Project Number 683612

V2 of the VIMpay app

1.0

11 February 2016

Final

Public distribution

petaFuel

Project Partner Contact Information

petaFuel GmbH
Ludwig Adam
Muenchnerstrasse 4
85354 Freising
Germany
Tel: +49 8161 40 60 202
E-Mail: ludwig.adam@petafuel.de

Table of Content

Document Control

Version	Status	Date
0.1	Document outline and first content	02.01.2016
0.5	first version	11.01.2016
1.0	Final Review	11.02.2016

Executive Summary

This document constitutes deliverable *D 1.2 V2 of the VIMpay app* of Work package 2 (WP2) of the VIMpay project.

While the deliverable itself is the second version of the demonstrator app of the VIMpay app, which can be downloaded online (c.f. [1]), this document describes the details of the implementation of the various functionalities and acts as a report on the application.

1 Scope of this Deliverable

This deliverable describes the Functionalities and implementation of them for version 2 of the VIMpay app.

2 Overview of new functionalities

Version 2 of the VIMpay App extends existing functionalities by integrating full KYC and SEPA payment possibilities. There are also some new functions to increase usability and payment safety.

The following table contains the new functions and a short description of them.

Function	**Description**
upgrade to VIMpay Premium	The user is able to upgrade his VIMpay account to get full bank account features.
Push notifications	The user will receive notifications of new transactions for his VIMpay card.
Plastic card order	The user is able to order a plastic card for his VIMpay credit card to be able to pay in shops or withdraw money from ATMs.
full SEPA payment	The user is now able to execute SEPA transactions using his VIMpay card.
withdraw money from ATM	The user is now able to withdraw money from atm. The user needs a plastic card.
trust model	The user is able to choose between several storage and access layers.
VIMpay Valet	VIMpay Valet helps the user to stay up to date on the latest transactions of his bank account. VIMpay Valet will automatically refresh his bank account in the background and analyse the revenues for the user.
Security state	The user is now able to permit each transactions separately using new security states.

3 Architecture overview

The existing libraries are still used, some of them added new features mainly VIMpay App Engine.

A new service is VIMpay Valet. Valet runs on the user device and uses the HBCI Engine to update bank account data. It is triggered by the VIMpay Backend using push notifications

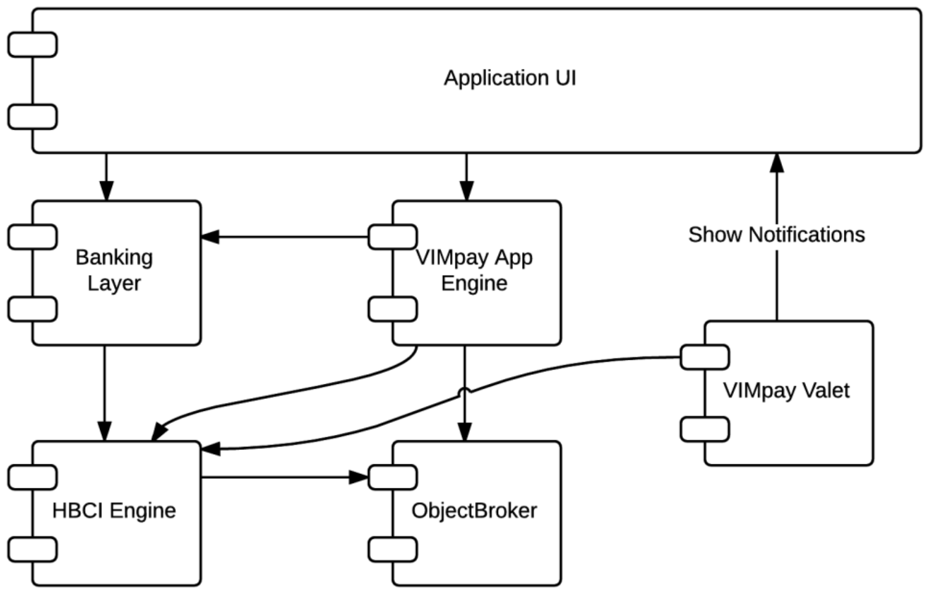

Figure 1 App architecture

4 Use case implementations

4.1 Use case 1: Upgrade to VIMpay Premium

In order to upgrade to VIMpay Premium the user must have a VIMpay account using VIMpay Standard or VIMpay Plus. The user starts the upgrade process in the upgrade screen and will now be asked for required data to proceed. The requirements and fees for this upgrade can be found in [2].

4.2 Use case 2: Push-Notifications

The VIMpay app receives push notifications from the server and shows them to the user. The push notification itself only contains a message type and an ID, so that no sensitive data is transferred over any third party server. The message content will be loaded from the VIMpay Backend and then processed. Some notifications aren't directly displayed to the user because they trigger app internal functions like automatic HBCI account update.

4.3 Use case 3: withdraw money from ATM

The user is able to withdraw money from ATM using his plastic card. The process is secured and the transaction must be permitted by the user using the "ATM"-button in the VIMpay Board. The app unlocks the card for a specific time for the next ATM transaction by sending a request to the VIMpay Backend. After the next ATM transaction, or a timeout, the card will be reset again. The card can also be reset by clicking the "lock"-button in the ATM screen.

4.4 Use case 4: online payment

The user is now able to permit online payments by clicking on the "internet" button in the VIMpay Board. The app then unlocks the card for a specific time for the next internet transaction by sending a request tot the VIMpay Backend. After the next internet transaction, or timeout, the card will be reset again. The card can also be reset by clicking the "lock"-button in the internet screen.

4.5 Use case 5: POS payment

The user is now able to permit shop payments by clicking on the "terminal" button in the VIMpay Board. The app then unlocks the card for a specific time for the next POS transaction by sending a request tot the VIMpay Backend. After the next POS transaction, or timeout, the card will be locked again. The card can also be reset by clicking the "lock"-button in the terminal screen.

4.6 Use case 6: change unlock time

The user is able to change the time while a card is unlocked for next transactions. Those settings can be changed in the security settings screen.

4.7 Use case 7: password reset

The user is now able to reset his VIMpay account password using the app. To start this process the user has to click on the "forgot password" button in the VIMpay login screen. Then the app sends a request to pfREST and receives to safety question of the user. The user has to answer the safety question and set a new password. The backend checks the entered data and will send a approval code via SMS or eMail. The user has to click to link in eMail or submit the approval code in the app. Afterwards he has to login again with new password to complete the process. The Backend process is described in [3].

5 Overview on the implementation of functionalities

5.1 Push notifications

The VIMpay App receives push notifications from Google or Apple using GCM (Google Cloud Messaging) and APNS (Apple Push Notification Service). This messages only contains unique IDs and message types so that no sensitive data is transferred over third party server.

When a push notification is received, the message is processed in different ways because not all notifications are shown directly to the user. Some messages are just wake up events and contains no data. These events trigger HBCI bank account refresh for example and after such events the app still may display the results to the user by showing a notification for "There are new revenues on your account".

In the future some notifications may require an user action like allow a transaction for his VIMpay Card.

The messages which actually contains contents that should be shown to the user are stored in the Prepaid Backend and will be loaded from the open API layer after receiving the push notification.

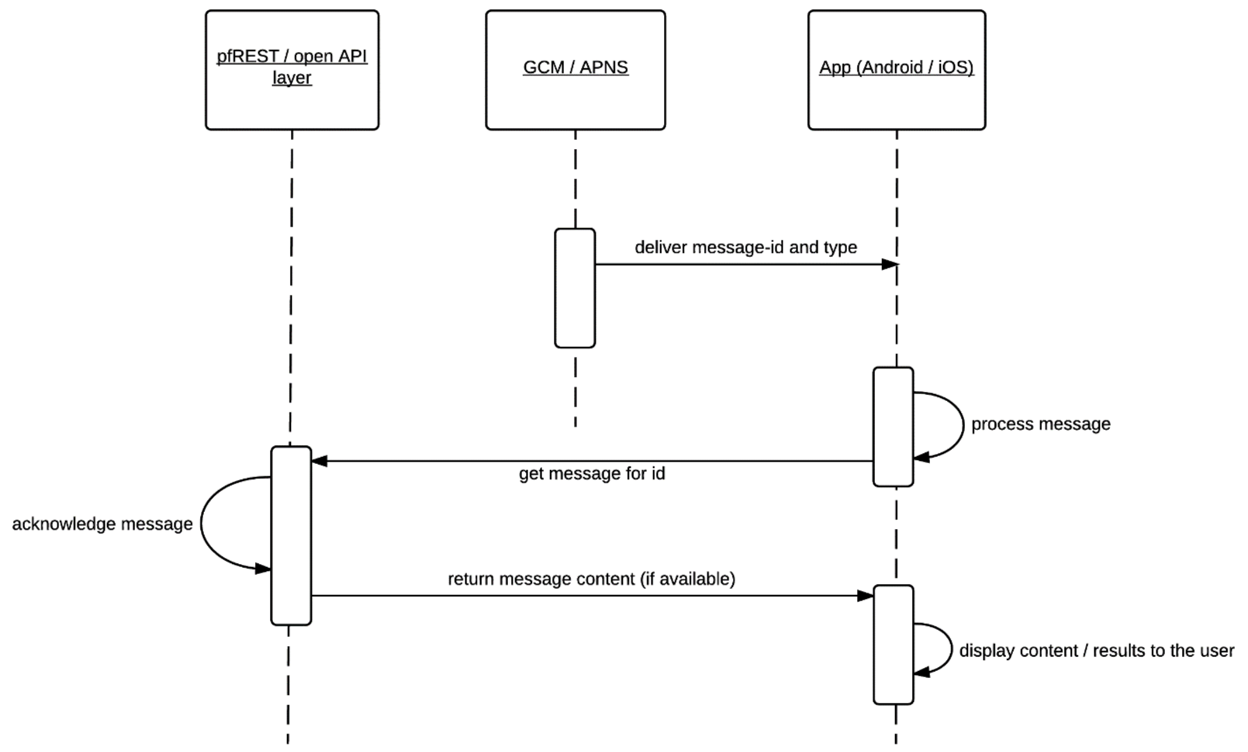

Figure 2 Push Notification app side

5.2 Trust model

The user has now the option to store the private key on a secured petaFuel server. This allows several background features like automatic bank account refreshing. The key will be loaded from the server using the user login credentials.

The trust model is separated into data storage and data access parts. The user will be able to choose between three different stages for the data storage and two different stages for data access. In this version of VIMpay stage 2 for the data access will be added. For data storage only stage 1 is available.

	Storage	**Access**
Stage 1	Data complete stored on the device	Encrypted private key on device
Stage 2	Data stored on the device but synced across several devices	**Encrypted private key on petaFuel server**
Stage 3	Data stored on petaFuel server	

The app basically creates a RSA keypair and upload the plain private key to the Backend. The public key stays on the device and is used to encrypt the symmetric database key. The encrypted symmetric key is stored in the database on the device

In the following diagram the initialization of the Object Broker for stage 1 and stage 2 is described.

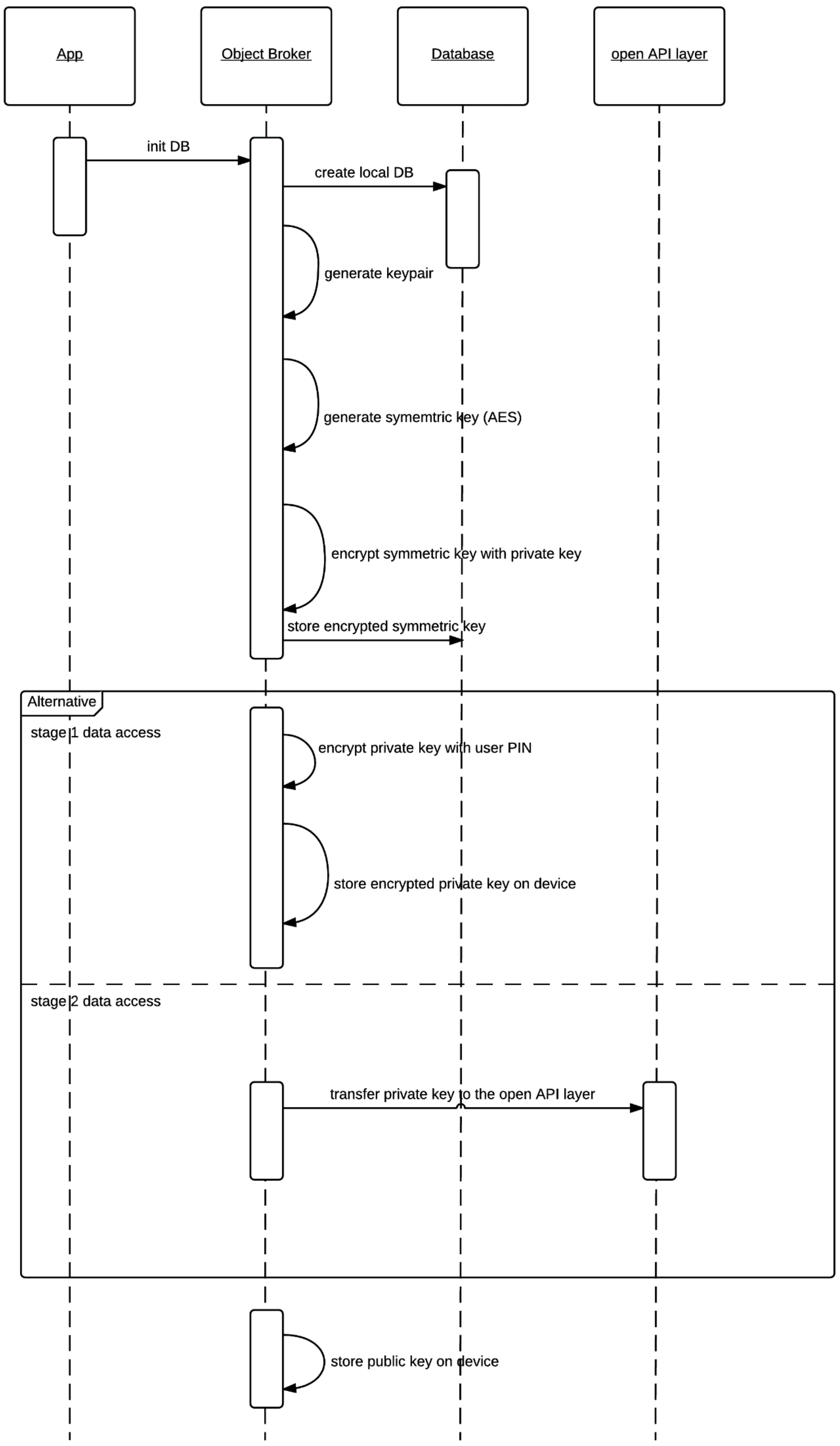
App
Object Broker
Database
open API layer
init DB
create local DB
generate keypair
generate symemtric key (AES)
encrypt symmetric key with private key
store encrypted symmetric key
Alternative
stage 1 data access
encrypt private key with user PIN
store encrypted private key on device
stage 2 data access
transfer private key to the open API layer
store public key on device

Figure 3 Trust model

The following diagram describes the process when encrypted data is accessed. The object broker loads the encrypted symmetric key from the database and decrypt it with the plain private key.

To get the private key to app load it from either the open API layer (pfREST) in case of stage 2 or the device storage in case of stage 1.

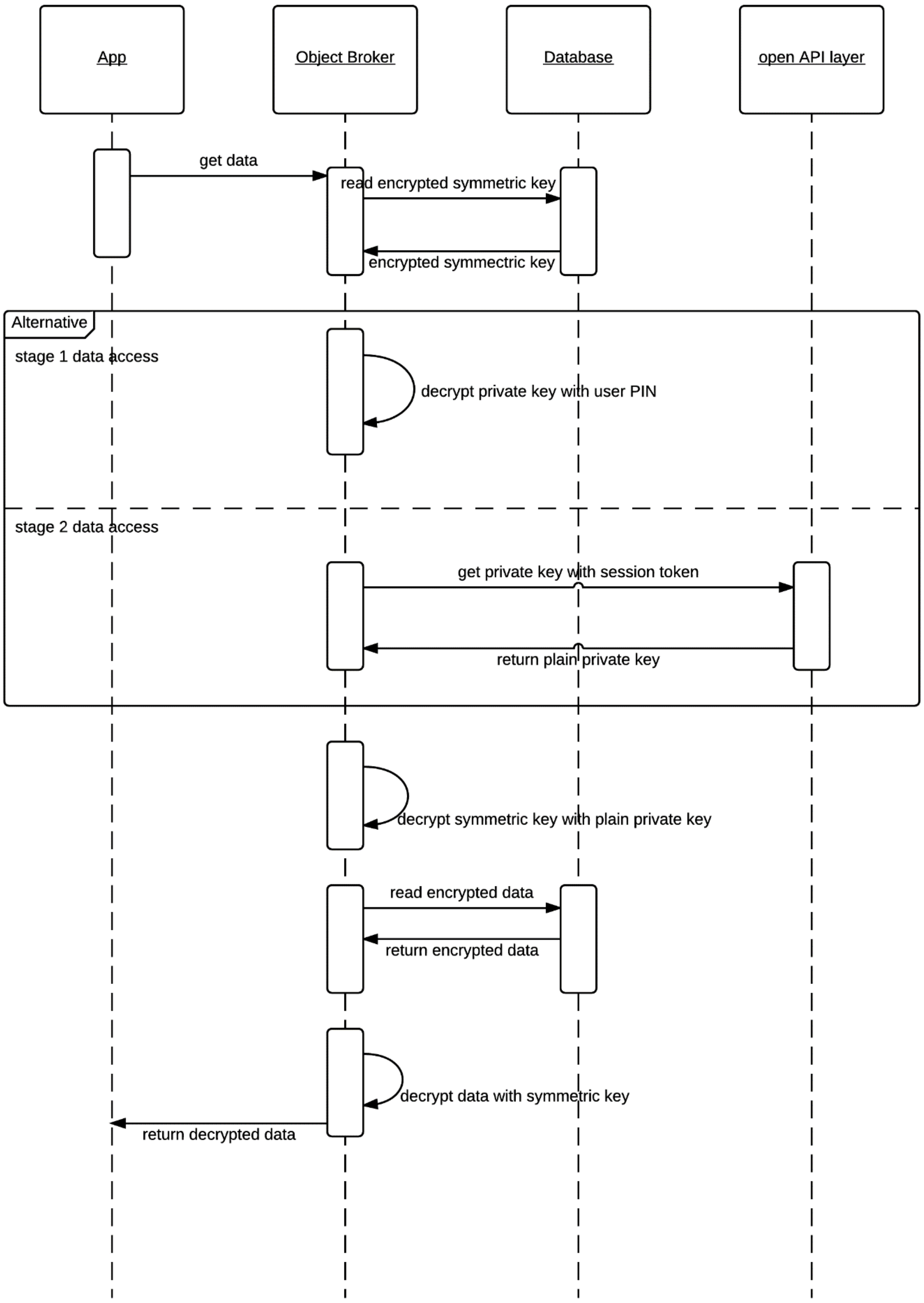

Figure 4 Trust model key exchange

5.3 VIMpay Valet

VIMpay Valet is a new service which helps the user to stay up to date. The Service run on the device itself and processes bank account data. No bank account data of the user is processed on a petaFuel server. The bank account refresh is triggered by a push notification. The app will now update all bank accounts using the HBCI Engine (which is already described in [4]). Afterwards VIMpay will show a notification that new revenues are available.

In order to activate VIMpay Valet the user needs at least a VIMpay Card Basic and have to enable data access stage 2 (see Trust Model), those requirements are shown to the user and he is guided to enable Valet.

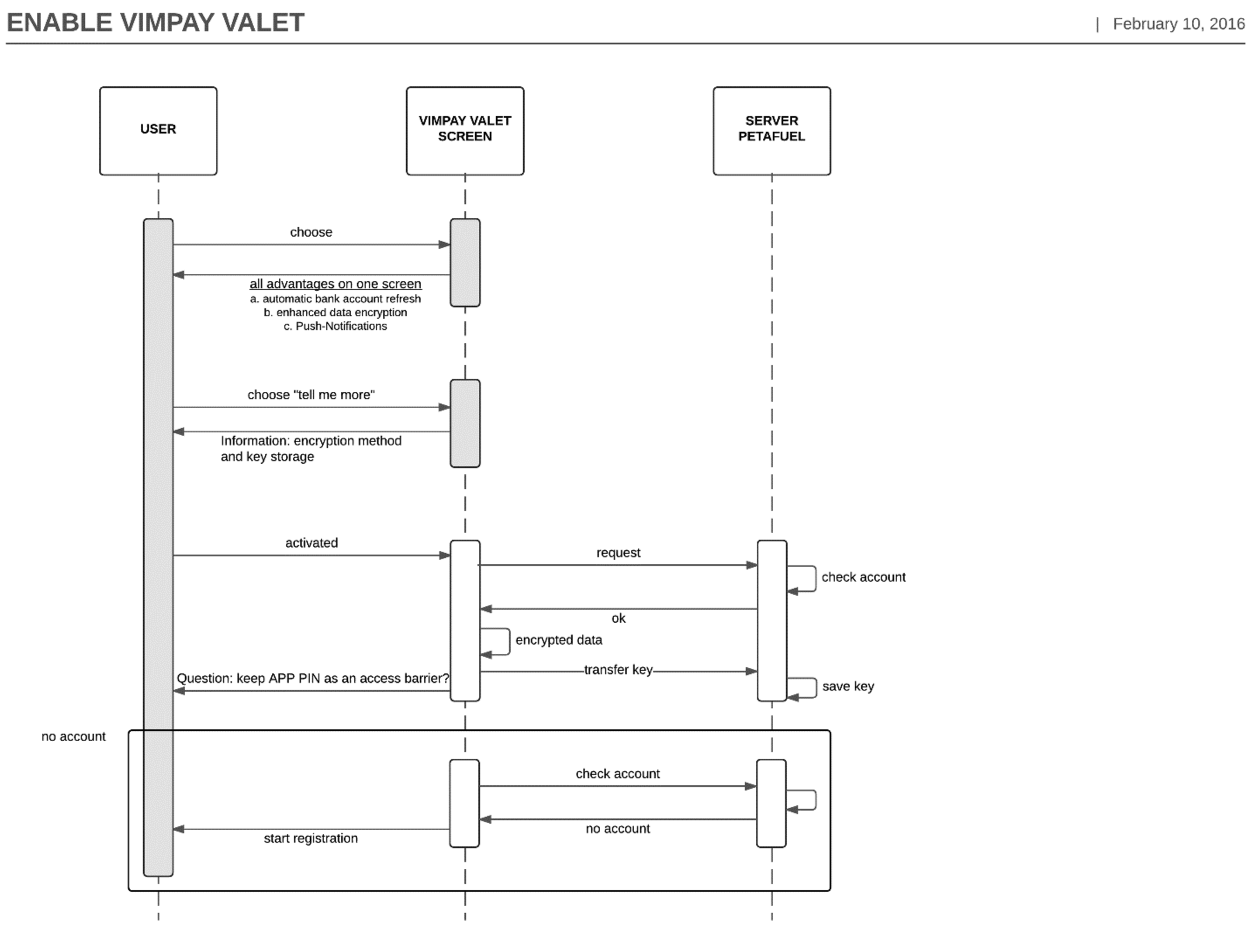

Figure 5 VIMpay valet activation

5.4 Security status

The new Security state provides better control of the users VIMpay card. It manages the limits of the card and enables the ability to permit every transaction just in time. The user unlocks the card before each purchase and afterwards the card will be reset again.

State	**Limits**
Secure	No transactions can be performed without unlocking the card
Standby	Only transactions up to 15€ can be performed without unlocking the card
Open	All transactions are allowed

5.5 Card details

The details of the VIMpay card are supported by the open API layer (pfREST). The app retrieves the data and shows them to the user. In advantage the data, like the CVC or PAN, can now be displayed dynamic. The network traffic is also reduced because the background picture of the card is now stored on the device and it only needed to be loaded once.

For security reasons the card details won't be shown to the user if his device is rooted. The app basically checks three different methods if a device is rooted.

	Description
Check 1	Check the system build tags if it contains test-keys
Check 2	Check if a superuser.apk is present on the device
Check 3	Check if a su binary exists in multiple common directories

6 VIMpay App Engine / pfREST client updates

There are new REST functions added to the open layer API that matches the new requirements.

Method	Function	Access
/securitystatus/reset	reset / lock a card for a transaction	Restricted
/securitystatus/	get the current security state (open, standby, secure)	Restricted
/securitystatus/	set a new security state	Restricted
/securitystatus/timeouts	get the time how long the card is unlocked	Restricted
/securitystatus/unlock	unlock a card for new transaction	Restricted
/card/	retrieve card details (PAN, CVC, cardholder name, expiry date)	Restricted and secured (HMAC)
/key/	get private key	Restricted and secured (HMAC)
/key/	upload an new private key	Restricted and secured (HMAC)

7 Availability

7.1 Android

Version 2 of the VIMpay App is currently available in the development channel on Google Play.

7.2 iOS

On 14.01.2016 the iOS Version V1 of the VIMpay App was published to the Apple App Store. Version 2 is available in the development channel (Testflight) in the Apple App Store.

8 References

[1] petaFuel GmbH, „VIMpay App on Android Playstore (closed beta),“ [Online]. Available: https://play.google.com/store/apps/details?id=net.petafuel.mobile.vimpay&ah=trIxUns4btC3AqPV9UxyCbDyxcs&hl=de.

[2] petaFuel GmbH, „D 5.2 Business requirements for version 2 of the App,“ 2016.

[3] petaFuel GmbH, „D 2.2 Backend Support for Version 2 of the VIMpay App“.

[4] petaFuel GmbH, „D 1.1 Version 1 of the VIMpay App,“ 2015.