2013 IEEE 10th International Conference on Ubiquitous Intelligence & Computing and 2013 IEEE 10th International Conference on Autonomic & Trusted Computing

Enhancing the Security of Mobile Applications by using TEE and (U)SIM

Zaheer Ahmad, Lishoy Francis*, Tansir Ahmed, Christopher Lobodzinski, Dev Audsin, Peng Jiang
Trust Team, Orange Labs UK
Orange, Building 10, 566 Chiswick High Road
Chiswick Park, W4 5XS, London, United Kingdom.
(Zaheer.Ahmad;Lishoy.Francis;Tansir.Ahmed)@orange.com
(Christopher.Lobodzinski;Dev.Audsin;Peng.Jiang)@orange.com

ABSTRACT

Mobile phone platforms are increasingly becoming vulnerable to security attacks and is untrusted to host security sensitive applications, content, and services. Open source mobile ecosystems such as Android allow increased flexibility for developing and deploying applications. However, there are industry-led initiatives to increase the security of mobile phone platforms by using virtualisation and hardware abstraction techniques. In this paper, we explore the potential of the recently introduced Trusted Execution Environment (TEE) ecosystem for mobile phones in order to compliment the security-proven (U)SIM based security functions. We present a security architecture and a novel mobile payment and multimedia content playback solution leveraging on the existing post-paid billing method. We integrate TEE with (U)SIM based security techniques to provide enhanced security for user authentication, content purchase, protected storage and secure content viewing.

1. INTRODUCTION

The past few years have shown a paradigm shift in the consumers choice of personal communications and computing with increasing number of mobile phone users. In 2011, the number of smartphones sold worldwide surpassed the number of PCs sold [1]. The processing power and storage capacity of a typical mobile phone have also increased and it supports multiple communication interfaces. The conventional applications and services have also migrated to this new platform, and new interesting applications have also been found implemented. The mobile ecosystems have also evolved with open environments such as Android becoming more widely used. Albeit with these advancements, attackers have also been targeting the increasingly untrusted mobile phone [2], [3], [4]. A mobile phone user are exposed to security attacks such as keyloggers, malware, phishing and sophisticated attacks such as Man-In-The-Middle (MITM). It is also worth noting that Android application signature verification system was exposed such that attackers were able to inject malicious applications along with legitimate applications [2].

On one hand, mobile phones are becoming ubiquitous with their entry into enterprises and businesses through Bring-Your-Own-Device (BYOD) whilst on the other hand the applications on the mobile phones are subject to attacks similar to that of the attacks on personal computers. This necessitates the need to improve the security of mobile phones. The recent developments in enhancing the security of mobile phones include software and hardware virtualisation. A promising development is Trusted Execution Environment (TEE) [7] which provides hardware and software abstraction to enable a secure execution environment for mobile applications. We discuss more details on this in Section 2. We think that TEE has got the most potential in enhanced user authentication, user-friendly billing, and provisioning of content by complimenting (U)SIM based security techniques. TEE is resistant to software based attacks whereas (U)SIM based smart card is security evaluated for software as well as hardware attacks and is the accepted platform for user authentication across Mobile Network Operator (MNO) ecosystem.

The mobile billing techniques have evolved but less over the recent years mainly with pre-paid and post-paid payment options. However, there are several interesting pricing schemes currently available for the customers. Nevertheless, the pricing is regulated by local and regional legal bodies. We think that existing post-paid payment model can be utilised for the benefit of customers which introduces alternate and new ways of payment for the customers. On the other-hand, provisioning platforms of mobile applications have also evolved such as Google Play, Apple Store, Orange Primezone, etc. For instance when a user makes a purchase of an application from an application store, currently he/she needs to provide payment details such as credit or debit card, or other details such as Paypal. These methods are time-consuming and complex such that they may not seem to be attractive for a user with a valid subscription with a MNO. For such an user, it would be interesting if the payment can be made by using existing method such as the post-pay. From the perspective of the MNO it would be desirable to charge the user from his/her existing subscription as it provides increased revenue. Importantly, security is paramount for any mobile based payment. In this paper, we realise the above scenario where the subscriber can utilise existing post-paid billing method to pay for purchases of content and services.

The contributions of the paper are summarised as follows. We present a security architecture and a novel mobile payment solution leveraging on the existing post-paid mobile billing method and integrating the recently introduced TEE

*Corresponding author: Lishoy Francis (email: Lishoy.Francis@orange.com).

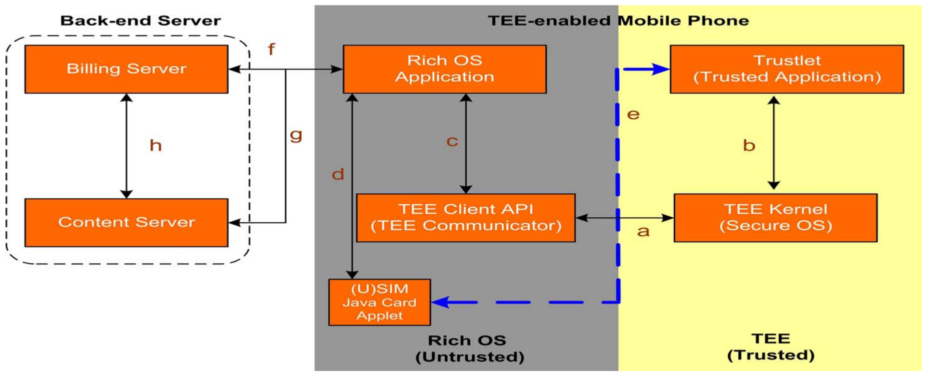


Figure 1: Security Architecture with TEE-enabled Mobile Phone and (U)SIM

with security-proven (U)SIM based security techniques. We provide details on the technical feasibility of implementing such a solution that comprises of TEE based applications working with Rich OS Applications (RA) and Java Card applet installed on the UICC platform. We present a secure channel protocol for secure communications between TEE based applications and UICC which is currently lacking in the ecosystem. To our understanding, our work is the first academic paper presenting a practical implementation of a mobile application based on TEE and utilising (U)SIM capabilities.

This paper starts with a general discussion of virtualisation, TEE and current industry initiatives and relevant standards in Section 2. We present the architecture in Section 3, and the details of our implementation of the solution on mobile phones, its effectiveness and experimental observations are discussed in Section 5. For ease of reading, we refer to post-paid billing system as "On-Bill" system and an MNO subscriber as a "user". Although we refer to mobile phone in the paper, our solution applies to devices such as tablets, smart-TV, set-top box, etc. The terms (U)SIM applet and SIM applet refers to Java Card applet installed on UICC (Universal Integrated Circuit Card) platform. UICC is a multi-application smart card on which SIM and USIM applications are provisioned, and used in mobile devices for connecting to mobile telecommunication networks.

2. BACKGROUND

In this section, we present a brief overview of TEE. We then go on to discuss related work.

2.1 Trusted Execution Environment

The concept of TEE is based on virtualisation with hardware and device-resource abstraction to host and execute security sensitive applications and store sensitive data within the mobile device platform. As previously discussed, the Rich OS environment such as Android does not provide an adequate secure environment and is vulnerable to security attacks. Typically, TEE provides execution extension to the mobile phone processor (CPU) whereby it can act in normal mode or in secure mode or in a high-secure mode. Generally, two modes are supported by the CPU: normal mode and secure mode. By design TEE provides only secondary security as compared with Secure Element (SE) [7]. TEE aims to provide security isolation of resources such as CPU, memory, GPU and storage such that only authorised applications known as a Trusted Application (TA) is provided access with, and executes within a high security context. The security core of TEE is realised by a secure kernel that manages the separation of critical processes, security domains and secure access to device resources and any associated drivers. A commercial example of secure kernel is T-Base [7] which evolved from MobiCore [13]. The "root-of-trust" is provisioned on the device by the OEM or the TEE provider. Based on this security domains or containers are provisioned to 3^{rd} party service providers who wishes to utilise TEE platform for deploying security-sensitive applications on mobile devices. These security containers are associated with cryptographic keys that allows the service provider to remotely provision applications, data and services. A set of activation and binding keys realise the management of security containers and applications associated with it. By design the security container can host multiple TAs, and the ecosystem supports multiple security containers. It is important to note that the trust relationship described above is broken once the mobile device is "rooted" or "jail-broken".

The underlying principles of TEE can be summarised as follows,

- Separation of security critical processes from normal processes on the mobile device.
- Provide isolated security containers or security domains that are capable of hosting security sensitive applications (similar to smart card security domains).
- Protect sensitive assets and device resources based on access control and cryptography.
- Provide highly-trustworthy user interface for security sensitive operations such as PIN input.

An example commercial deployment of TEE on mobile devices is based on ARM TrustZone system-on-chip technology that is now offered by Trustonic[1] [7]. The prototype discussed in this paper is based on TrustZone technology.

Several use cases have been identified for TEE [7] such as content management, mobile financial and loyalty services,

[1]Trustonic was originally founded by ARM, Giesecke & Devrient (G&D) and Gemalto/Trusted Logic Mobililty (TLM) in 2012.

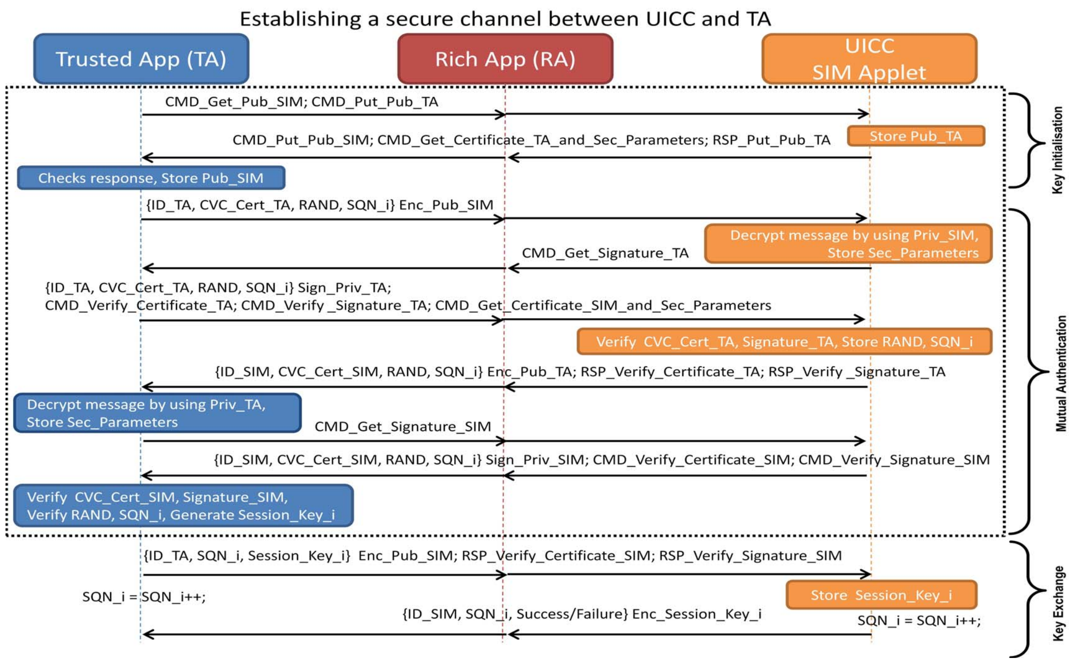

Establishing a secure channel between UICC and TA
Trusted App (TA)
Rich App (RA)
UICC SIM Applet
CMD_Get_Pub_SIM; CMD_Put_Pub_TA
CMD_Put_Pub_SIM; CMD_Get_Certificate_TA_and_Sec_Parameters; RSP_Put_Pub_TA
Store Pub_TA
Checks response, Store Pub_SIM
{ID_TA, CVC_Cert_TA, RAND, SQN_i} Enc_Pub_SIM
Decrypt message by using Priv_SIM, Store Sec_Parameters
CMD_Get_Signature_TA
{ID_TA, CVC_Cert_TA, RAND, SQN_i} Sign_Priv_TA; CMD_Verify_Certificate_TA; CMD_Verify _Signature_TA; CMD_Get_Certificate_SIM_and_Sec_Parameters
Verify CVC_Cert_TA, Signature_TA, Store RAND, SQN_i
{ID_SIM, CVC_Cert_SIM, RAND, SQN_i} Enc_Pub_TA; RSP_Verify_Certificate_TA; RSP_Verify _Signature_TA
Decrypt message by using Priv_TA, Store Sec_Parameters
CMD_Get_Signature_SIM
{ID_SIM, CVC_Cert_SIM, RAND, SQN_i} Sign_Priv_SIM; CMD_Verify_Certificate_SIM; CMD_Verify_Signature_SIM
Verify CVC_Cert_SIM, Signature_SIM, Verify RAND, SQN_i, Generate Session_Key_i
{ID_TA, SQN_i, Session_Key_i} Enc_Pub_SIM; RSP_Verify_Certificate_SIM; RSP_Verify_Signature_SIM
SQN_i = SQN_i++;
Store Session_Key_i
{ID_SIM, SQN_i, Success/Failure} Enc_Session_Key_i
SQN_i = SQN_i++;
Key Initialisation
Mutual Authentication
Key Exchange

Figure 2: Security Protocol for Establishing Secure Communication Channel between UICC and TA

trusted user interface, on-device hardware and resource isolation, data storage, strong authentication, and access control. Mobile ecosystem stakeholders such as device manufacturers and service providers including MNOs and media content providers have realised the potential benefits of TEE that it facilitates securely deploying mobile based solutions bringing increased revenues.

2.1.1 *Standards*

The core standards related to TEE are based on several GlobalPlatform [5], [8] specifications. GlobalPlatform is a not-for-profit association that identifies, develops and publishes specifications that facilitates interoperable solutions for embedded technologies. It supports specifications for TEE, especially the messaging between devices with back-end systems enabling issuance and device personalisation of TEE technology. GlobalPlatform API [9], [10] is supported on TEE-enabled mobile devices. It also supports advanced specification [17] for implementation and management of tamper-resistant chip based solutions such as SEs and smart cards.

3. PROPOSED ARCHITECTURE

In this section, we present the proposed architecture (as illustrated in Figure 1) for hosting secure applications based on TEE and (U)SIM and leveraging the On-Bill payment model. Our solution realises a generic content purchase and management service on TEE-enabled mobile devices. The proposed architecture is based on a typical TEE framework that includes a secure kernel, kernel driver, TA, and TEE connector interface API. The notation used in this paper is presented in Table 1.

As shown in Figure 1(c), RA communicates with TA by invoking the TEE Client API or the TEE Communicator interface. The TEE Client API connects the untrusted world with trusted environment (Figure 1(a)). The secure kernel driver provides communication interface with secure OS (Operating System) and in turn enables communication with any TA (Figure 1(b)). The RA has also got access to other trusted resources such as (U)SIM (Figure 1(d)). However, since the current TEE implementations does not support direct access with (U)SIM as yet, we created a secure channel for any communication between TA and (U)SIM over RA (Figure 1(e)). The secure channel protocol is detailed in Section 3.2. The provisioning of content and billing is achieved by using Back-end Server that comprises of Billing Server and Content Server, and these components of the architecture is accessed over the 3G/4G or Wi-Fi data channels (Figure 1(f, g, h)).

3.1 Integrating On-Bill Payment

We think by integrating existing MNO billing methods for mobile based applications and services would provide subscribers with added convenience. The On-Bill payment method in this paper refers to a method of payment where a subscriber can request the MNO to add the cost of his/her purchase to their post-paid subscription, typically a monthly plan. The customers would need to opt-in or sign-up for such a service. The MNO would also need to have an agreement and arrangement with the merchant who would provide such a payment facility to their customers. MNO could charge a nominal fee to the subscriber and/or to the merchant for using the On-Bill service.

On-Bill payment method enables a convenient way for subscribers to pay for services. It is provided as an alternative to other payment methods such as credit or debit card, PayPal, mobile wallet, etc. On-bill payment service is not a competing, rather a complementary payment option as compared to other existing payment options available today. The On-Bill payment method is typically realised in practice by integrating it with the merchant portal or by having a dedicated application on the mobile device that communicates with merchant's back-end server. The details of the On-Bill system is out of scope of this paper.

3.2 Secure Channel Protocol

Currently, the TEE-enabled mobile devices do not support direct access to (U)SIM from TEE. So any communication between (U)SIM and TEE needs to be protected by using a secure channel. A secure channel protocol enables an on-card entity such as an applet on (U)SIM and an off-card entity such as TA within TEE to mutually authenticate, and to establish cryptographic keys to protect integrity and confidentiality of any subsequent communications across untrusted communication channels such as over Rich OS environment. We propose a secure channel protocol based on asymmetric and symmetric cryptographic techniques. This protocol is motivated from [14].

Table 1: Notation used in this paper.

TA	Trusted Application.
RA	Rich OS Application.
UICC	Universal Integrated Circuit Card.
Pub_X	Public Key of entity X.
Priv_X	Private Key of entity X.
CMD_Get_Pub_X	Command to retrieve Pub_X.
CMD_Put_Pub_X	Command to store Pub_X.
Store_Pub_X	Store Pub_X.
RSP_Put_Pub_X	Response for CMD_Put_Pub_X.
Cert_X	Certificate of entity X.
CVC_Cert_X	Certificate of entity X in CVC [15] format.
ID_X	Unique Identifier of entity X.
RAND	Random Number.
SQN_i	Sequence Number used in i^{th} session.
Enc_Pub_X	Encrypt by using Pub_X.
Sign_Priv_X	Generate signature by using Priv_X.
Verify_X	Verify signature or certificate.
Session_Key_i	i^{th} Session Key.
K_s	Session Key.
ICCID	Integrated Circuit Card Identifier.
MSISDN	Mobile Station International Subscriber Directory Number.
BS	Billing Server.

3.2.1 *Description*

The goal of secure channel protocol is to mutually authenticate UICC based applet with TA, and to compute and exchange a symmetric session key that is used to secure any subsequent communication occurring during a session or a transaction. We use asymmetric cryptographic techniques to achieve mutual authentication. Mutual authentication, integrity, and confidentiality, are achieved by the sender encrypting and signing the message. Signing also provides non-repudiation. The sequence number and random are used to detect and prevent any replay and masquerade attacks.

Figure 2 illustrates the secure channel protocol between UICC or (U)SIM and TA. It involves the follow steps.

- Key Initialisation: This step involves exchanging the public keys of both TA and applet residing in the UICC.
- Mutual Authentication: Both parties are mutually authenticated by verifying their certificates. This step starts by TA sending ID_TA, its certificate in CVC format, RAND and SQN all encrypted with public key of SIM based applet. This ciphertext is then decrypted by the applet by using its private key. Security parameters such as RAND and SQN is temporarily stored by the SIM. The SIM then requests signed data of the original message from TA. The signature thus provided by TA is then verified by SIM based applet. In response SIM provides its certificate, RAND and SQN encrypted with TA's public key and then signature of this message. Once the signature is verified and certificate checked for correctness, both TA and UICC are mutually authenticated. TA then generates the session key that is used to secure any subsequent communication.
- Key Exchange: TA sends the generated session key and SQN to UICC encrypted with public key of SIM. It also sends the response for signature verification to SIM and increments the SQN. SIM based applet then stores session key, increments the SQN and returns the result of this operation. The session key is valid only for this particular session. For a new session, a new session key is generated based on the above steps.

3.3 Role of (U)SIM

(U)SIM is a key MNO asset and is capable of providing "root-of-trust" in any MNO deployed service based on cryptographic keys, certificates, security policies, PIN, etc. In the proposed architecture, (U)SIM plays an important role in enabling trust to any services provided. (U)SIM is capable of identifying and authenticating the user for billing purposes. The underlying tamper-resistant platform is adequate to store user and application data securely. With the "root-of-trust" placed in the (U)SIM, it is convenient to port any application security when the user changes the mobile device to a new one. In the proposed architecture, (U)SIM is responsible for user authentication, transaction validation, storing cryptographic keys for securing content, and establishing secure communications via secure channel with any TA residing within the TEE.

3.4 Trusted Application

TA executes within the TEE context and is protected by means of software and cryptographic isolation. Such an application is typically a small binary code that implements TEE APIs. They are cryptographically signed, securely loaded and is responsible for security sensitive operations. TA is also known as a Trustlet. Any TA communicates with other applications such as RA over TEE Client API that provides a connector interface between secure kernel and Rich OS. TA has also got access to secure storage within TEE. TA facilitates the creation of a secure channel with other trusted entities such as (U)SIM for the purposes of exchanging security sensitive data. It plays a key role in securely capturing user credentials such as PIN or password and executing security sensitive operations such as decrypting secured content. A work in progress within the GlobalPlatform TEE roadmap is securely displaying and managing the user interface [11]. However, commercially available TEE-enabled mobile device do not support secure user interface API as yet.

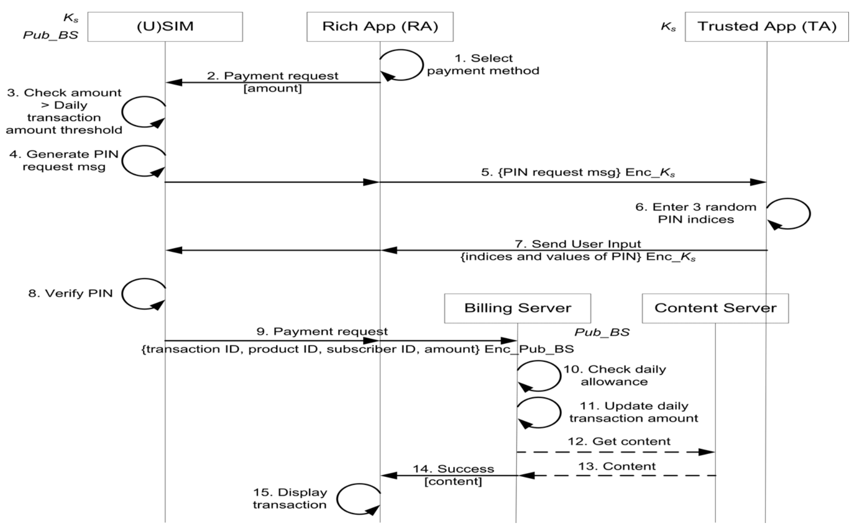

K_s
Pub_BS
(U)SIM
Rich App (RA)
K_s
Trusted App (TA)
1. Select payment method
2. Payment request [amount]
3. Check amount > Daily transaction amount threshold
4. Generate PIN request msg
5. {PIN request msg} Enc_K_s
6. Enter 3 random PIN indices
7. Send User Input {indices and values of PIN} Enc_K_s
8. Verify PIN
Billing Server
Content Server
9. Payment request {transaction ID, product ID, subscriber ID, amount} Enc_Pub_BS
Pub_BS
10. Check daily allowance
11. Update daily transaction amount
12. Get content
13. Content
14. Success [content]
15. Display transaction

Figure 3: On-Bill Payment Data Flow Diagram

3.5 Rich OS Application

RA serves as the entry-point to any secure application or processes residing within TEE and (U)SIM platforms. It interacts with (U)SIM on behalf of the TA as needed, based on the established secure channel between (U)SIM and TA. RA plays a role in managing non-security operations such as displaying transaction details. It communicates with the Back-end Server over available data channels such as mobile Internet and Wi-Fi. Due to the limitations in current TEE API [12], where the mobile devices do not yet support direct communication with (U)SIM; RA currently facilitates the communication between TA and (U)SIM. RA communicates with TEE Client API or TA Connector in order to interact with TEE Kernel. Albeit, with RA operating in a untrusted context secure channel protocol presented in Section 2 is needed for any secure-sensitive operations.

3.6 Back-end Server

For design simplicity, Billing and Content Servers have been integrated into a single server called Back-end Server. The Back-end Server interacts with any relevant database as required. The Billing Server receives transaction and payment messages from the mobile device. The transaction and payment details are processed as needed based on the business and application logic. Content server is responsible for distributing and managing any media content. It also facilitates the purchase of the available content. It works in conjunction with Billing Server and the user's mobile device.

4. USE CASES

In this section, we discuss few use cases that we have identified for our proposed security architecture that was presented in Section 3. We describe the use cases based on a application called "Pay & Play". This application enables an user to purchase digital content from an online store or an application provisioning it by using On-Bill payment method. As with any payment process, On-Bill system requires high-security in order to correctly bill the user. This would involve user authentication and transaction validation within a trusted environment. We envisage that our proposed architecture is capable of realising "Pay & Play" securely. We proceeded further to implement a proof-of-concept of "Pay & Play" solution by using TEE and (U)SIM. "Pay & Play" solution on the user mobile phone include TA, RA and corresponding (U)SIM applet. This solution is described in Section 5.

4.1 Secure On-Bill Payment

As presented in Section 3, the components of "Pay & Play" application consist of TEE-enabled mobile phone, (U)SIM, Content Server and the Billing Server. An online content store makes the content available for the users to purchase by using Content Server. RA provides interface to this content store for the users. The On-Bill payment method is typically realised in practice by integrating it with the merchant portal or by having a dedicated application on the mobile device that communicates with merchant's back-end server.

Figure 3 shows the detailed user journey such as PIN entry, validation and transaction check, etc, upon user choosing the On-Bill payment method for purchasing, downloading and viewing any media content.

As a first step, the user downloads and installs the "Pay & Play" application from application store onto a TEE-enabled mobile phone. The user then launches the application. By default this application runs in the Rich OS context, and selects any content available on the online store to purchase. He/she is then given an option to pay for the content by using On-Bill payment (illustrated in Figure 4(a)).

Upon user confirming the amount, the transaction message is sent to (U)SIM in order to authenticate the user and to check the security and payment policies. A payment policy controls the maximum amount that a user can pay by On-Bill payment without entering PIN (in this case, the payment is implicitly authorized using (U)SIM).

(U)SIM interacts with TEE for secure PIN verification depending on the payment policy stored in (U)SIM. For instance, PIN verification is required if the transaction amount is above a threshold value. This threshold value and a daily allowance limit is set by the service provider such as the

MNO. After the completion of purchase, the "Pay & Play" application communicates with the digital media content server to securely download the content and then makes it available on the mobile phone. Throughout the entire process of content download and media viewing the digital rights of the content provider are maintained.

The main purpose of TEE is to complement (U)SIM security functions by securely capturing the user PIN and verifying within a secure execution environment in order to securely process On-Bill payment. The communication between TEE and (U)SIM is encrypted. The user identity is fulfilled by (U)SIM application to generate the payment and service details such as unique transaction ID, amount, subscriber ID, etc.

4.2 Content Management

TEE is capable of securely provisioning licensed digital contents to a user. For instance, user with a TEE-enabled mobile phone can download protected content and then securely view in accordance to the policies set by the content provider. During the process of the content purchase, the Content Server ensures that the content is encrypted in such a way so that it can only be decrypted by the device with the authorized (U)SIM that stores the required license. The encrypted content is first downloaded to the user's mobile phone. If the content is stored in the Rich OS security context in encrypted form. When the user wants to view the content, TEE decrypts and displays it on a content viewer. For this TA interacts with (U)SIM to retrieve the required license file. In this case (U)SIM acts as the "root-of-trust" controlling the entire trust management. The user can view the content purchased as he/she wishes on the authorized device according to the policies set by the Content Server.

4.3 Secure PIN Input

One of the use cases identified for TEE is secure input of credentials in a high-security context. A user can be identified and authenticated based on universally accepted security mechanism such as a PIN. At the time of writing this paper, none of the commercially available TEE-enabled mobile phone supported secure display in order to realise a secure PIN entry mechanism. Therefore, we think current security can be improved by using a custom-built randomised PIN entry method (illustrated in Figure 4(b)) rather than the standard approach supported on Rich OS ecosystems. We think that a PIN entry mechanism by using randomized keypad layout can be developed that utilises secure communication between TEE and (U)SIM. Here we rely on the secure channel between TA and (U)SIM to perform the PIN validation process.

This method is considered as relatively secured because the PIN is still captured in the Rich OS security context. However additional security mechanisms are deployed to safeguard PIN entry process. The process is secure as long as the device is not rooted/jail-broken. In a rooted/jail-broken device, one can argue that a hacker can potentially install a sniffing tool to capture the PIN. But a rooted/jail-broken phone will also destroy the secured boot process of a TEE enabled handset, which implies that the installed TA will no longer work. On the contrary, TEE controlled secured display based PIN entry system is entirely isolated from the Rich OS context; hence is always secured even if the mobile phone is rooted/jail-broken. During the PIN entry and verification, there are a number of interactions between (U)SIM and TEE. The pre-configured PIN is stored in (U)SIM, which verifies the user's entered digits. The user is asked to enter random parts of PIN and this information is communicated by TA to the (U)SIM.

The process can be summarised as follows:

1. (U)SIM initiates the PIN validation process.
2. (U)SIM generates a PIN request message and sends it in encrypted form to TA.
3. TA prompts RA to display random PIN input UI and user to enter random PIN fields. TA then captures the partial PIN and the corresponding PIN indices.
4. TA encrypts the user input and sends it to the (U)SIM for validation.

4.3.1 *Secure PIN Initialisation and Update*

The above technique can be used to initialise and also update the user PIN. For instance, at the time of installing "Pay & Play", a default PIN can be generated by (U)SIM. Then the option is made available to the user to setup the secret PIN. The process can be summarised as follows:

1. RA initiates the PIN initialization/changing process.
2. (U)SIM generates a PIN request message and sends it in encrypted form to TA.
3. TA captures the PIN data by using RA's randomised PIN UI.
4. TA encrypts the user input and sends it to the (U)SIM for storing.

5. PRACTICAL IMPLEMENTATION

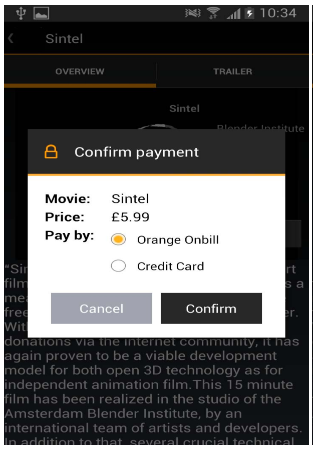
As a proof of concept of the proposed architecture, we implemented our "Pay & Play" prototype. In this section, we describe the details of this practical implementation. "Pay & Play" prototype implementation has the following components: RA, TA, TEE Communicator Interface, Java Card applet installed on the (U)SIM and the Back-end Server. The RA and TA was realised on a TEE-enabled Android based device such as Samsung Galaxy S3 which supports secure OS TBase-198 provided by Trustonic. The Android version used is 4.2.2.

5.1 Rich OS Application

The RA is based on an Android application that implemented a front-end for an application store. The application store provided content for users to purchase and was integrated with On-Bill billing system. RA communicates with Back-end Server over mobile Internet.

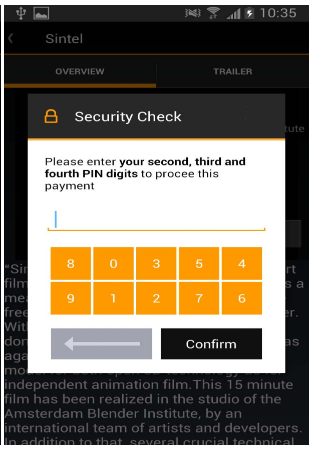
The RA is responsible for following:

- Interact with (U)SIM to purchase a digital media content.
- Initiates and facilitates creation of secure channel between (U)SIM and TEE for the purpose of transferring PIN related data.
- Forward PIN request to TEE initiated by (U)SIM, captures PIN in normal world and sends it to TEE.
- Send transaction details to billing server and display transaction.



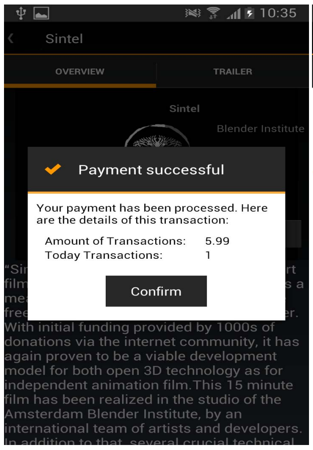
Movie: Sintel
Price: £5.99
Pay by: [✓ Orange Onbill] Credit Card

(a) On-Bill Payment Option



Please enter **your second, third and fourth PIN digits** to procee this payment

(b) Randomised PIN UI

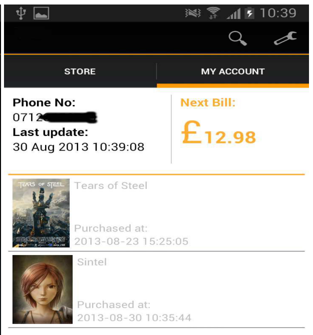


Your payment has been processed. Here are the details of this transaction:

Amount of Transactions: 5.99
Today Transactions: 1

With initial funding provided by 1000s of donations via the internet community, it has again proven to be a viable development model for both open 3D technology as for independent animation film. This 15 minute film has been realized in the studio of the Amsterdam Blender Institute, by an international team of artists and developers.

(c) Payment Confirmation



Phone No: 0712[redacted]
Last update: 30 Aug 2013 10:39:08
Next Bill: £12.98

Tears of Steel — Purchased at: 2013-08-23 15:25:05
Sintel — Purchased at: 2013-08-30 10:35:44

(d) On-Bill Details

Figure 4: Prototype screenshots.

5.2 TEE Communicator

TEE Communicator or TA Connector (TLC) facilitates communication between the "Pay & Play" application in Rich OS (Android), and the corresponding TA residing within TEE, and it consists of a Java Native Interface (JNI) and TA Interface. The TA Interface that communicates with the secure OS kernel is written in C language, which interfaces with TA through the kernel module by accessing raw memory. This code is then compiled down to processor level instructions. The JNI layer provides a Java interface to the TA Interface. The TA Interface uses the secure OS driver API (such as MobiCore) to initiate starting and stopping of a TA and to exchange any data with it. To the upper layer, e.g. the RA (Android application), it makes the TA's security features accessible. Therefore, the TA Interface was developed based on [21] MobiCore 1.6 API. RA then communicates with TA via its specific API.

5.3 TEE Application

The TA is implemented on a TEE enabled device, Samsung Galaxy S3 based on TBase-198 SDK [7]. Each TA is assigned with a UUID (Universal Unique Identifier). The load process of a TA can be summarised as follows: the secure kernel driver lookup the UUID of the TA in the file system and then loads it to the registry. Once the TA installed it runs perpetually, it is loaded when the mobile phone is booted and runs until it is unloaded or un-installed explicitly. Once unloaded its run-state gets cleared, any parameters associated with the TA such as application program counters, security associations, etc. It is worth noting that as any TA would need to utilise the device resources such as memory, it is a best practise not to keep TA loaded when not needed. This is to cater the resourcing needs of other TAs. Currently, the shared memory reserved for communication between RA and TA is limited and finite. TA implemented the functions discussed in Section 4. The TA is responsible for the following:

- Establish a secure channel with (U)SIM for the purpose of exchanging security sensitive data such as PIN.
- Initiate PIN request to RA, capture input and send it for verification to (U)SIM.
- Retrieve cryptographic key from (U)SIM for the decrypting the license file.
- Decrypt the secured content by using (U)SIM based cryptographic key.

5.4 Java Card Applet

(U)SIM applet is based on Java Card platform [16] that supports GlobalPlatform card management system [17]. RA communicates with (U)SIM applet via SIM Alliance Open Mobile API (Smart Card API/ SEEK) interface [18] by sending and receiving ISO 7816 [19] based APDUs. (U)SIM applet supports PKCS#15 access control mechanism defined in [20]. The access to (U)SIM applet is restricted and the access rules are stored on (U)SIM card. (U)SIM applet supports cryptographic algorithms such as RSA, AES and random number generation. The (U)SIM applet is responsible for the following:

- Check the payment policy (transaction limit) on behalf of the On-Bill payment module.
- Create a secure channel with TA for the purpose of exchanging data (PIN related data).
- Initiate PIN request to TA and verifies the PIN.
- Create and securely transfers the payment details (unique purchase ID/transaction, amount, subscriber ID, etc.) to the billing server.
- Update daily transaction amount that is used to verify daily allowance.

5.5 Billing and Content Server

For the simplicity of the demonstrator, we integrated the billing and the content server onto a single server (Back-end Server) that interacts with the relevant database.

The billing server has application logic attached to a back-end database. The application logic processes and calculates the transaction information, while the back-end database acts as a persistent data store for the transaction information. The billing server receives encrypted messages from the (U)SIM containing the payment details. The messages are encrypted by (U)SIM with a Server's public key. The payment message from (U)SIM contains the transaction ID, user ID, product ID and price of the product in the Tag Length Value (TLV) format. The Table 2 shows the format of the payment message sent from (U)SIM to the billing server.

Table 2: Payment Message Format

Transaction ID	User ID	Product ID	Price
$MSISDN \| RAND$	$ICCID$		

The Billing Server then decrypts the received message and extracts the required values. It then queries the database to calculate the transaction amount within the last 24 hours. The server calculates the sum of all the transactions made by the user within the last 24 hours, verifies the sum against the user's daily allowance, and sends the total payment amount and number of transactions in the past 24 hours to the mobile phone for further processing.

The Content Server is responsible for providing the purchased media content to the user. The Content Server generates, distributes, and manages the purchased media content. It communicates with the On-Bill Server as well as with the (U)SIM on the TEE-enabled mobile phone. The main components of the Content Server platform include media file store or content store, content management to encrypt the media files, and a license management platform. The Content Server provisions the license file containing the content decryption keys to (U)SIM by encrypting the data with (U)SIM's public key. The content management module arranges the encrypted content along with the corresponding content decryption key.

6. CONCLUSION

TEE-enabled in the mobile phones use trusted software and hardware to provide an isolated environment for secure processing and storage. TEE addresses some of the security problems and lack of trust in Android ecosystem due to rooting and re-flashing with custom software. It provides an additional layer of security binding such that the trust worthiness of a mobile phone is removed once it is rooted or re-flashed. Several use cases can benefit from TEE in combination with the security capabilities of (U)SIM. On-Bill payment can be used to securely buy digital media content by using TEE and (U)SIM based security credentials.

In this paper, we described a security architecture for securing mobile applications by using the recently introduced TEE ecosystem for mobile devices along with security-proven (U)SIM mechanisms. We presented practical implementation feasibility of this proposed security framework by prototyping an application, "Pay & Play", that uses both TEE and (U)SIM security capabilities.

(U)SIM plays a major role as the "root-of-trust" by providing necessary security keys and certificates, security policies and storing the user's authentication PIN. TEE-enabled On-Bill payment prototype has been implemented on a mobile phone in which Trustonic's secure OS creates and maintains the TEE security container. Since currently secure PIN capturing is not available, randomized keypad based PIN capturing mechanism has been proposed, which has been generated under control of the TEE application. As future work, we would be looking into the performance of the proposed architecture on different platforms. We note that provisioning of trusted applications is not yet opened-up for third parties within the TEE ecosystem. Further research is needed in this area.

7. REFERENCES

[1] Smart phones overtake client PCs in 2011, Canalys, February 2012, Online: `http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011`.

[2] J. Forristal, Android: One Root to Own Them All, Blue Box, DEFCon 2013, USA.

[3] Top 10 Mobile Security Risks, DecompilingAndroid, Online: `http://www.decompilingandroid.com/mobile-app-security/top-10-mobile-security-risks/`

[4] T. Muller, M. Spreitzenbarth, and F. C. Freiling, FROST: Forensic Recovery Of Scrambled Telephones, In Proc. The 11th International Conference on Applied Cryptography and Network Security (ACNS 2013), Banff, Alberta, Canada, June 2013, Online: `https://www1.informatik.uni-erlangen.de/frost`.

[5] Global Platform, Online: `http://www.globalplatform.org/`.

[6] TrustZone (TZ) Technology, Online: `http://www.arm.com/products/processors/technologies/trustzone.php`.

[7] Trustonic, Online: `http://www.trustonic.com/products-services/trusted-execution-environment/`.

[8] GlobalPlatform Device Technology, TEE System Architecture, v1.0, December, 2011.

[9] GlobalPlatform Device Technology, TEE Client API Specification, v1.0, July, 2010.

[10] GlobalPlatform Device Technology, TEE Internal API Specification, v1.0, July, 2011.

[11] GlobalPlatform Device Technology, Trusted User Interface API Version, v0.99, March 2013.

[12] GlobalPlatform Device Technology, TEE Secure Element API, v0.65, April 2013

[13] G&D MobiCore, Online: `http://www.gi-de.com/`.

[14] L. Francis, G. P. Hancke, K. Mayes, and K. Markantonakis, A Security Framework Model with Communication Protocol Translator Interface for Enhancing NFC Transactions, Proceedings of The Sixth Advanced International Conference on Telecommunications (AICT 2010), pp. 452-461, May 09-15, Barcelona, Spain, IEEE Computer Society, 2010.

[15] ISO/IEC 7816-8: Identification cards Integrated circuit cards Part 8: Commands for security operations, International Standard, 2004.

[16] Oracle/Sun Microsystems, Java Card Platform Specification v2.2.1.

[17] Global Platform, Card Specification v2.1.1.

[18] SIM Alliance Open Mobile API, Smart Card API (SEEK).

[19] International Organization for Standardization, ISO/IEC 7816.

[20] Global Platform, "Secure Element Access Control", Version 1.0, Public Release, May 2012.

[21] MobiCore Application Developers Guide, Version 1.6, Giesecke & Devrient GmbH, 2011.