(12) **APPLICATION**

(11) **20172033** (13) **A1**

(19) NO

**NORWAY**

(51) Int Cl.

***G06Q 20/32 (2012.01)***
***G06Q 20/12 (2012.01)***
***G06Q 20/38 (2012.01)***
***G06Q 20/40 (2012.01)***
***H04L 9/32 (2006.01)***
***H04W 12/06 (2009.01)***
***G06Q 30/06 (2012.01)***

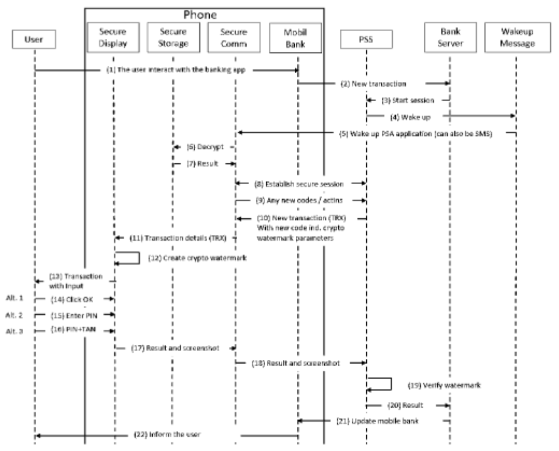
**Norwegian Industrial Property Office**

| | | | | |
|---|---|---|---|---|
| (21) | Application nr | 20172033 | (86) | Int. application day and application nr |
| (22) | Application day | 2017.12.22 | (85) | Entry into national phase |
| (24) | Date from which the industrial right has effect | 2017.12.22 | (30) | Priority |
| (41) | Available to the public | 2019.05.06 | | |

| | | |
|---|---|---|
| (71) | Applicant | Protectoria AS, Klingenberggt. 7b, 0161 OSLO, Norge |
| (72) | Inventor | Anders Moen Hagalisletto, Skibakkeveien 3B, 0881 OSLO, Norge<br>Erik Vasaasen, Høyenhallveien 2A, 0667 OSLO, Norge |
| (74) | Agent or Attorney | OSLO PATENTKONTOR AS, Postboks 7007 M, 0306 OSLO, Norge |

(54) Title **Secure mobile platform**
(57) Abstract

A system for securing transactions performed on a mobile platform comprising a user interacting with a banking application on a mobile terminal, said banking application communicating with a banking server, said banking server triggering a secure server when extra security is required, said secure server activating a secure application on said mobile terminal and the secure application interacting with the banking application initiating required extra security, characterised in that said extra security comprises an executive code compiled, obfuscated and transferred just-in-time before each use, all communication is encrypted, and a communication protocol which is unique to the device, everything displayed on the mobile terminal can be watermarked with a cryptographic code identifying the mobile terminal and current operation and changes to data communication, executive code and content displayed on the mobile terminal can be detected by the secure server.


Phone
User
Secure Display
Secure Storage
Secure Comm
Mobil Bank
PSS
Bank Server
Wakeup Message
(1) The user interact with the banking app
(2) New transaction
(3) Start session
(4) Wake up
(5) Wake up PSA application (can also be SMS)
(6) Decrypt
(7) Result
(8) Establish secure session
(9) Any new codes / actins
(10) New transaction (TRX) With new code incl. crypto watermark parameters
(11) Transaction details (TRX)
(12) Create crypto watermark
(13) Transaction with input
Alt. 1
(14) Click OK
Alt. 2
(15) Enter PIN
Alt. 3
(16) PIN+TAN
(17) Result and screenshot
(18) Result and screenshot
(19) Verify watermark
(20) Result
(21) Update mobile bank
(22) Inform the user

## Technical field

The present invention regards a system and a method for secure transactions using a mobile platform, and more particularly a system and method for securing transactions performed on a mobile platform regardless if a hostile third party has infected the device with malware or not.

## Background of invention

There is currently a major shift happening in how end users are accessing their bank accounts and retail web sites. The majority of end users are no longer using web browsers on computers; they are using apps on mobile phones and tablets. This creates a major challenge for service providers, as the majority of mobile units are not receiving security updates in a timely manner. A further complication of the situation is that while mobile phones used to be a safe 2nd channel for authorization and authentication, the user is now expecting to use a single device for everything.

As the internet is going mobile, and in many regions most user sessions are already from Android™ or iOS™. This means that many users are using a single device i.e. no PC, no extra hardware dongles, only a smartphone. Customers might have an old laptop in a closet, but when it breaks they might not replace it. Companies must follow this trend, or lose customers, both current and future. This raises a major problem, in particular with Android devices since in most cases they are only updated as long as both the device manufacturer and network provider agree to push updates to the devices.

As this in practice does not happen for anything but the current generation devices the result is that most Android phones are as secure as e.g. a PC running Windows XP™, which no longer receives even security updates from Microsoft. An example of such vulnerability in Android is with Stagefright™, which is used for decoding multimedia on Android™ devices newer than Android 2.2™. This well-known security vulnerability means that devices that have not been updated are vulnerable to remote code execution and privilege escalation through ordinary MMS messages, without the user even opening the message. The result of the privilege escalation is that a remote attacker can gain full “root” access to the phone, in many ways similar to how malware on Windows™ get full access to a computer.

Once malware is present on the phone the malware distributor can in practice see everything which happens on the screen, record everything written on the keyboard, read and change all files, including protected ones, and even put images and text on top of other applications. It should be noted that this form of malware is different from a "rooted device", where the user is in control of what software runs with higher privileges. Using software, it is simple to detect that a device is rooted by the user, but it can be impossible to tell if a device is infected with malware with root access.

Using mobile apps for banking tends to be safer than logging in via your mobile browser. However, every mobile platform has unique characteristics that these apps must prepare for. Developers may not fully understand the risks associated with mobile banking and accidentally leave vulnerabilities open for fraudsters to exploit as a result.

**Summary of invention**

It is therefore an object of the present invention, as stated in the set of claims to solve the problems mentioned above.

This is done by introducing a Secure Mobile Platform (SMP) that enables smartphones to do operations requiring a high level of security under the assumption that the device is infected with malware, and that the underlying operating system and network thus cannot be trusted.

The Secure Mobile Platform (SMP) comprises two parts: A main application, which we presume to be untrusted. In the following explanation this is designated as MobileBank. Further, there is a secured application or library, which comprises the following components: a secure display, which is the part of a Secure Application (SA), deals directly with the user, building an interface and wherein watermarking happens. Further there is a Secure Storage (SecureStorage), wherein encryption and storage of the private key is done. A Secure Application environment (SecureApp), deals with application "business logic", and code blocks. A Secure Communication module (SecureComm) deals with encryption, signing and communicating with the Secure Storage. The user downloads and installs the necessary software to the phone from an AppStore which typically is either Google play™ or iTunes™. A Secure Server (SS) handles connections from the SAs. A phone service (PhoneService) makes calls using the Public Switched Telephone Network (PSTN). The MobileBank communicates with a banking server (BankServer), and the BankServer triggers the SS when extra security is required. Finally, a Push Notification Service (PNS) avoids a battery draining persistent connections between the client and the server and an operating system messaging level service is used for waking up the SA.

The goal of the present invention is to enable Payment Service Providers and eCommerce vendors to implement apps that are immune even to malware targeting their app in particular, and even if malware is created for targeting one installation of an app it does

not automatically translate to targeting every other installation.
In order to protect the app the Secure Mobile Platform implements a number of innovative security mechanisms, such as invisible watermarks, just-in-time executable code (depending on platform), and continuously changing protocols and storage. As this requires a lot of extra software complexity the Secure Mobile Platform is provided in two ways:

As a separate executable that can be bundled with an existing application, but branded to look similar to it, and which only purpose is to secure authentications and transactions.

As a SDK (Software Development Kit), where parts of the full Secure Mobile Platform functionality can be implemented in an existing application.

In the first case the user should not notice switching between existing/non-secure functionality, but practice this means that the application is split in two: A normal, fast, user friendly part, and a “super secure” part, which only handles the secure tasks. In the second case it is up to the developers of the application to use functionality from the SDK as they see fit.

The present invention has the following security mechanisms to help ensure this:

- The executive code which is to be run on each device is transferred just-in-time, before each use.
- All communication is encrypted, and the protocol is unique to the device.
- Everything displayed on the smartphone can be watermarked with a cryptographic code identifying the device and current operation.
- Changes to data communication, executive code, content displayed on the smartphone can be detected server-side.
- Transaction integrity is ensured and checked wherever possible.

## Brief description of the drawings

Figure 1 is a systems overview of the entire system and a display of the communication flow according to a preferred embodiment of the present invention.

Figure 2 is a systems overview of the entire system and a display of the communication flow according to alternative embodiment of the present invention.

Figure 3 is an example of the detailed communication flow according to the embodiment of the present invention.

Figure 4 is a detailed overview of the modules involved in a regular scenario of an embodiment of the present invention.

Figure 5 is a detailed structure of the sequence of modules according to an alternative embodiment of the present invention.

Figure 6 is a basic overview of the secure app according to a preferred embodiment of the present invention.

Figure 7 is an overview of the communication flow in the secure app according to a preferred embodiment of the present invention.

Figure 8 is an example of the first three steps (1-3) using the secure app, here it is displayed a user login with a PIN.

Figure 9 is an example of steps 4-6 in using the secure app, here it is displayed a transaction authorization with OK.

Figure 10 is an example of steps 5-7 in using the secure app, here it is displayed a transaction authorisation using a PIN.

Figure 11 is an example of steps 4-9 in using the secure app, here it is displayed a PIN entry with a phone call for a Transaction Authentication Number (TAN).

Figure 12 is an example of using the secure app in a Financial Transaction Services (FinTS) using a PIN entry with a Transaction Authentication Number (TAN) for authorisation.

## Detailed description

Figure 1 is a systems overview of the entire system and a display of the communication flow according to a preferred embodiment of the present invention.

Here it is displayed the relationship between a mobile app (“Example Bank”) and the SA application:

In this figure the Mobile banking app and the Secure App (SA) runs on the smartphone, as separate executables, but bundled together when distributing. The SA communicates with the Secure Server (SS). The Mobile Banking App communicates with Bank server. The Bank server calls the SS when extra security is required. There is no direct connection between the Mobile Banking App and the Secure Application (SA) except during installation and enrolment.

The Secure Server (SS) communicates with banking server(s) and the SAs. The Secure Server itself is quite simple, but calls other more complicated programs. It consists of two main parts:

A control and report module, the Process Controller (PC):

- A simple “control panel” for reporting the state of the service:
    - Status of the server and any services the server depends upon
    - List of all SAs, and their state
    - Enabling or disabling functionality
    - Showing hacking attempts
    - Displaying latest transactions
- A framework for reporting possible security violations to the banking server through a message queue, which not specified here.

A communication module:

- Functionality for communicating in a proper encrypted manner with the SA.
- Functionality for keeping track of SAs, their state, and which versions of code blocks are running on each SA.
- Functionality for initializing a new SA on enrolment (i.e. first start of the application).
- An API interface for calls from the bank server,

In order to use the API interface the bank server connects through HTTPS, where the SS is configured to require a certificate signed by the master certificate on the SS to access the API interface. Without the correct client certificate communication with the SS is impossible. As the SA is assumed to be located in the same secure server environment as the banking server exactly what security is required on the API interface is up to the security requirements of the bank or hosting provider.

When there is a requirement for extra authentication or transaction authorization the SA is called on the smartphone, the result is returned back to the SS, which returns it back to the bank, updating the interface. It is important to note that we assume that the functionality for authentication or transaction authorization is moved to the SA, and that the “business logic” of the transaction verification is done on the server side. E.g. it is not possible to simply skip the transaction authorisation by manipulating the banking app.

In addition to the secure entry of a PIN / TAN and the secure display of the transaction data, the SA / SS also include functionality for having a voice call being part of the authorization as an extra channel.

As can be seen the transactions come from the bank server, it is received by the Secure Server, then passed on to the Secure Application before seen by the user. On every step there are mechanisms for ensuring the integrity of the transaction. The purpose of the integrity mechanisms is that (a) to ensure that the transaction details are transferred correctly to the user, (b) that there are no malware modifying what happens inside the Secure Application, (c) that there are no attempts at replay attacks or tricking the Secure Server by creating false transactions, (d) that the information shown on the screen is directly linked to the input from the bank server through a watermark, (e) that the keys used are correct, and finally (e) that the result is transferred correctly back to the Secure Server.

Bank server: The security of the bank server is implicit – if the back- end services of the bank are compromised the attacker would not require the fraudulent transactions to go through the SMP at all.

The information involved is customer information, transaction details and risk assessment of the transaction.

Secure Server: Once received, the transaction metadata is stored on the SS. Parts of this metadata, such as a timestamp, are used to ensure that an attacker is not trying to do various forms of replay attacks, where the same transaction is returned to the SS multiple times.

The information involved is text to be displayed to the user, possibly a phone number if a voice call is required, a RefID, a Device ID for connecting to the correct SA installation.

Secure App:

Secure Communication: Secure communication uses PKI mechanisms. The information involved is pinned PKI keys for the server, PKI keys for the SA, code blocks for protocol obfuscation, Hardware ID for symmetric encryption.

Secure Application: This layer ensures the following:

- That the transaction is processed with “just in time” blocks of code
- That if the PIN of the user is entered it is encrypted with the public key of the banking server, so that it cannot be decrypted by the SS.

- That the code blocks used have not had their integrity compromised.

The information involved is code blocks, information required to build the user interface and reply (text, code blocks involved, background images), checksums, pinned key for bank server.

Secure Display: Metadata about the transaction, such as a reference ID (RefID) and a timestamp is encoded in the image shown to the user, so that changes to the image are detected. On submit a screenshot is taken, and this screenshot is passed together with what the user entered back to the Secure Server.

The information involved is for the watermark: RefID, timestamp, information used for cryptographic encoding (nounce, device ID)

Secure Server (SS): All transaction data is verified here, and integrity checks from the SA are compared with the expected values. The result is passed on to the Bank Server for the final decision regarding if the transaction should be considered valid.

The information involved is the user input (possibly encrypted), a watermarked screenshot and transaction metadata (If the device is rooted or not, checksums of blocks, if checks of the accelerometer detected movement and if any honeypots were trigged).

Bank server: The mobile banking server receives the result of the transaction. The result consists of the following three factors:

- The result of user input (e.g. encrypted PIN, if the TAN was entered correctly, or if the user pressed OK or Cancel).
- The result of watermark check.
- The result of verifying the application integrity.

Together these three factors can be used to calculate the risk of compromise.

Figure 2 is a systems overview of the entire system and a display of the communication flow according to an alternative embodiment of the present invention.

In this embodiment called the Okay scenario, there are three additional entities in addition to the scenario described in figure 1.

Distribution separate from any application, where the app is used purely as a 2nd factor (the Okay scenario). In this scenario a single app can connect to more than one possible service, making this a multi-tennant security service.

One major potential use case for the Okay approach is that the separate application can function as a 2nd channel, as an addition to an already protected main application.

OkayApp: The Okay application, which deals with application “business logic”, and code blocks. This is similar to the SA described above, but implemented as a separate application.

TenantServer: This server is equivalent to the BankServer in the previous scenario.

MultiTenantGateway: The Gateway allows for multiple TenantServers to communicate with a single SMP instance.

Figure 3 is an example of a simplified view of the process and communication flow according to the embodiment of the present invention.

This is a simplified view of the process and communication flow and a lot of information is not presented in this diagram.

The SA is split into three different parts, secure display, secure storage and secure communication. This means that the secure application layer is not shown. The secure application layer would receive new code blocks for showing a dialog to the user, new code blocks for encryption, as well as possibly one or more honeypots.

In the above sequence diagram the assumption is that the transaction authorization is for a single transaction, but it can also be for multiple transactions, where the user can switch between transactions and authorize them individually.

1: The first action is that the user does something on the mobile banking app.

2: The mobile banking app communicates with the Bank server.

3: The user action is recognized by the banking system as requiring being verified by the end user, in this case a transaction authorization process. While the process is taking place, the mobile banking app is waiting.

4: The bank server connects to the Secure Server (SS) in order to initiate the transaction authorization process.

5: Here there are two alternatives: (a) SS (using a cloud messaging service on behalf of SS) sends a wake up message to the SA, and (b) the Mobile Bank app sends the wakeup message directly to the SA. This wakeup message does not contain any sensitive information, but it does contain a pattern which when used with white box encryption on the SA can be used to read a file private to the SA.

6/7: The information to decrypt the secure storage is received from the SS encrypted with a session key sent by the SA to the SS.

8: The SA now has access to its private key, and can establish a secure connection to the SA. The two parties are now able to authenticate 2 ways and exchanges SS/SA session encryption keys, provide Pin Pad generation parameters etc. The goal is here both to establish a secure connection and to be sure that the SS is connecting to the correct SA.

9: The SS and SA exchanges new code blocks within the secure channel.

10: The SA transfers the code blocks required to show a transaction authorization screen to the user.

11: Once the data and code blocks are received by the secure communications layer the data can be transferred to the secure display.

12: The secure display builds the screen, and watermarks the entire screen with the transaction details, the IMEI of the mobile phone and any hardware identificators available. In addition to the transaction details the information used to watermark the screen is all known by both the SA and the SS.

13: The transaction authorization screen is displayed to the user. Depending on how security critical the transaction is the bank might decide to do one of the following:

14: Have the user click OK (scenario 1).

15: Have the user enter their PIN (scenario 2).

16: Enter a PIN + a TAN from a call (scenario 3).

17/18: If a PIN is entered it is encrypted with the public key of the banking server and transferred together with the screenshot to the SS.

19: The SS verifies the watermark and checksums.

20: The result (encrypted PIN + metadata for the transaction) is transferred back to the banking server.

21: The banking server updates the mobile banking app

22: The user is informed by the mobile banking app.

Figure 4 is a detailed overview of the modules involved in a regular scenario of an embodiment of the present invention.

The modeling covers several aspects of the solution lifecycle. The models of the Secure Mobile Platform (SMP) system include the specifications of five main interactions, in addition to a collection of three concrete examples deploying the three authentication scenarios

Platform setup: Assumptions for the setup of the present invention. This includes the creating of appropriate keypairs for both SS and the BankServer, and the required exchange of the public keys between the SS component and the BankServer component.

Mobile bank app installation: This includes the installation of a SA in a phone – typically through the installation or upgrade of new Banksoftware – providing the appropriate public key, hard coded into the SA (pubPSSPinned), and an appPNStoken provided by the PNS.  After installation the user is not yet enrolled.

Receiving POTC: This protocol is describing behavior outside the SMP solution. User information is submitted from the MobileBank to the BankServer, which returns the POTC to the MobileBank.

Enrollment of the SA: User triggers enrollment after installation, the required credentials are transmitted to the SS – (including the appPNStoken, the client public key kpClient), and the SA creates a long term keypair to be used in the communication and authentication with the SS. Enrollment happens the first time the app is run, or in the OKAY case can be trigged by a secondary app (this is dependent on implementation scenario). Hardware identifying tokens, such as serial numbers are collected in this step.

Starting a bank transaction: User triggers a new bank action (eg. money transfer) in mobile banking app, then the app calls bank server which creates a list of parameters which will be used during authentication.

Starting a SMP session: The bank server asks the SS to start a new authentication session and transfers a list of parameters which will be used during authentication.

Waking-up the SA: The SS sends a wakeup message to the SA through a notification message, using the PNS. This is done with a cloud message, e.g. Firebase Cloud Messaging on Android™ and Remote Notifications on iOS™.

Handshake: In this step the transmission channel between the SA and SS is secured, and initial keys are exchanged.

Obtaining session parameters: Once the session is secured the SA can connect to the SS to download the code blocks and other information required to initiate the session with the end user.

Submission of ‘OK’: This is authentication scenario 1.

Submission of PIN: This is authentication scenario 2 (and a part of scenario 3).

Submission of authentication results: The results of authentication (user input and integrity verification results) are returned to the SS.

Submission of TAN): This is authentication scenario 3, where a phone call is involved.

Checking the SMP session status: The bank server asks the SS for session status and in case session is completed initiates the transaction finalization steps.

Finishing the bank transaction: Bank server checks an authentication session status, finishes a transaction and sends result to a user.

Figure 5 is a detailed structure of the sequence of modules according to an alternative embodiment of the present invention.

Multitenant platform setup: Assumptions for the setup of the OK solution. The interaction equals the setup in the standard scenario, except that MutiTenantGateway plays the role of the BankServer.

Registering a tenant: A new TenantServer registers to a MutiTenantGateway. A tenantID and a tenantSecret are constructed.

Installing the Okay application: This includes the installation of a SA on the mobile terminal through AppStore™.

Initializing Okay application: The User starts the application for the first time, then immediately afterwards, an appPNStoken is requested and returned. After this step it is possible to enroll the user.

Regular enrolment for the SA:  A regular enrollment procedure is performed.

User-tenant linking: As the user wants to use two factor authentication, a link to a new TenantServer is established. The MultiTenantGateway is providing the linkingcode to be used in the association of a tenantID and a linkingcode.

Starting a tenant SMP session: Authentication starts with the TenantServer creating the required parameters, and then triggers an authentication request on the MutiTenantGateway.

Starting a SMP session: The MultitenantGateway creates the artifacts required for authentication and triggers a standard authentication scenario with SS.

Waking-up the SA: The SS sends a wakeup message to the SA through a notification message, using the PNS. This is done with a cloud message, e.g. Firebase Cloud Messaging on Android and Remote Notifications on iOS.

Handshake: In this step the transmission channel between the SA and SS is secured, and initial keys are exchanged.

Obtaining session parameters: Once the session is secured the SA can connect to the SS to download the code blocks and other information required to initiate the session with the end user.

Submission of “OK”: This is authentication scenario 1.

Submission of PIN: This is authentication scenario 2.

Submission of authentication results: The results of authentication (user input and integrity verification results) are returned to the SS.

Finishing the tenant SMP session: The MultitenantGateway checks the session status in the SS and in case the session is completed it performs analysis of session artifacts and transfers the result of analysis to the TenantServer.

Figure 6 is a basic overview of the secure app according to a preferred embodiment of the present invention.

The SA consists of four major parts:

- A secure application layer, which implements just-in-time replacement of code blocks.
- A secure communication layer which ensures that the client is talking to the correct server, in a way which cannot be intercepted by third parties.

- A secure storage layer, where keys and other information is stored in a secure manner.
- A secure display layer, which checks for manipulation of the screen.

The secure application layer protects the other parts of the SA as illustrated in this diagram:

The secure application layer protects the display, storage and communication layers. In practice these three layers are implemented as code blocks which are received and run through the secure application layer.

Figure 7 is an overview of the communication flow in the secure app according to a preferred embodiment of the present invention.

Data, such as transactions requiring authorization, is received through TLS encrypted mobile data or Wi-Fi, encrypted text messages or a combination. The raw data is then processed by a secure communication layer, which has regular format changes based on new code blocks from the secure application layer. In addition keys are stored in the secure storage layer, which is rewritten regularly in a both encrypted and obfuscated manner. On top is the secure display layer, which watermarks and displays the information, using transaction data and information unique to the device.

The resulting input and screenshot is then passed down through the stack and back to the SS, so that the SS can verify that the correct information was shown before passing the result on to the banking server.

The second level is the secure communication layer, where application level encryption of communication takes place.

This involves decryption and signatures, using standard PKI in addition to symmetric encryption based on hardware IDs and shared secrets.

On the third (application) and fourth level (storage) the secure execution environment is established, and secure storage is performed. The code blocks are created on the fly for each transaction. Creation of a code block involves compiling and obfuscating the source code for the user interface together with the transaction details. This makes the execution environment unique for each invocation. There are an almost unlimited potential number of different versions of each code block, each with a somewhat different obfuscation. The SS keeps track of what is supposed to be running on the SA, and any difference in protocol, hardcoded encryption secrets or running versions is reported. The application layer is also responsible for exchanging the code blocks running the display layer and interfacing with the communication layer, so that it is never the same code running for a long period of time.

A voice call can be used as a separate channel for providing information to the user and for recording user input.

Figure 8 is an example of the first three steps (1-3) using the secure app, here it is displayed a user login with a PIN.

SECURITY LEVELS FOR AUTHENTICATION:

Based on how many mechanisms are used there are different levels of security, with varying degrees of verification that the user actually is the person he/she is claiming to be. Roughly ranked:

1. Low risk: PIN entry for login, with transaction authorization by pressing “OK”.

2. Medium risk: PIN entry for login, followed by PIN entry for transaction authorization.

3. High risk: PIN entry for login, followed by PIN entry combined with voice call for entering a TAN received in a voice call for transaction authorization.

A TAN is only used in the third alternative, where the user also has to enter information seen on the screen in a voice call. As it is significantly harder for malware to manipulate voice calls than to manipulate applications this represents a higher level of security, which would be suitable for larger transactions or unknown recipients.

Depending on the security level and risk assessment a number of honeypots can be added to the compiled code blocks. If the code in any of these honeypots is modified or called it will trigger a warning sent to the secure server.

AUTHENTICATION WITH PIN:

The use-case scenario here is as follows:

1. The user clicks on their mobile phone banking application as normal.

2. When PIN entry is required the SA application pops up. The SA is branded similarly to the normal mobile phone banking application, so this switch of active application is invisible to the user.

3. On entering the PIN the SA application is closed, leaving the user back on the banking app. The banking application shows the user as logged in.

In the above images the difference between the Example Bank app and the SA has been exaggerated. In a production environment the SA would have the same branding as the app being protected. Normally there would be some text in the SA authorization similar to “To authorize login to Example Mobile bank please enter your PIN” in the above image.

In the second image, the PIN entry, there are two features worth noting:

- The randomised keypad. This is to deter attempts at recording what the user enters simply by recording where the user presses. (Exactly what keypad is used is up to the banking server)
- The background, which can be unique to each instance. Combined with slightly translucent buttons this will make it harder for screen readers. The exact look and branding is customizable.

After the user successfully enters the PIN the focus goes back to the mobile bank app.

Not visible in the screenshots of the SA is the invisible watermark feature. This allows the SA to encode encrypted information about the login/transaction in what is displayed on the screen, then transfer this to the server, so that the server can be sure that the user’s device has not been tampered with.

Figure 9 is an example of steps 4-6 in using the secure app, here it is displayed a transaction authorization with OK.

This scenario would follow the scenario for login with PIN. The use-case scenario here is as follows:

4. After entering a transaction the user is prompted to authorize the transaction.

5. The SA pops up with details of the transaction, and a button marked “Authorize”.

6. After pressing “Authorize” the user is returned to the Example Bank, where it says, “Your transaction has been approved”

Figure 10 is an example of steps 5-7 in using the secure app, here it is displayed a transaction authorisation using a PIN.

This scenario would follow the scenario for login with PIN. The use-case scenario here is as follows:

4. The end user tries to transfer money to a new recipient, but which is considered medium risk.

5. The user is asked to verify the transaction.

6. The SA pops up, with the following information:

   a. Details of the transaction, as decided by the bank server.

   b. The user is asked to enter their PIN.

7. After entry the control is passed back to the mobile bank, which displays that the transaction is authorized.

Figure 11 is an example of steps 4-9 in using the secure app, here it is displayed a PIN entry with a phone call for a Transaction Authentication Number (TAN).

This scenario will typically only come into play for large transactions, or where the bank sees some other reason to increase the security level. As in the two previous scenarios this scenario would follow the scenario for login with PIN.

The use-case scenario here is as follows:

4. The end user tries to transfer a large amount to a new recipient.

5. The SA pops up, with the following information:

   a. Details of the transaction, as decided by the bank server.

   b. The user is asked to verify the transactions with their PIN.

6. The users enters their PIN, and presses “OK”

7. A screen is shown telling the user that she will receive a call and a TAN generated by the SS.

8. The call starts by repeating basic information about the transaction, ending with asking for the TAN. Exactly what is said is decided by the banking server.

9. The user enters the TAN during the call, using the phone keyboard.

10. The user is sent back to the banking app, where the result of the authorization is shown.

The method described above works on both Android™ and iOS™. On Android™ it is additionally possible to silently take the call, and to display an application on the screen during the call, which makes it possible to (a) make the customer enter data from the SA on the phone’s keypad during the call, or (b) make the customer enter data from the voice call on a keypad shown by the SA. In scenario (b) the flow is as follows:

7. A screen is shown telling the user that he/she will receive a call.

8. When the user accepts the call a window is shown on top of the screen showing relevant transaction data and a TAN. The TAN is contained in the messages received from the SS. There are two alternatives for the TAN: It can be displayed on the screen, requiring the user to enter it again, or it can be read out to the user, requiring the user to enter it through the phone keyboard.

9. The call starts by repeating basic information about the transaction, ending with the user being asked to enter the TAN. Exactly what is said is decided by the banking server.

10. The user enters the code from the displayed window in the call using the keypad.

11. The user is sent back to the banking app, where the result of the authorization is shown.

Figure 12 is an example of using the secure app in a Financial Transaction Services (FinTS) using a PIN entry with a Transaction Authentication Number (TAN) for authorisation.

This scenario supports older applications, which still require entry of TAN. We believe this scenario offers more security than SMS TAN, and the display of transaction information with the SA is more secure than found in other solutions

The use-case scenario is as follows:

1. The user enters a transaction in an app, which supports SMS TAN but is not integrated with the SA.

2. The transaction details are shown, and the user selects “Authorize with SMS TAN”.

3. The SA pops up, with the following information:

a. Details of the transaction, as decided by the bank server.

b. The user is asked to verify the transactions with their PIN.

4. The SA shows a TAN, the text “Please verify this TAN in the next screen” and a button for Next.

5. The user clicks on Next.

6. The user is sent back to the app requiring a TAN, where the TAN is already filled out.

7. The user clicks on “Authorize” to authorize the transaction.

**Claims**

1. A secure application for securing transactions performed on a mobile platform comprising a banking application on a mobile terminal communicating with a banking server, said banking server triggering a secure server which activates said secure application on said mobile terminal and the application interacting with the banking application, c h a r a c t e r i s e d  i n that said secure application comprises a secure application layer, which implements just-in-time replacement of code blocks, a secure communication layer, where transaction details are compiled into obfuscated code blocks together with the source code required to run a secure application environment, a secure storage layer, where keys and other information is stored and a secure display layer, which checks for manipulation of the screen.

2. A secure application for securing transactions according to claim 1 wherein said checks manipulation of the screen is in the form of watermarking.

3. A secure application for securing transactions according to claim 1 wherein depending on a security level and a risk assessment a number of honeypots can be added to the code blocks.

4. A secure application for securing transactions according to claim 2 wherein levels of security  is low risk comprising PIN entry for login, with transaction authorization by pressing “OK”.

5. A secure application for securing transactions according to claim 2 wherein said levels of security is medium risk comprising PIN entry for login, followed by PIN entry for transaction authorization.

6. A secure application for securing transactions according to claim 2 wherein said levels of security is high risk comprising PIN entry for login, followed by PIN entry combined with voice call for entering a TAN received in a voice call for transaction authorization.

7. A secure application for securing transactions according to claim 1 wherein said secure server can communicate with one or more secure applications installed on one mobile terminal.

8. A secure application for securing transactions according to claim 1 wherein said mobile terminal can have one or more banking applications installed which interacts with one or more secure applications.

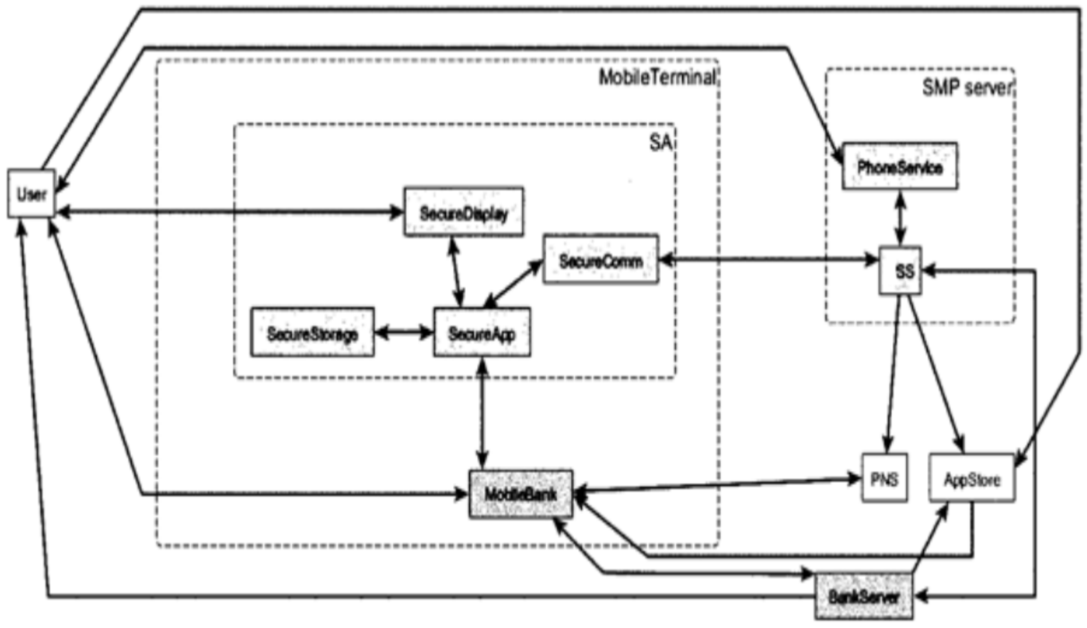


Figure 1

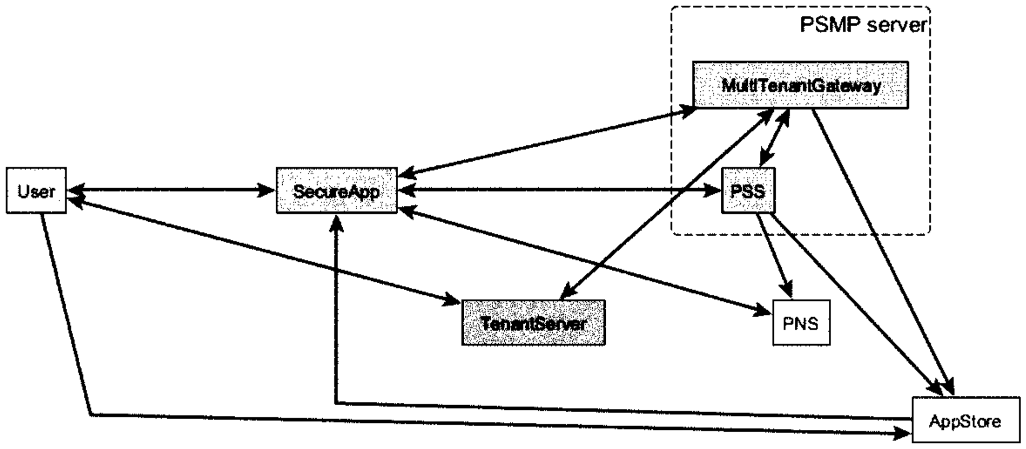


Figure 2

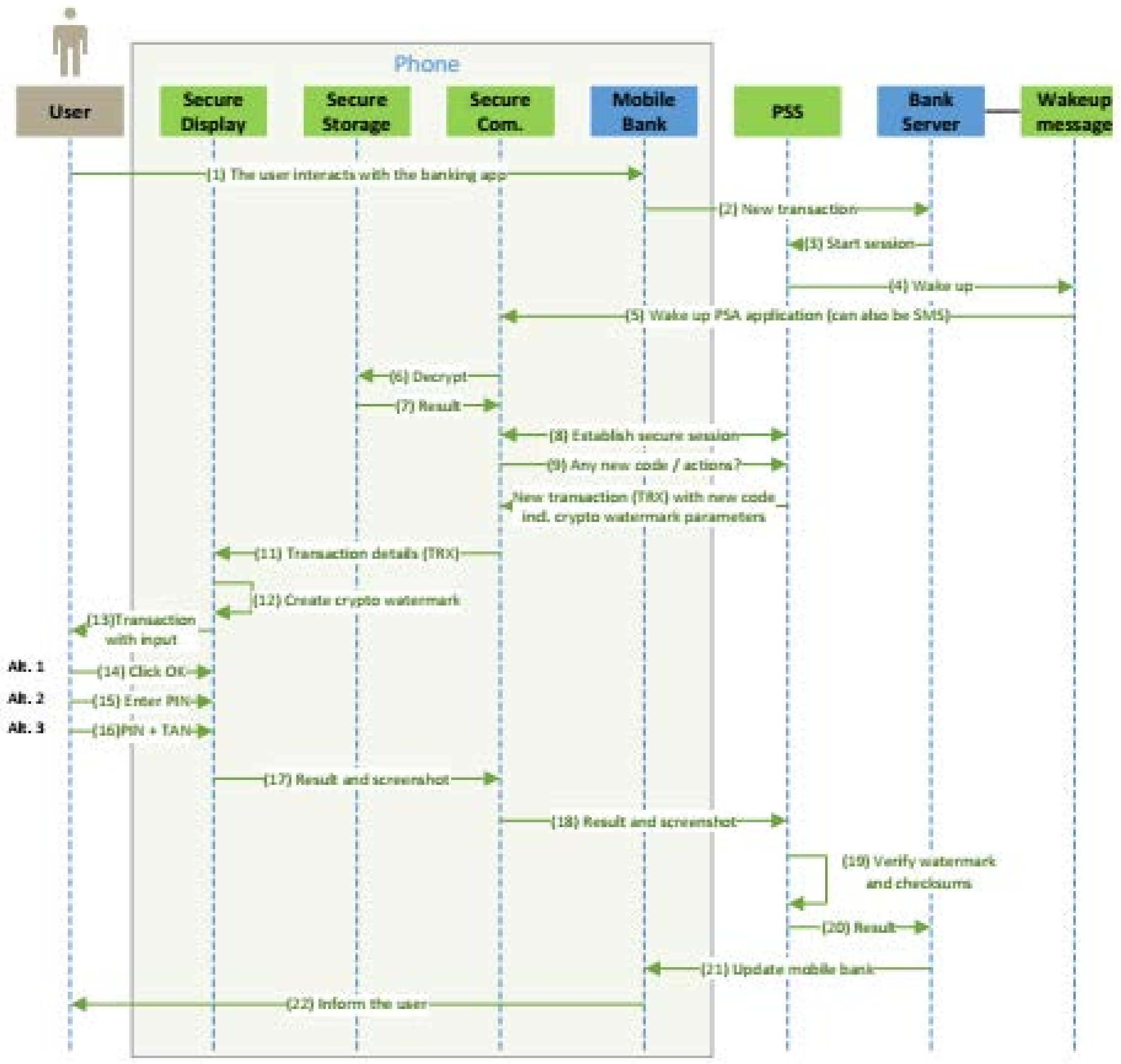

Phone
User
Secure Display
Secure Storage
Secure Com.
Mobile Bank
PSS
Bank Server
Wakeup message
(1) The user interacts with the banking app
(2) New transaction
(3) Start session
(4) Wake up
(5) Wake up PSA application (can also be SMS)
(6) Decrypt
(7) Result
(8) Establish secure session
(9) Any new code / actions?
New transaction (TRX) with new code incl. crypto watermark parameters
(11) Transaction details (TRX)
(12) Create crypto watermark
(13)Transaction with input
Alt. 1
(14) Click OK
Alt. 2
(15) Enter PIN
Alt. 3
(16)PIN + TAN
(17) Result and screenshot
(18) Result and screenshot
(19) Verify watermark and checksums
(20) Result
(21) Update mobile bank
(22) Inform the user

Figure 3

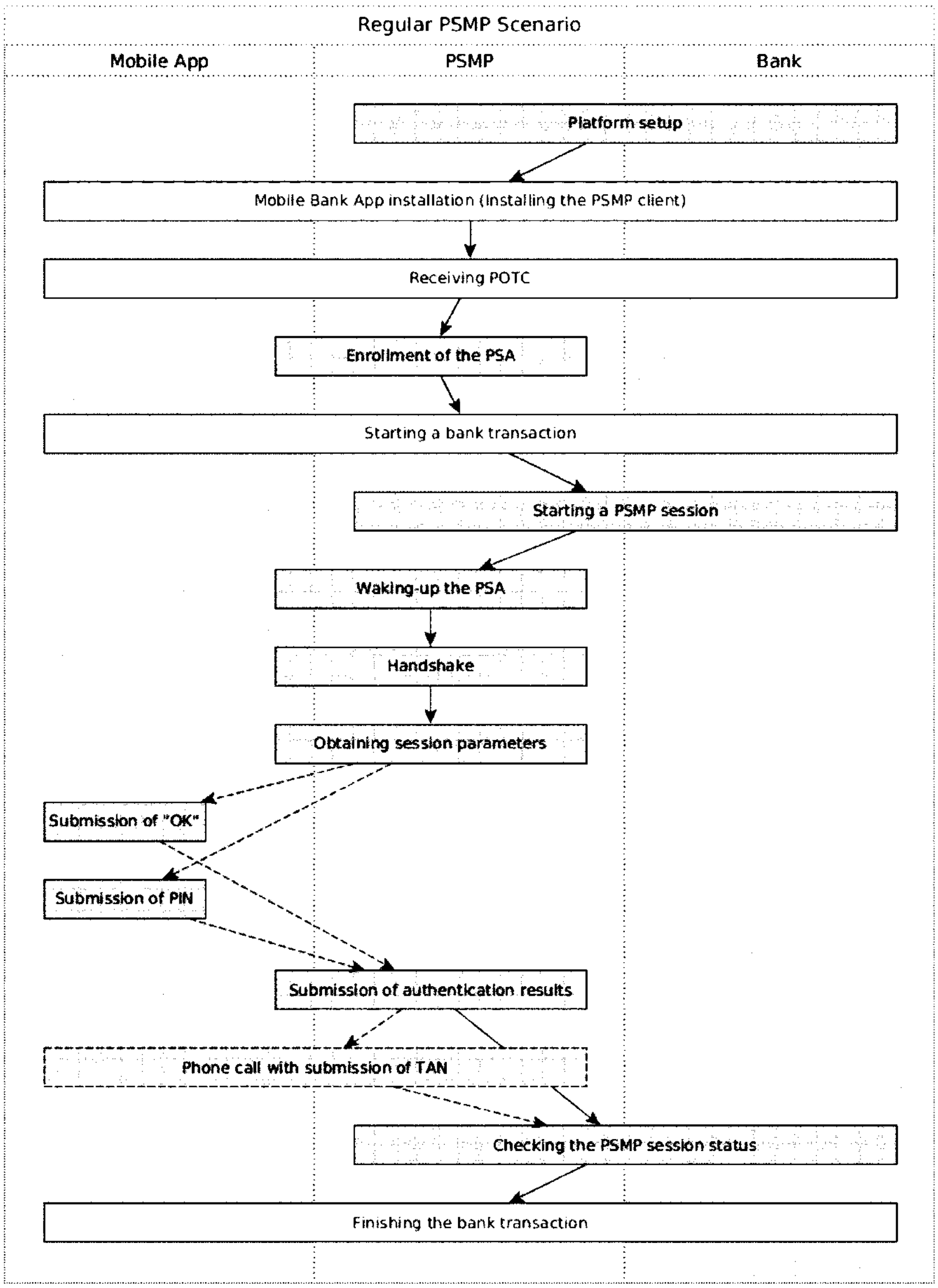

Regular PSMP Scenario
Mobile App
PSMP
Bank
Platform setup
Mobile Bank App installation (Installing the PSMP client)
Receiving POTC
Enrolment of the PSA
Starting a bank transaction
Starting a PSMP session
Waking-up the PSA
Handshake
Obtaining session parameters
Submission of "OK"
Submission of PIN
Submission of authentication results
Phone call with submission of TAN
Checking the PSMP session status
Finishing the bank transaction

Figure 4

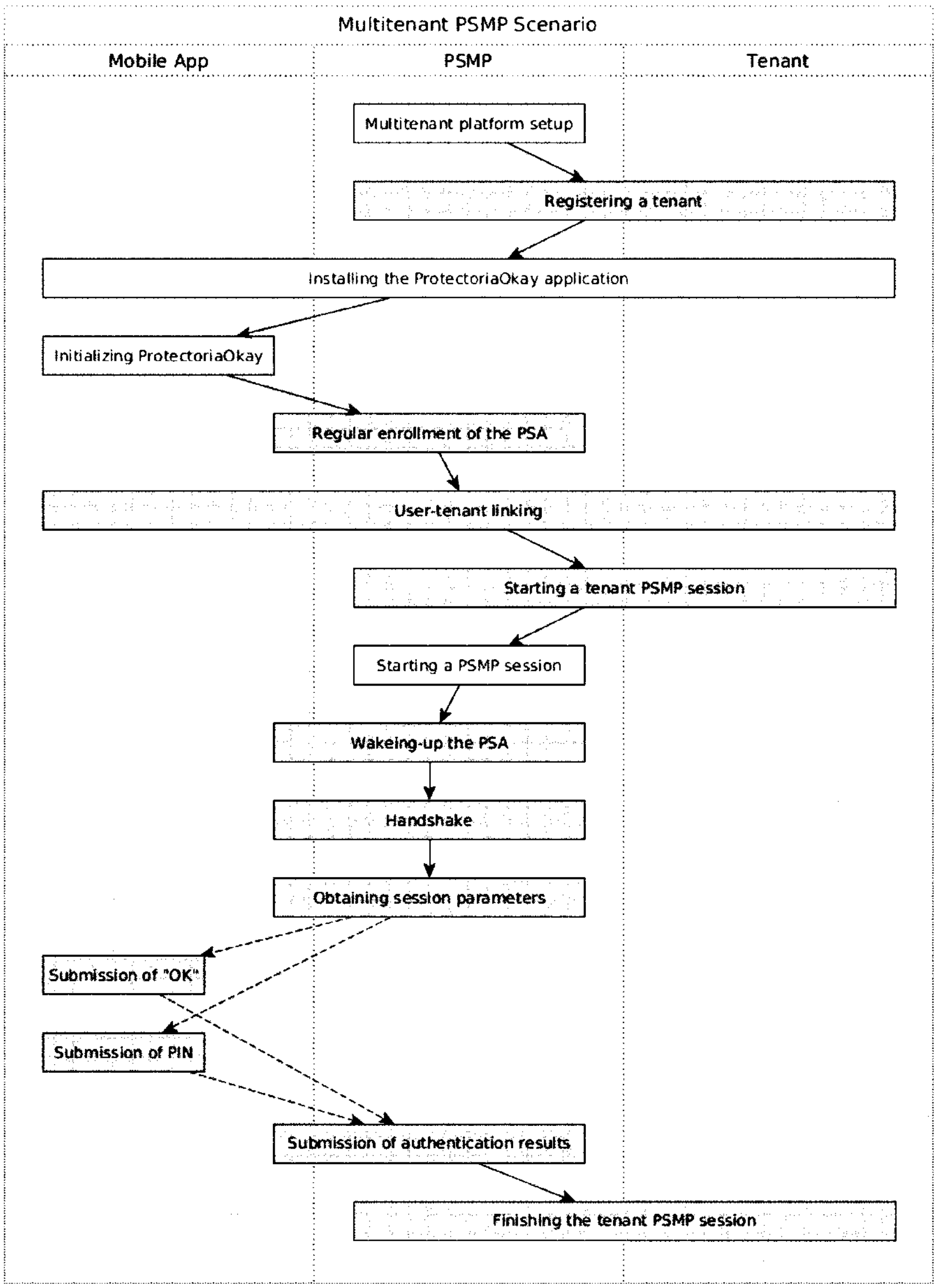

Multitenant PSMP Scenario
Mobile App
PSMP
Tenant
Multitenant platform setup
Registering a tenant
Installing the ProtectoriaOkay application
Initializing ProtectoriaOkay
Regular enrollment of the PSA
User-tenant linking
Starting a tenant PSMP session
Starting a PSMP session
Wakeing-up the PSA
Handshake
Obtaining session parameters
Submission of "OK"
Submission of PIN
Submission of authentication results
Finishing the tenant PSMP session

Figure 5

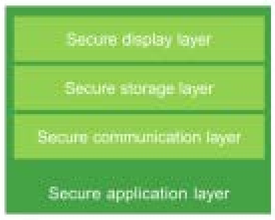


Figure 6

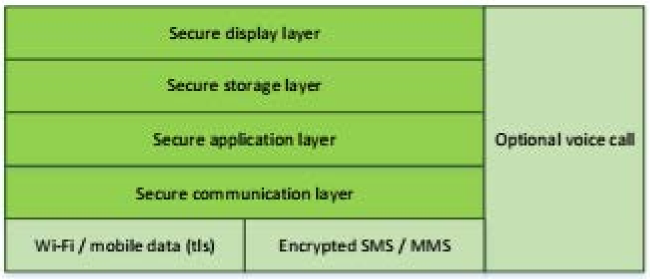


Figure 7

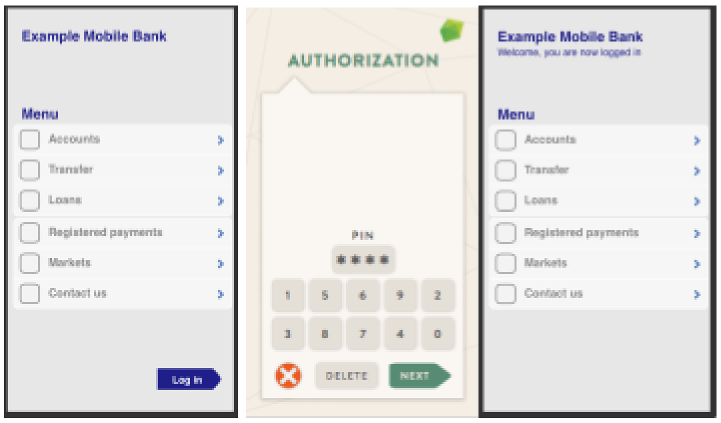
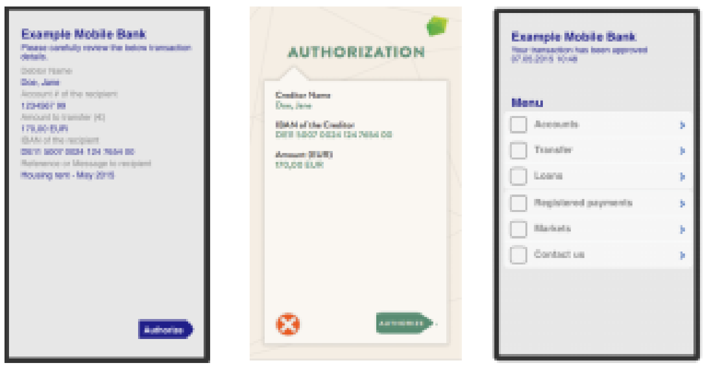


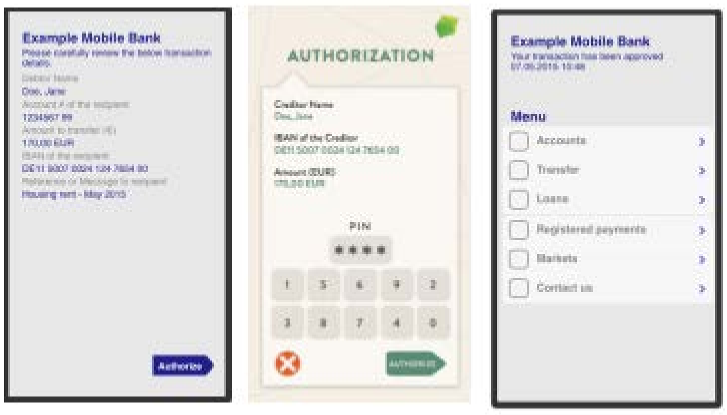
Figure 8


Example Mobile Bank
Please carefully review the below transaction details.
Debtor Name
Doe, Jane
Account # of the recipient
1234567 89
Amount to transfer (€)
170,00 EUR
IBAN of the recipient
DE11 5007 0024 124 7654 00
Reference or Message to recipient
Housing rent - May 2015

AUTHORIZATION
Creditor Name
Doe, Jane
IBAN of the Creditor
DE11 5007 0024 124 7654 00
Amount (EUR)
170,00 EUR

Example Mobile Bank
Your transaction has been approved
07.05.2015 10:48

Figure 9



Example Mobile Bank
Please carefully review the below transaction details.
Debtor Name
Doe, Jane
Account # of the recipient
1234567 89
Amount to transfer (€)
170,00 EUR
IBAN of the recipient
DE11 5007 0024 124 7654 00
Reference or Message to recipient
Housing rent - May 2015

Authorize

AUTHORIZATION

Creditor Name
Doe, Jane
IBAN of the Creditor
DE11 5007 0024 124 7654 00
Amount (EUR)
170,00 EUR

PIN
✱✱✱✱

1 5 6 9 2
3 8 7 4 0

Example Mobile Bank
Your transaction has been approved
07.05.2015 15:48

Figure 10

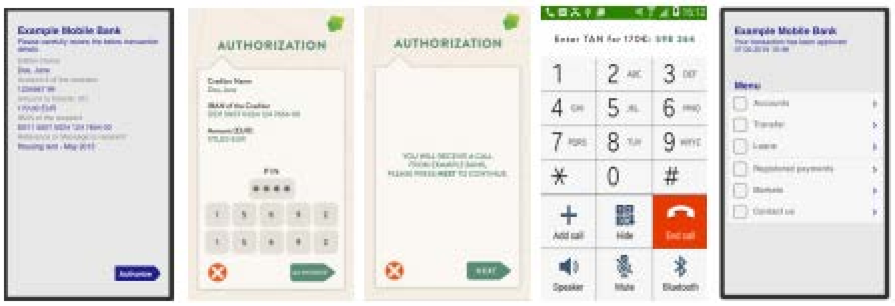


Figure 11

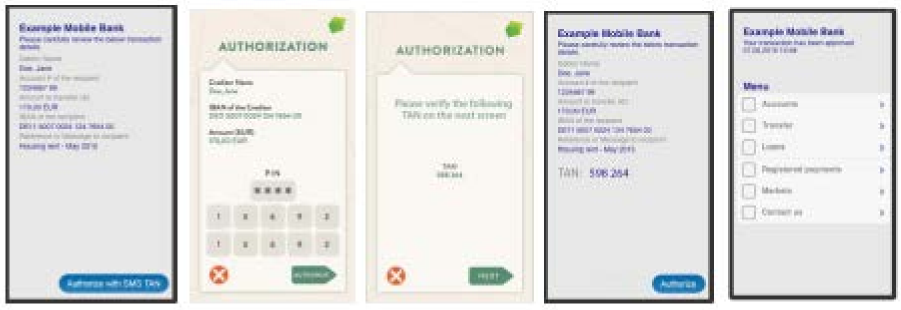


Figure 12