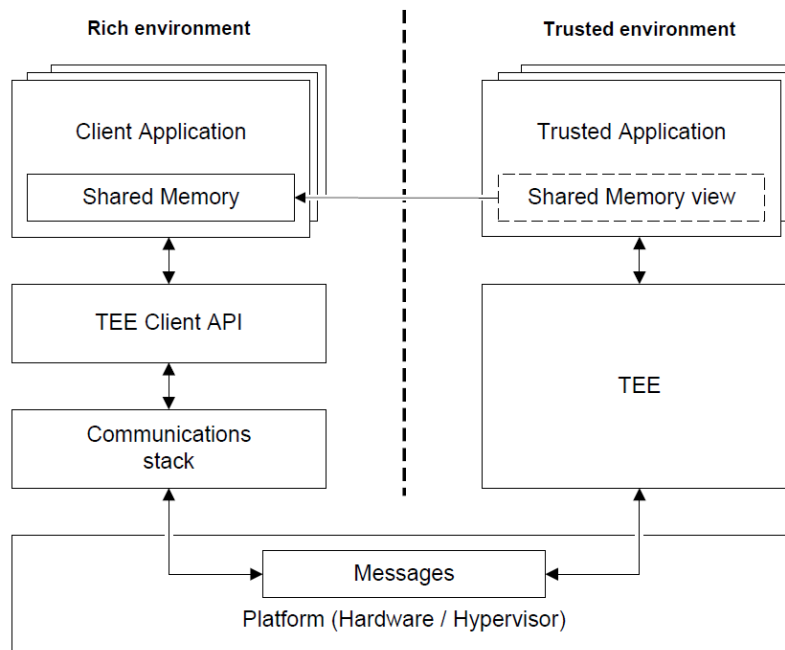


Figure 10: TEE Client API System Architecture as standardized in Global Platform

Some implementation-defined support is required to provide separation between the rich environment and the TEE. The mechanisms used to achieve this, and the level of security these mechanisms provide, are outside of the scope of this specification.

Within the trusted environment this specification identifies two distinct classes of component: the hosting code of the TEE itself, and the Trusted Applications which run on top of it. There is no definition of the expected implementation of these blocks in this specification; they are only used as logical concepts inside this document.

Within the rich environment this specification identifies three distinct classes of component:

- The Client Applications which make use of the TEE Client API.
- The TEE Client API library implementation.
- The communications stack which is shared amongst all Client Applications, and whose role is to handle the communications between the rich environment and the trusted environment.

As before, there is no mandated architecture for these components and they are only used as logical constructions within this specification document. Note that the TEE Client API may be exposed to either, or both, the privileged or user layers of the rich environment. If exposed in the privileged layer, then drivers or any other privileged components may be considered to take the place of Client Applications.

A.4 TEE Client API Principles and Concepts

This section explains the underlying principles and concepts of the TEE Client API in detail, explaining how each class of features should be used.

A.4.1 TEE Client API Design Principles

The key design principles of the TEE Client API are:

- C language
 - C is the common denominator for practically all of the Application frameworks and operating systems hosting Client Applications.
 - It is accepted that alternative language bindings – such as a Java API – may be needed in the future, but these are outside of the scope of this specification.
- Blocking functions:
 - Most Client Application developers are familiar with synchronous functions which block waiting for the underlying task to complete before returning to the calling code. An asynchronous interface is hard to design, hard to port in rich OS environments, and is generally difficult to use for developers familiar with synchronous APIs.
 - In addition it is assumed that multi-threading support is available on all target platforms; this is required for Implementations to support cancellation of blocking API functions.
- Source-level portability:
 - To enable compile-time and design-time optimization, this standard places no requirement on binary compatibility. Client Application developers will need to recompile their code against an appropriate implementation-defined version of the TEE Client API headers in order to function correctly on that Implementation.
- Client-side memory allocations:
 - Where possible the design of the TEE Client API has placed the responsibility for memory allocation on the calling Client Application code. This gives the Client developer choice of memory allocation locations, enabling simple optimizations such as stack-based allocation or enhanced flexibility using placements in static global memory or thread-local storage.
 - This design choice is evident in the API by the use of pointers to structures rather than opaque handles to represent any manipulated objects. To enable compile-time and design-time optimization, this standard places no requirement on binary compatibility. Client Application developers will need to recompile their code against an appropriate implementation-defined version of the TEE Client API headers in order to function correctly on that Implementation.
- Aim for zero-copy data transfer:
 - The features of the TEE Client API are chosen to maximize the possibility of zero-copy data transfer between the Client Application and the Trusted Application, provided that the host operating system and hardware implementation can support it. This minimizes communications overhead and improves software efficiency, especially on cached processors where data copies are an expensive operation because of the cache pollution they cause.
 - However, short messages can also be passed by copy, which avoids the overhead of sharing memory.

- Support memory sharing by pointers:
 - The TEE Client API will be used to implement higher-level APIs, such as cryptography or secure storage, where the caller will often provide memory buffers for input or output data using simple C pointers. The TEE Client API must allow efficient sharing of this type of memory, and as such does not rely on the Client Application being able to use bulk memory buffers allocated by the TEE Client API.
- Specify only communication mechanisms:
 - This API focuses on defining the underlying communications channel. It does not define the format of the messages which pass over the channel, or the protocols used by specific Trusted Applications. These must be defined in other specifications.

A.4.2 TEE Client API Fundamental Concepts

This section outlines the behaviour of the TEE Client API, and introduces key concepts and terminology:

A.4.2.1 TEE Contexts

A TEE Context is an abstraction of the logical connection which exists between a Client Application and a TEE. A TEE Context must be initialized before a Session can be created between the Client Application and a Trusted Application running within the TEE which that TEE Context represents. The TEE Context should be finalized when the connection with the TEE is no longer required, allowing resources to be released.

It is possible for a Client Application to initialize multiple TEE Contexts concurrently, either with the same underlying TEE, or with multiple TEEs if they are available. The number of concurrent contexts which may exist is implementation-defined, and may additionally depend on run-time resource constraints.

A.4.2.2 Sessions

A Session is an abstraction of the logical connection which exists between a Client Application and a specific Trusted Application. A Session is opened by the Client Application within the scope of a particular TEE Context. The number of concurrent Sessions which may exist is implementation-defined, depending on the design of the TEE and the Trusted Applications in use, and may additionally depend on run-time resource constraints.

When creating a new Session the Client Application must identify the Trusted Applications which it wishes to connect to using the Universally Unique Identifier (UUID) of the Trusted Application. The open session operation allows an initial data exchange to be made with the Trusted Application, if this is required in the protocol between the Client Application and the Trusted Application.

Connection Methods: Login

Some Trusted Applications may require the Implementation to identify or authenticate the Client Application or the user executing it. For example, a Trusted Application may restrict access to the data or functionality it provides based on the identity of the user running the Client Application in the rich operating environment. When opening a Session the Client Application can nominate which connection method it wants to use and hence which login credentials are presented to the TEE or Trusted Application. It is likely that the connection method will form part of the protocol

exposed by the Trusted Application in use; attempting to open a Session with an incorrect connection method may result in a failed attempt.

A.4.2.3 Commands

A Command is the unit of communication between a Client Application and a Trusted Application within a Session. When starting a new Command the Client Application identifies the function in the Trusted Application which it wishes to execute by passing a numeric identifier, and may also provide an operation payload in accordance with the protocol the Trusted Application exposes for that function. The Command invocation blocks the Client Application thread, waiting for an answer from the Trusted Application. A Client Application may use multiple threads to have multiple Commands which are outstanding concurrently. The number of concurrent Commands which may exist is implementation-defined, depending on the design of the TEE and the Trusted Applications in use, and may additionally depend on run-time resource constraints.

Operation Payload

An operation to open a Session or to invoke a generic Command can carry an optional payload, the definition of which is passed inside a set of Operation Parameters (see section 3.2.5 of the Global Platform document [GP10]) stored in the operation structure. In this version of the specification up to 4 Parameters can be specified for each operation.

Each Parameter is either a Memory Reference or a Value Parameter and is associated with a direction: it can be input, output, or both input and output. For Memory Reference Parameters, the specified direction of data flow determines when the underlying memory buffers need to be synchronized with the Trusted Application.

Memory Reference Parameters are used to exchange data through shared memory buffers. Value Parameters carry a small amount of data in the form of two 32-bit integers without the burden of sharing or synchronizing memory.

The format of the data structures held in the Memory References or Value Parameters is defined by the protocol of the Trusted Application function in use, and hence outside of the scope of this specification.

Temporary Shared Memory Registration

Memory References refer either to a Registered Memory Reference or a Temporary Memory Reference:

- a Registered Memory Reference is a region within a block of Shared Memory (see section 3.2.4 of the Global Platform document [GP10]) that was created before the operation
- a Temporary Memory Reference directly specifies a buffer of memory owned by the Client Application, which is temporarily registered by the TEE Client API for the duration of the operation being performed

A Temporary Memory Reference may be null, which can be used to denote a special case for the parameter. Output Memory References that are null are typically used to request the required output size.

Return Codes and Return Origins

The answer to an open Session and a invoke Command operation always contains a Return code, which is a 32-bit numeric value indicating success or the reason for failure, and an Return origin, which is a 32-bit numeric value indicating the source of the return code in the Implementation. The standard error codes and return origins are described in sections 4.4.2 and 4.4.3 of the Global Platform document [GP10].

When the return origin is TEEC_ORIGIN_TRUSTED_APP then the return code is defined by the Trusted Application's protocol. Note that, critically, this means that a Client Application cannot just test against TEEC_SUCCESS, as the Trusted Application may use another code to indicate success. To enable simpler error handling code in the Client Application it is recommended that the Trusted Application developers choose „0” as their literal value of their success return code constant.

Events and Callbacks

This specification does not define a primitive way for a Trusted Application to spontaneously signal an event to the Client Application or perform callbacks to the Client code. However, these types of usage patterns can be constructed using Commands. For example, event signals can be implemented by having the Client Application send a Command which blocks inside the Trusted Application until the event occurs inside the TEE. When the event occurs the Trusted Application passes control back to the Client Application; the TEEC_InvokeCommand will return and the Client Application can handle the event which was signaled.

A.4.2.4 Shared Memory

A Shared Memory block is a region of memory allocated in the context of the Client Application memory space that can be used to transfer data between that Client Application and a Trusted Application.

A Shared Memory block can either be existing Client Application memory which is subsequently registered with the TEE Client API, or memory which is allocated on behalf of the Client Application using the TEE Client API. A Shared Memory block can be registered or allocated once and then used in multiple Commands, and even in multiple Sessions, provided they exist within the scope of the TEE Context in which the Shared Memory was created. This pre-registration is typically more efficient than registering a block of memory using temporary registration if that memory buffer is used in more than one Command invocation.

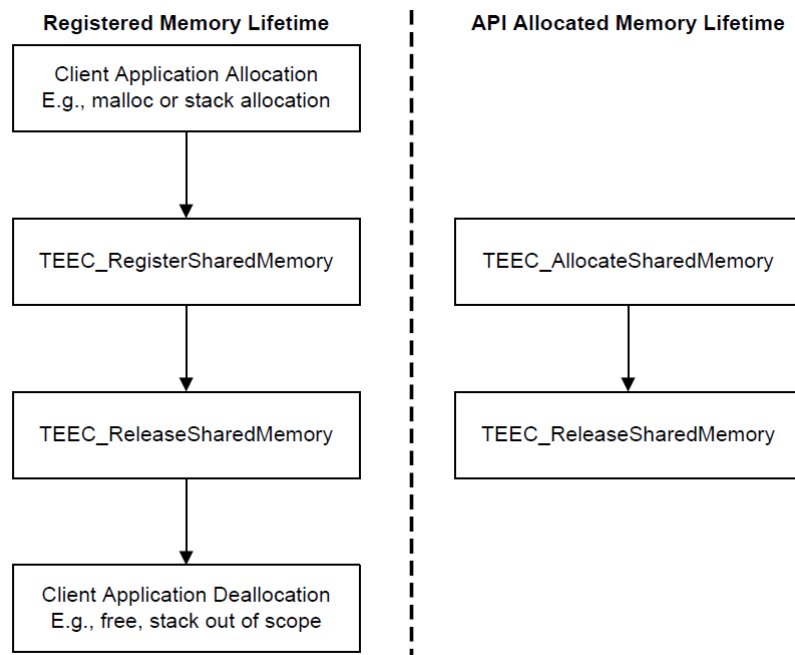


Figure 11: TEE Client API Shared Memory Buffer Lifetime

Zero-copy Data Transfer

When possible the implementation of the communications channel beneath the TEE Client API should try to directly map Shared Memory in to the Trusted Application memory space, enabling true zero-copy data transfer. However this is not always possible; for example, the TEE may not have access to the same physical memory system as the platform running the Client Application, or may only be able to achieve zero-copy for some types of memory. As a result this specification defines synchronization points where the TEE Client API Implementation is allowed to synchronize the data in a Shared Memory block with the TEE to ensure data consistency. The Client Application and Trusted Application must assume that the data is only synchronized when within the scope of these synchronization points. Otherwise, data corruption may result. This process is described in more detail in section A4.2.5.

Client Application developers should note that letting the TEE Client API allocate the memory buffers using the function `TEEC_AllocateSharedMemory` maximizes the chances that it can be successfully shared using a zero-copy exchange. If Client Application developers have the option to use this type of allocated memory in their code, without needing an explicit copy from another buffer, then they should aim to do so. However, it is not always possible to allocate memory without a copy in the Client Application, and in these cases registration of the buffer using `TEEC_RegisterSharedMemory` is the preferred option as there is still a possibility that it could be zero copy. Note that for small amount of data, it is recommended to use a Value Parameter instead of a Memory Reference to avoid the overhead of memory management.

Overlapping Blocks

The API allows Shared Memory registrations and allocations to overlap. A single region of Client Application memory may be registered multiple times, or a block may be allocated and then subsequently registered. The Client is responsible for ensuring that the overlapping regions are consistent and meet any timing requirements when used by multiple actors; specifying an input buffer to one Trusted Application which is concurrently used as an output for another can produce undefined results, for example.

The rules which the Client must conform to when overlapping memory ranges are used concurrently are described in the synchronization sub-section of section A4.2.5.

A.4.2.5 Memory References

A Memory Reference is a range of bytes which is actually shared for a particular operation. A Memory Reference is described by either a `TEEC_MemoryReference` structure (see section 4.3.7) or a `TEEC_TempMemoryReference`. It can specify:

- A whole Shared Memory block.
- A range of bytes within a Shared Memory block.
- Or a pointer to a buffer of memory owned by the Client Application, in which case this buffer is temporarily registered for the duration of the operation. This type of Memory Reference uses the structure `TEEC_TempMemoryReference`.

A Memory Reference also specifies the direction in which data flows for that particular command. Memory References may be marked as input (buffer is transferring data from the Client Application to the Trusted Application), output (buffer is transferring data from the Trusted Application to the Client Application), or both input and output.

When a Memory Reference points to a Shared Memory block, the data flow direction must be consistent with the set of flags defined by the parent Shared Memory block; for example, trying to make an input Memory Reference with a parent Shared Memory block which has only the `TEEC_MEM_OUTPUT` flag is invalid.

Synchronization

As the underlying communications system may not support direct mapping of Client memory into the Trusted Application, it may be necessary to copy a portion of memory from the Client memory space into the Trusted Application memory space. Memory References provide a token which indicates what memory range needs to be synchronized, and their use within an operation indicates the duration of the synchronization scope. The temporal states in this synchronization process are indicated in the Figure below.

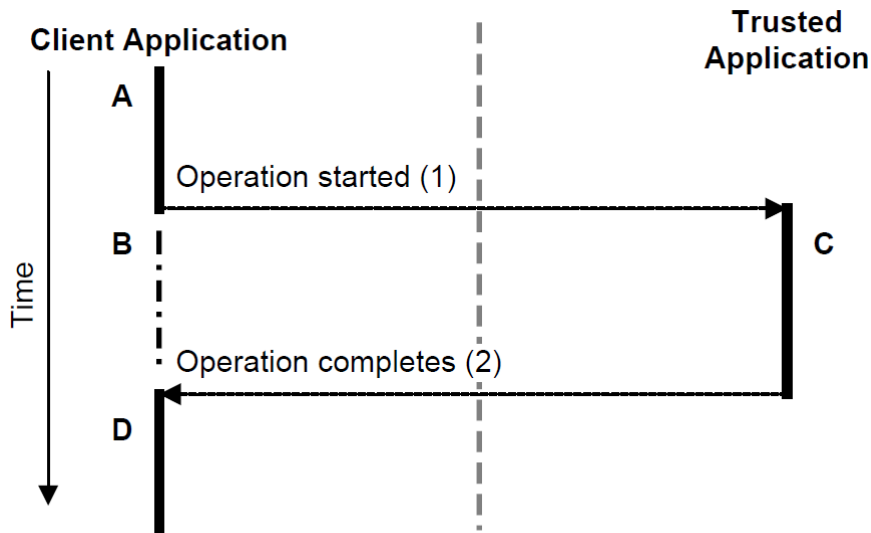


Figure 12: TEE Client API Memory Reference timing diagram

In this figure there are three temporal states for the Client Application (A, B, D) and one for the Trusted Application (C), as well as two synchronization operations (1, 2).

When performing synchronization operation 1 – transitioning from state A to states B and C (which exist in parallel in the two environments) – the Implementation needs to ensure that input buffers are synchronized from the Client Application’s view of memory to the Trusted Application’s view of it. When performing synchronization operation 2 – transitioning from states B and C (which exist in parallel in the two environments) to state D – the Implementation needs to ensure that output buffers are synchronized from the Trusted Application’s view of memory to the Client Application’s view of memory.

The range of bytes referenced in a Memory Reference is considered live, for synchronization purposes, the moment that the containing operation structure is passed in to either `TEEC_OpenSession` or `TEEC_InvokeCommand`; this live period corresponds to the temporal states B and C in the figure. A Memory Reference is considered to be no longer live when the called API function returns. While a Memory Reference is live the Client Application and the Trusted Application must obey the following constraints:

- For ranges within a Memory Reference marked as input only, the Client Application may read from the memory range, but must not write within it (states B and C). The Trusted Application may read from the memory range during state C.

- For ranges within a Memory Reference marked as input or input and output, the Client Application must neither read nor write within the memory range (state B). The Trusted Application may read and write to the memory range (state C).

If these synchronization rules are ignored by the Client Application or the Trusted Application then data corruption may occur.

Overlapping Ranges

The API allows Memory References to overlap, either within a single operation or across multiple operations. The Client is responsible for ensuring that the overlapping regions are consistent and meet any timing requirements when used by multiple actors; specifying an input buffer to one Trusted Application which is concurrently used as an output for another will produce undefined results, for example.

It may be necessary for constraints on overlapping ranges to be defined as part of the Trusted

Application's protocol. A Trusted Application which accepts an input buffer and an output buffer, but which writes to the output buffer before using the input, cannot use the same memory for both activities as writing the output will destroy the input.

Memory Reference Types

The specification supports the following types of Memory Reference which may be encoded in an operation payload.

- `TEEC_MEMREF_TEMP_INPUT`, `TEEC_MEMREF_TEMP_OUTPUT`, or `TEEC_MEMREF_TEMP_INOUT`: A temporary Memory Reference indicates that the Parameter points to a buffer of memory to be shared rather than to a Shared Memory control structure. This Client Application buffer will be temporarily shared for the duration of the operation being performed. If the buffer pointer is NULL then no memory buffer is actually referenced. Some Trusted Applications may associate a specific meaning with a null Memory Reference, so for full details the Client Application developer must refer to the protocol specification for the Trusted Application they are targeting. A null Memory Reference can also be used to fetch the required size of an output buffer.
- `TEEC_MEMREF_WHOLE`: A whole Memory Reference enables a light-weight mechanism of sharing an entire parent Shared Memory block without the need to duplicate the content of the Shared Memory structure control fields inside the Memory Reference. When this memory type is used the entire Shared Memory region is shared with the direction flags the parent Shared Memory specifies.
- `TEEC_MEMREF_PARTIAL_INPUT`, `TEEC_MEMREF_PARTIAL_OUTPUT`, or `TEEC_MEMREF_PARTIAL_INOUT`: A partial Memory Reference refers to a sub-region of a parent Shared Memory block, allowing any region of bytes within that block to be shared with the Trusted Application.

Note that an Operation Parameter can also be a Value Parameter, carrying two 32-bit integers.

Variable Length Return Buffers

In many cases the Trusted Application will want to write a variable length of data in to the Shared Memory buffer. For buffers which are configured as an output buffer, the size of the Memory Reference when starting an Operation on the TEE is the maximum size of the output data that the Trusted Application may write into the referenced region. When the Trusted Application responds it may reduce the size of the referenced memory region to reflect the actual number of bytes it wrote into the output buffer. In this case the Implementation must update the size field of the Memory Reference in the Client Application operation structure to indicate the number of bytes which were used by the Trusted Application.

In these cases the Implementation only needs to synchronize the number of bytes which the Trusted Application has modified when passing control back to the Client Application; other data

within the scope of the originally referenced memory range should be unchanged, although this may depend on Trusted Application behaving correctly.

Note that output data can only be written in the lowest address in an output Memory Reference; it is not possible to synchronize a high region in the buffer without synchronizing the lower parts of the buffer.

In any scenario using variable length outputs there is the possibility that the output buffer provided by the Client Application is not large enough to contain the entire output. In these scenarios the Trusted Application is allowed to return the required output size to the Client Application. The size field of the Memory Reference in the operation structure is then updated to reflect the required size, but the Implementation does not synchronize any data with the Client Application, as this is viewed as an error condition. It is recommended that a Trusted Application use the defined “short buffer” error code `TEEC_ERROR_SHORT_BUFFER` to signal this type of response to the Client Application.

This type of “short buffer” response is allowed for null Memory Reference, enabling a design where a first invocation uses a null Memory Reference to fetch the required size of output buffer, and then uses a second invocation with another non-null Memory Reference containing an output buffer of the necessary size.

A.4.3 TEE Client API Usage Concepts

The section outlines some of the usage patterns which the design of the TEE Client API makes use of:

A.4.3.1 Operation Instantiation

To enable reliable multi-threaded implementations of cancellation this specification defines the concept of Instantiation – a mechanism which can be used to put `TEEC_Operation` structures in to a known state. If an Operation may be cancelled by the Client Application then the Client Application must set the `started` field of the structure to 0 before calling either the `TEEC_OpenSession` or `TEEC_InvokeCommand` function. If a Client Application is single threaded, or is multi-threaded but will never cancel the operation by design, then there is no need for the `started` field to be initialized.

Atomicity of Field Access

To enable multi-threaded TEE Client API implementations to effectively use the `started` field across multiple-threads without the need for OS level locking, the underlying processor architecture must allow atomic operations – such as “test and set”, “swap”, or “exclusive load and store” – to operate on the `started` field. For this reason the `started` field has been chosen to be 32-bits, as this is a commonly supported data size for atomic operations on the processor architectures of interest.

This atomicity requirement typically means that the `started` fields must be naturally aligned (aligned on a 4-byte boundary); otherwise the atomic instructions in the processors will not function correctly. This requirement is automatically met by compliant C code and toolchains, but many toolchains allow extensions to the C language which allow packed and / or unaligned structures. Client Applications must not use these extensions; TEE Client API implementations are allowed to assume the `started` field can be read or written atomically.

A.4.3.2 Multi-threading

The TEE Client API is designed to support use from multiple threads concurrently, using a combination of internal thread safety within the implementation of the API, and explicit locks and serialization in the Client Application code. Client Application developers can assume that all of the API functions can be used concurrently unless an exception is documented in this specification. The main exceptions are indicated below.

Note that the API can be used from multiple processes, but it may not be possible to share contexts and sessions between multiple processes due to rich OS memory separation mechanisms.

Behavior which is not Thread-safe

TEE Contexts, Sessions, and Shared Memory structures all have an explicit lifecycles defined by pairs of bounding “start” and “stop” functions:

- TEEC_InitializeContext / TEEC_FinalizeContext
- TEEC_OpenSession / TEEC_CloseSession
- TEEC_RegisterSharedMemory / TEEC_ReleaseSharedMemory
- TEEC_AllocateSharedMemory / TEEC_ReleaseSharedMemory

These functions are not internally thread-safe with respect to the object being initialized or finalized. It is not valid to call TEEC_OpenSession concurrently using the same TEEC_Session structure, for example. However, it is valid for the Client Application to concurrently use these functions to initialize or finalize different objects; in the above example two threads could initialize different TEEC_Session structures.

In cases where global shared structures need to be initialized the Client Application must ensure that the initialization of each structure only occurs once using appropriate platform-specific locking schemes to ensure that this requirement is met.

Once the structures described above have been initialized it becomes possible to use them concurrently in other API functions, provided that the TEE and Trusted Application in use support such concurrent use. A Client Application can concurrently register two different Shared Memory blocks using the same TEE Context, or invoke two Commands within the same Session for example.

A.4.3.3 Resource Cleanup

The specification of the “stop” functions described in section 3.3.2 is stateful and requires clean Client Application resource unwinding:

- when releasing Shared Memory, the Client code must ensure that it is not referenced in a pending operation
- when closing a session, there must be no pending operations within it
- when finalizing a TEE Context there must be no open sessions within its scope TEEC_RegisterSharedMemory / TEEC_ReleaseSharedMemory

The Client Applications must ensure these conditions are true, using platform-specific locking mechanisms to synchronize threads if needed. Failing to meet these obligations is a programmer error, and will result in undefined behavior.

A.4.4 Security

This section outlines the security policies of the TEE Client API, and highlights some of the design requirements which are placed on an Implementation:

A.4.4.1 Security of the TEE and Trusted Applications

The implementation of the TEE and any Trusted Applications must treat any input from the rich environment as potentially malicious; Client Applications are running outside of the TEE security boundary and as such it must be assumed that they may be compromised by attack or may be purposefully malicious.

In particular the following details may be of interest to a TEE or a Trusted Application developer:

- when closing a session, there must be no pending operations within it
- when finalizing a TEE Context there must be no open sessions within its scope `TEEC_RegisterSharedMemory / TEEC_ReleaseSharedMemory`

Shared Memory is memory owned by the rich environment and mapped into the TEE memory space. Code inside the TEE and Trusted Applications must assume that the content of Shared Memory is both untrusted and volatile; data stored in Shared Memory may be changed maliciously at any time with respect to the execution of code inside the trusted environment. Note that a well formed Client Application must follow the conventions for sharing memory, as described in section A.3.2.5, in order to run with defined behavior.

Login Connection Methods

This specification defines a number of connection methods which allow an identity token for a Client Application to be generated by the Implementation and presented to the Trusted Application. This identity information is generated based on parameters controlled by some trusted entity inside the rich operating system, such as the OS kernel, and as such it is a valid security model for these login tokens to be generated by a trusted process within the rich operating system rather than by the TEE itself. Trusted Application developers must therefore note that the validity of this login token is therefore bounded by the security of the rich operating system, not the security of the TEE.

A.4.4.2 Security of the Rich Operating System

In most implementations the TEE is a separate operating system which exists in parallel to the rich operating system which runs the Client Applications. It is important that the integration of a TEE alongside the rich operating system cannot be used to weaken the security of the rich operating system itself. The implementation of the TEE Client API, the TEE, and the Trusted Application must ensure that Client Applications cannot use the features they expose to bypass the security sandbox used by the rich operating system to isolate processes.

[Quote Ends]

Note: The above represented text is a direct quote from the text published by GlobalPlatform in [GP10]. G&D's activities in contributing to the below mentioned standardization results were coordinated with G&D's activities in PrimeLife. More detailed specifications and sample code can be found in the Chapters 4ff of [GP10].