Building the Bridge Towards an Open Electronic Wallet on NFC Smartphones

Kevin De Kock, Thierry Van Herck, Glenn Ergeerts, Rud Beyers, Frederik Schrooyen, Marc Ceulemans and Luc Wante
Department of Applied Engineering
Artesis University College of Antwerp
Antwerp, Belgium
glenn.ergeerts@artesis.be

***Abstract*—Many recent initiatives indicate an evolution towards an open electronic wallet to perform all sorts of electronic transactions, like for example micropayments, loyalty, and transport ticketing. Furthermore, the smart phone is emerging as an indispensable tool for many, containing more and more personal information. Hence, it seems like the ideal medium for carrying the electronic wallet as well. The fast and intuitive touch-and-go philosophy and the integration in mobile devices, makes Near Field Communication (NFC) the perfect technology for an electronic wallet. However, the complex ecosystem is holding back the world-wide integration of this technology in mobile handsets, resulting in a low market penetration of NFC smartphones. This paper discusses the use of an active NFC Bluetooth sticker as an intermediate step towards an open electronic wallet on NFC smartphones. Two requirements are set: backwards compatibility with an existing DESFire smart-card solution and support for all the different smart phone platforms. The first requirement will be satisfied through the deployment of a DESFire emulator on the Java Card of the NFC Sticker. The second requirement will be fulfilled by using the Smart Card Web Server functionality of the secure element, which will provide a platform independent interaction with the content of the electronic wallet. Finally, the proposed solution was evaluated in terms of user-friendliness, transaction speed, compliance with the imposed requirements and feasibility of success in the current NFC ecosystem.**

Keywords-NFC; eWallet; NFC Sticker; SCWS; Java Card.

I. INTRODUCTION

The giant leaps of progress in the field of microelectronics since the 1970s have created a solid foundation for the smart card technology of today. An intelligent smart card contains both a microprocessor and data storage, which are integrated on a single silicon chip. This allows cryptographic algorithms to be executed in order to protect sensitive data against tampering and other security threats [1].

This high level security environment has opened the door for applications that resolve around electronic payments (e.g., online/offline debit cards or E-purse cards). However, a lot of the commercial payment systems that exist today are geared towards the general transaction of solely e-money; whereas cases exist that ask for a more specific approach.

The Artesis University College of Antwerp in Belgium has started the Tetra EVENT project [2], which focuses on the development of an open electronic wallet for the event sector. This project will use an online/offline hybrid payment system and several types of items can be stored on the wallet (e.g., vouchers, tickets, coupons, credits, loyalty).

The wallet itself resides on a passive DESFire tag and relies upon terminals equipped with Near Field Communication (NFC) technology to initiate the actual transactions. One of the main advantages in comparison to other existing electronic wallet systems is its inherently scalability aspect in function of big events, because of its online/offline hybrid system. Another advantage is its open character, which allows other 3rd party modules to co-exist on the same physical wallet.

Even though the current market situation imposes the project to focus on NFC tags, a proof of concept will be carried out to test the feasibility of using mobile phones instead. These devices hold a number of intrinsic advantages over tags such as allowing users to view and interact with the contents of the wallet, whereas tags rely solely on terminals instead, which can be deemed to be a shortcoming.

Unfortunately, there is still a low market penetration of NFC enabled mobile phones [3]. A temporary solution to this problem is the use of NFC stickers, which allows any handset to gain NFC functionality through a Bluetooth connection. An extra benefit gained from the use of an NFC sticker is its build-in NFC reader, which allows external DESFire tags to be accessed.

Since current terminals are intended to be compatible with only passive DESFire tags, one of the goals is that the mobile counterpart adopts the current protocols used between terminals and tags. A DESFire emulator will be deployed on a Java Card to ensure backwards compatibility with the currently used system.

Furthermore, there is a lot of differentiation between the various handsets currently available on the market, which means that the wallet implementation needs to be as cross-platform as possible. A Smart Card Web Server (SCWS) servlet [4] will be employed to provide the means of interaction with the contents of the electronic wallet.

Nevertheless, the result of deploying an electronic wallet on a mobile phone could prove to be very useful in the long run. The subsequent sections of this paper will focus on the development of an electronic wallet, which allows the user to interactively work with the contents of the wallet, taking into account the various aspects and hindrances mentioned

 ISBN: 978-1-61208-164-9

before. This paper is structered as follows: the next section goes into details about the research and development done. The third section describes the results and is followed by a concluding section.

II. Research and Development

The research put forth in this paper focuses on carrying out a proof of concept, which will be used as a basis to determine the feasibility of using mobile handsets for the deployment of an open electronic wallet system. The results will be compared with the passive tag version that is currently used in the EVENT project. Furthermore, the project will have to meet certain requirements in order to be accepted as a full-fledged and valid alternative to the use of passive tags. These requirements are the following:

- Finding an intermediate solution to the NFC ecosystem problem, which is currently limiting and/or preventing the further expansion and deployment of NFC services on a larger scale.
- Providing backwards compatibility for the existing passive DESFire Smartcards [5], which are used in the EVENT project for holding the wallet data.
- Allowing the wallet to be used on various different handsets and thus broadening the pool of potential users, by making the system cross platform.
- Taking the aspect of security in account since sensitive data will reside locally on a mobile device.
- Comparing aspects like user friendliness and transaction speed with a passive NFC tag.

A. Project architecture

The software architecture (Figure 1) and hardware used during this project is dictated by the earlier mentioned requirements and can be divided in a number of components. All components are interconnected with each other using either internal or external communication links.

The first component is the interface of the wallet, taking full advantage of the readily available screen and keyboard provided by the mobile device, which is in contrast with the passive tags that rely purely on terminals for the readout of its content.

There is a lot of differentiation however between the mobile devices of different manufacturers (e.g., a variety of different available mobile platforms), which makes creation of a universal interface across platforms currently very hard. A second issue presents itself in terms of secure data storage since the data stored in the memory of the mobile device has some monetary value and the memory itself is intrinsically unsafe (tampering may occur by both the user itself or third parties).

Both the cross platform issue and the secure data storage issue can be solved by the use of a Java Card with a SCWS [6] deployed on it. The Java Card is used as a secure element to prevent any illegal access to the data and an HTML

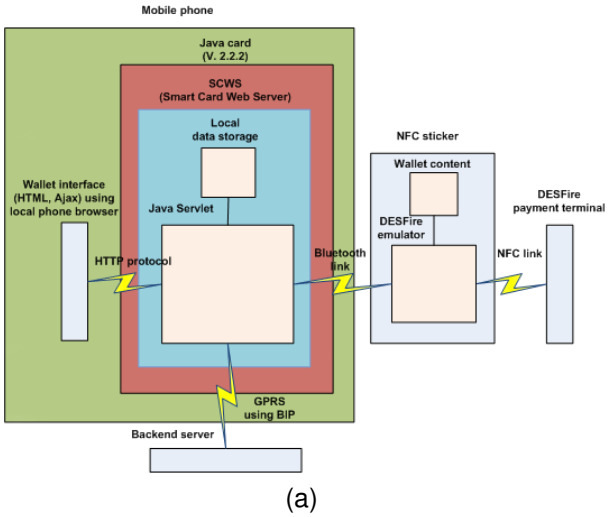


Figure 1. General setup of the system

interface is rendered in the browser of the mobile phone by the SCWS servlet, which will generate the necessary HTML pages. The operation of the Java Card is done by its own local OS and it is thus separated from the general operation of the mobile device OS itself.

The second component of the architecture is the DESFire emulator, which will provide backwards compatibility with the DESFire tags currently used in the EVENT project. This emulator is utilized to link the mobile wallet with the payment terminals through to use of the DESFire protocol. This allows communication links to be setup without having to alter the existing payment terminals.

The third and last component deals with the NFC ecosystem problem, namely the lack of NFC functionality in many currently available mobile devices. We made the decision of using NFC stickers, which are designed specifically for this issue and will provide NFC functionality through a Bluetooth link to a mobile device. The sticker is attached to the back of a mobile phone.

The next sections of this paper will be dedicated towards providing a more in depth discussion regarding the components mentioned before and the actual hardware that has been used.

B. Smart Card & Java Card

The main objective in the use of smartcards is providing the necessary level of security for the storage of sensitive data to an otherwise unsafe environment. This allows applications to be developed around this sensitive data using a subset of the Java programming language [7]. A well-known example is the SIM (Subscriber Identity Module) card, which is used for the identification of a mobile phone user in order to give secure access to the GSM/UMTS network.

 ISBN: 978-1-61208-164-9

Most SIM smartcards are Java Card based these days, which offers the benefit of allowing third-party software to be loaded on the card and executed through a Java Card API.

A high level of security is still maintained on Java Cards by isolating the memory regions of each individual applet from each other through a software firewall, each applet runs in its own context. Furthermore, cryptographic functions can be executed to secure data communication and applets can run in parallel with each other after having been selected by the OS.

However, there are cases where an applet still requires the data and functionality from a foreign applet. A way of acquiring this is through the use of SIO (Shareable Interface Objects) [8][9]. It is important to note that the access mentioned before is only limited in scope. Solely the functions which are defined through the use of one or multiple interfaces are visible outside the applet who grants outside access. Additionally, this access is usually only granted to an outside applet that can identify and authorize itself through the use of its unique AID (Application Identifier).

C. Interface

Mobile handsets offer a very large advantage in terms of user interactivity when compared to passive tags. The former comes equipped with a functional keyboard and screen and grants the possibility of an interactive interface for the mobile electronic wallet. This section will provide the necessary information about the interface part of the project.

The first step in the development of the interface was to compare several types of interfaces with each other and determine which type best suits the basic project requirements mentioned in section II.

The possible interface types are:

- Using an external SATSA MIDlet residing in the memory of the phone itself
- Using SIM Application Toolkit (SAT)
- Using a SCWS servlet residing on the Java Card

Even though SATSA MIDlets [10] offer a lot in terms of the visual interface options and adequate security, these MIDlets lack in terms of the cross platform requirement since they are designed specifically for a J2ME environment. Furthermore, this MIDlet needs to be installed on the handset itself and thus a reinstall is required every time a user decides to switch handsets. Lastly, only very few handsets are currently supporting the JSR 177 API required for using SATSA MIDlets.

The SAT on the other hand offers full interoperability since everything is stored on a SIM card, which is useable by any kind of handset and is furthermore deemed to be very secure. While this may sound tempting to use, the flipside is that it lacks seriously in terms of visual interface options. Moreover, its usability is limited to only SIM cards as SE.

A good compromise is the use of a SCWS installed on a Java Card. A SCWS is a HTTP 1.1 web server embedded on a Java Card and is available since the Java Card 2.2 version, offering a device independent way of the management of personal user data in a secure fashion. This option still puts the application on the SE for security and interoperability purposes. The difference is however that a relatively good HTML interface can be offered by providing static and dynamic content to the browser of a mobile phone through the http://127.0.0.1:3516/ address, which is OS independant.

The last interface type seems to be the most beneficial in terms of the electronic wallet project, because of its advantages in both maintaining a good visual interface as well as the interoperability, security and user friendliness aspects.

The application and data can be additionally managed externally through an Over The Air (OTA) link using web protocols. A secure tunnel will be opened between the SCWS on the Java Card and the OTA platform [11], which is used for the administration of the SCWS.

D. NFC Bluetooth sticker

NFC stickers are contactless cards/tags designed to be glued on the back of a mobile phone and are designed to offer a solution to the current complex NFC ecosystem that prevents a world-wide integration of NFC technology in mobile handsets. A ferrite backing layer prevents distortion to occur between the components of the phone and its radio signal. The sticker also has an internal antenna installed for communication purposes.

An alternative to this is the use of MicroSD cards, which have an embedded chip that will grant NFC functionality to the host device. The antenna itself can be either external or integrated in the package. Additionally, MicroSD cards use similar read/write functions as integrated NFC handsets.

Both have their advantages and disadvantages. An NFC MicroSD card is basically plug and play, thus eliminating difficult setups, but requires the handset to have a MicroSD slot. The NFC sticker on the other hand only needs a Bluetooth radio, which is a very common feature in most of the mobile phones currently produced. The drawback is however that these are less user friendly to setup compared to NFC MicroSD cards.

After carefully weighing both options, we decided to integrate a MyMax NFC sticker from Twinlinx in our project. This will make the project potentially compatible on a much larger variety of mobile handsets compared to the small number of handsets currently available that support a SE on a MicroSD card.

1) NFC sticker characteristics: The sticker has been designed to be as small as possible. A small low voltage battery is used to power the internal Bluetooth chip. This chip is responsible for setting up a connection with the handset and has the capability of making between 300 and

 ISBN: 978-1-61208-164-9

500 connections before the battery is drained. The battery itself can be wirelessly recharged using a specialized USB-charger.

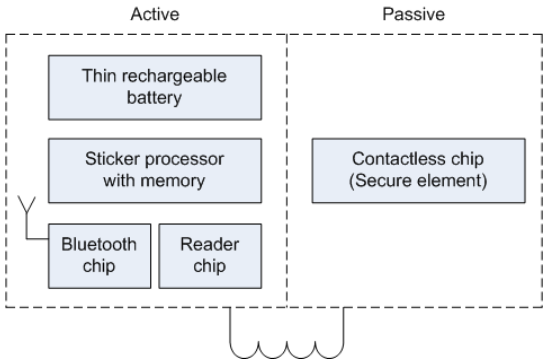


Figure 2. MyMax NFC sticker architecture

The MyMax sticker can act either passively or actively (Figure 2). An active sticker will rely on its own internal power source, while a passive sticker will use an emitted magnetic field from an external reader to draw power from.

Furthermore, the sticker can go into 3 different operation modes; the first mode is a passive mode where the sticker will act as a passive NFC tag. An external reader can be used to read and/or alter the contents of the NFC chip/internal SE on the sticker. It is important to note that a sticker operating in this mode will work completely independent from the handset that it is attached to (i.e., the handset is not required to be powered).

The second mode requires that both the MyMax sticker and its corresponding mobile phone draw power actively from an internal power source. This mode allows a connection to be established between the sticker and the mobile phone. Consequently, the content of the internal SE/NFC chip can be read or changed by the handset through this link.

The third mode takes advantage of the internal reader chip of the sticker, which makes it possible to create a connection to an external tag and allow the mobile phone to read or change the contents of this tag.

2) Testing the capabilities of the sticker: We carried out some preliminary tests with the NFC sticker to determine several key aspects of the sticker that are necessary in the development of the project.

First, we wanted to determine whether the active part and the passive part of the sticker share the same SE. In order to use the active part, we installed the MyMax demo projects on the handset to write new values to the internal NFC tag of the sticker. These values are readable by an external reader, thus proving that the active and passive part of the sticker share the same SE since the reader only reads the passive part.

The second test involved the use of the MyMax SDK to develop a J2ME test project Midlet. The purpose of this project is to read the values of an external tag through the reader chip of the MyMax sticker. DESFire APDU commands are transmitted to the external tag using functions found in the MyMax Library to send APDU.

3) Combining the sticker with the Java Card: Setting up a secure Bluetooth link between the Java Card residing on the handset and the external SE on the sticker is absolutely vital to safeguard the read/write keys that are used in the application for accessing and altering the wallet information from unauthorized use. The purpose of this is to prevent any counterfeiting from occurring.

It is important to note however that the JSR 82 API for Bluetooth is not supported in general by the currently available Java Cards. This obligated us to look for alternative ways to gain this functionality, since the sticker as mentioned before uses Bluetooth technology when accessed from a handset and it is imperative that the overall security of the system is kept at a high level.

A first potential solution for this issue is the use of SATSA (Security And Trust Services APIs) Midlets [12], which are designed specifically for the secure access of Java Card applets. The availability of the JSR 82 API for SATSA Midlets allows data to be passed between the Java Card and NFC sticker in a more indirect fashion.

Two different communication APIs are possible with SATSA Midlets. The first communication API is the SATSA JCRMI (Java Card Remote Method Invocation), which requires a Java Card applet to first extend the java.rmi.Remote interface before any data can be shared with an external application [13][14]. The second communication API is the SATSA APDU, which uses APDU messages to access the on-card objects.

The flipside however in the use of SATSA Midlets is losing the cross platform aspect of the project since SATSA Midlets are J2ME specific, thus narrowing down the number of compatible handsets. Additionally, it turns out that almost no mobile phones support SATSA JCRMI in the first place and only a small number support the SATSA APDU API.

Despite the fact that SATSA Midlets gave good results in terms of programming functionality, they are not viable to be used in our project due to lack of support.

A second potential solution to gain Bluetooth functionality is the use of the BIP (Bearer Independent Protocol) that allows OTA support for the Java Card. While it is theoretically possible to setup a Bluetooth connection using this protocol with the sticker, the lack of publically available practical information for development purposes has prevented us from actually implementing this functionality in our project [15]. The addition of this functionality has been scheduled for future work.

 ISBN: 978-1-61208-164-9

E. DESFire emulator

One of the goals of the project is to maintain backwards compatibility for the existing passive DESFire Smartcards that are used during the EVENT project for holding the wallet data.

The MyMax sticker uses a mifare 1K Classic chip with 1KB of memory, which is a lot smaller compared to its DESFire counterpart that can hold up to 8 KB. The latter can thus hold a lot more wallet data and also uses certain functionality and encryption algorithms (3DES) that the sticker lacks. This results into an inability to port the passive DESFire Smartcard wallet directly to the sticker.

A way to overcome this problem is the deployment of a DESFire emulator on the SE of the sticker to emulate the DESFire wallet functionality. Some additional advantages are its larger sticker SE memory (32KB EEPROM) and it can also be deployed on NFC phones or NFC MicroSD cards.

We will be using a NFC MicroSD card instead of the sticker for testing purposes during this project because of the problems with initiating a secure Bluetooth link from the SE as mentioned earlier.

III. RESULTS

This section will provide some deeper insight regarding the inner workings of the application. This application (Figure 3) exists out of an interface and two Java Card applets, which are responsible for providing the actual wallet data.

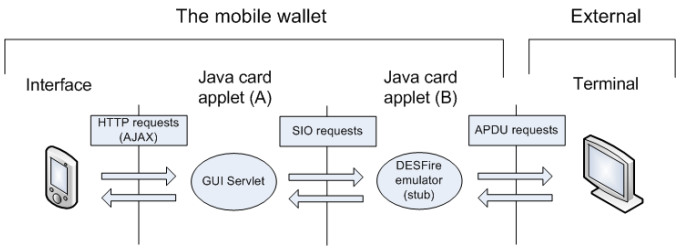


Figure 3. Application components

The content of the event wallet is displayed in the interface on the mobile phone of the user. This HTML based interface can be divided into a number of interface components that are individually requested from a SCWS residing on the Java Card through various HTTP requests.

The servlet that is responsible for providing the interface with the necessary data is running on the SCWS and will send specific SIO requests to the DESFire emulator on the Java Card. This emulator is another applet residing on the Java Card and will (after authentication) provide the servlet with the desired data. The servlet will then, based upon the collected data, give an answer to the previously made request by the interface.

Updates to the wallet content on the emulated DESFire card are done mainly through external points of sale, which are available on either the event itself or through an OTA purchase.

A. Application interface

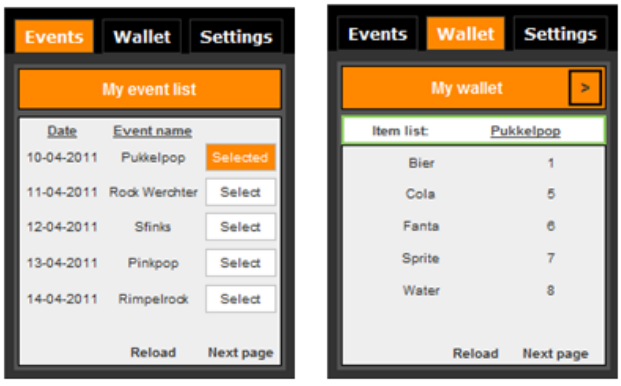
The design of a mobile interface requires some special attention, more so than its workstation counterpart.

First, there is a large differentiation between different cell platforms, each whom presents its own interface. Secondly, physical obstacles such as different screen sizes, aspect ratios and physical buttons need to be taken into account as well. Lastly, there is the user aspect, which demands that an interface needs to be intuitive and easy to learn.

We made the decision to use a SCWS to provide a consistent interface by taking full advantage of the mobile phone browser capabilities, which is tasked for the rendering of an HTML based interface for the wallet application. This interface is theoretically universally applicable to any mobile phone that supports Java Cards. Recent developments like jQuery Mobile and PhoneGap make it possible to build native looking and responsive HTML and javascript-based applications.

The design of the interface of the application itself is based upon known design principles [16], which dictate how information on a page is to be presented to its user. This includes but is not limited to bringing information to the top of the interface by limiting the amount of links a user has to go through thus, minimizing navigation or bringing a collection of relevant information together, based on the desired intent of the user.

1) Interface structure: The wallet interface consists of three main tabs that allow a user to browse through a number of subtabs. The main interface tabs are the ”Events”, ”Wallet” and ”Settings” tab.



(a) The event tab (b) The wallet tab

Figure 4. The interface

The ”My event list” subtab (Figure 4(a)) is used to list all the possible events that a user currently has access to. This

list is kept up to date through a connection with a backend server. A number of possible items are linked to each specific event and are shown in the ”My wallet” subtab (Figure 4(b)). Items can be bought and/or spend using the local terminals or through OTA functionality [11].

The settings tab allows for various options, including setting the number of listed items and/or events on the events and wallet tab pages. The interface is resolution independent and can thus be used on a number of different handsets. The layout can additionally be changed altogether depending on the preferred style of the user.

Finally, the wallet tab includes the option of purchasing items OTA. The purchased items will be listed on the ”My wallet” subtab as ”reserved items”, meaning that they still need to be synchronized by specialized terminals called ”sync points”. The main advantage of this is that wait time for the user at a terminal will be cut down significantly since a user only has to touch the terminal and through a NFC link will the previously purchased items be made available for use.

B. Data flow

The next part of the application consists of two elements, namely the GUI Servlet and the DESFire emulator, which are both tasked with the provision of the actual data to the interface mentioned earlier.

A GUI Servlet is a Java Card application that runs on the SCWS and will act both as a server and a client when data is requested by the interface, since it will serve information to the interface after it has requested the necessary data from the DESFire emulator.

The DESFire emulator is also a Java Card application and consists of an emulated DESFire card with additional communication logic. This emulated card follows the same wallet structure of the passive tags used in the Tetra EVENT project, which is responsible for holding the wallet content. The additional communication logic of the DESFire emulator allows a mobile phone to communicate with the terminals that are used for monetary transactions during an event.

As a result of our project still being in a relative early stage, we have opted to replace the DESFire emulator with a temporary stub in order to fabricate a working prototype. This stub will hold the same functionality from the point of view of the servlet compared to the actual DESFire emulator.

1) Retrieval of data: A backend server forms the backbone of the system, because all the event dates, event names and item names are requested from this server. Providing a page in the interface with values requires the servlet to request a list of AIDs and FIDs first from the DESFire emulator.

These IDs represent the various events and items tied to an event respectively and are required to be translated by the back-end server to their actual corresponding names. The item values shown on the wallet page are collected by using an AID to select a specific event on the DESFire emulator and then request the value of a specific item of that event using its FID.

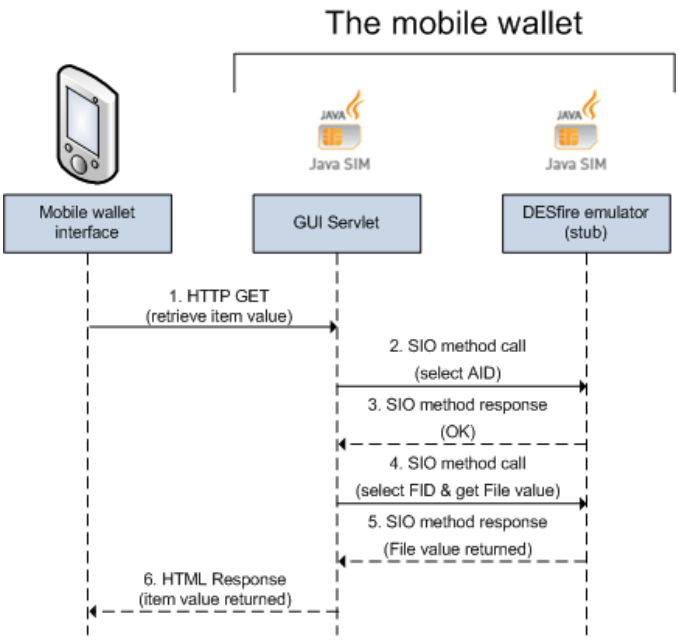


Figure 5. Retrieval of an item value

Every request done by the interface will thus trigger a series of steps in the background of the application (Figure 5). The GUI servlet will perform a SIO method call to select the AID that is linked to the desired event. The emulator will respond with a confirmation message and the actual file value will then be requested next by using the FID of a specific item. The File value is passed back to the servlet, which will pass it to the interface for visualization.

The backend server and its data have been temporary replaced in our project by a number of vectors that store the different names and dates mentioned before in a hardcoded manner. The reason for this is identical to the stub in that it is caused by the fact that the EVENT project is still in a relatively early stage.

C. Installation on a Java Card

This section describes the installation part of the application on an actual Java Card and any problems that occurred in the process [17].

The first step of the deployment on a Java Card is the conversion of the class files of the project, along with any additional required export files, into an executable binary CAP-file (Converted Applet File). This type of file format is designed specifically for Java Cards and is used by an on-chip installer to install the applet and link it with the classes that are already available on the card [18].

 ISBN: 978-1-61208-164-9

1) Used hardware and installation problems: In terms of hardware we used a G&D Sm@rtCafe Expert 5.0 microSD Java Card in combination with the Gemalto Developers Suite for the development of the project. The assumption was made that our project, which is developed for a Java Card version 2.2.2, would run on any physical 2.2.2 Java Card without problems, but this original assumption turned out to be false.

Different vendors use different implementations on their Java Cards [19], which resulted in our case in the use of libraries that are specifically made for Gemalto Java Cards and thus are unavailable on the G&D card that we are using. It should still be theoretically possible to deploy an applet in a cross platform manner by following only the actual global standards put forth by the Global Platform [20].

In order to test this hypothesis, we decided to rewrite the DESFire emulator stub using the Java Card Development Kit V2.2.2 from Sun and were able to successfully install it on the G&D Java Card, thus proving our earlier made point.

IV. Conclusions

The goal of this research was to determine how the functionality of a contactless smart card wallet on a mobile device can be incorporated and improved; using an intermediate step towards NFC enabled devices and taking aspects such as security, usability, backwards compatibility and interoperability into account.

By using a Java Card, a high level of security for safeguarding the sensitive data residing in its memory is maintained. The combination of a Java Card and the SCWS allows the wallet to be deployed on a wide range of mobile devices since no application is required to be installed on the device itself.

The structure of the interface is based upon known design principles to create an intuitive interface for the end-user. Furthermore, the interface has been designed to take aspects such as scalability into account, which is important for overcoming physical obstacles such as different screen sizes of the mobile device.

The current price for a MyMax sticker is 20 EUR, but its price is expected to drop relatively fast, to around 10 EUR, as a result of mass production. The length of the battery life is long enough for it to be used on daily basis. Moreover, it is easy to pair the sticker with a mobile phone.

These factors in combination with our research have proven the MyMax sticker to be a good intermediate solution in terms of the NFC ecosystem problem. However, we were unable to setup a working Bluetooth connection between the Java Card of the mobile handset and the sticker due to the lack of publically available BIP documentation.

The backwards compatibility requirement of the project can be fulfilled through the deployment of a DESFire emulator on the MyMax sticker. Since a working emulator was unavailable during the research phase, we developed a DESFire emulator stub to offer temporary functionality to prove the backwards compatibility of the mobile wallet by installing the stub on the SE of the MyMax sticker.

Our conclusion is that a contactless smart card wallet on a mobile device can be developed, in spite of the current NFC eco system problems, while also taking the fundamental requirements of the project into account.

References

[1] W. Rankl and W. Effing, *Smart Card handbook*. Wiley, 2010.

[2] elab, “Tetra EVENT project,” [accessed 19-September-2011]. [Online]. Available: http://event.e-lab.be

[3] N. Forum, “Whitepaper: Essentials for successful nfc mobile ecosystems,” [accessed 10-July-2011]. [Online]. Available: http://www.nfc-forum.org/resources/white_papers/NFC_Forum_Mobile_NFC_Ecosystem_White_Paper.pdf

[4] SIMalliance, “Whitepaper: Smart card web server, how to bring operators’ applications and services to the mass market,” [accessed 10-July-2011]. [Online]. Available: http://www.simalliance.org/en?t=/documentManager/sfdoc.file.supply&e=UTF-8&i=1185787014303&l=0&s=QcEgCcAUIYrk9KQfX&fileID=1234200160560

[5] B. F. Council, “Mifare desfire specification,” p. 20, 2009.

[6] SIMalliance, “Smart card web server stepping stones,” [accessed 10-July-2011]. [Online]. Available: http://simalliance.org/en?t=/documentManager/sfdoc.file.supply&e=UTF-8&i=1185787014303&l=0&s=K5Aqx7C9UCsQoShk&fileID=/en?t=/documentManager/sfdoc.file.supply&e=UTF-8&i=1185787014303&l=0&s=K5Aqx7C9UCsQoShk&fileID=1261058498628

[7] S. M. Inc, “Java card applet developer’s guide,” [accessed 10-July-2011]. [Online]. Available: http://www.oracle.com/technetwork/java/javacard/overview/index.html

[8] M. Montgomery and K. Krishna, “Secure object sharing in java card,” p. 10, 1999.

[9] D. Perovich, L. Rodriguez, and M. Varela, “A simple methodology for secure object sharing,” p. 7, 2000.

[10] gemplus, “Whitepaper: Integrating the sim card into j2me as a security element,” [accessed 10-July-2011]. [Online]. Available: http://whitepapers.zdnet.com/abstract.aspx?docid=175614

[11] T. C. Vilarinho, “Trusted secure service design: Enhancing trust with the future sim-cards,” p. 167, 2009.

[12] K. Mayes and K. Markantonakis, “Smart cards, tokens, security and applications,” p. 416.

[13] S. Chaumette, A. Karray, and D. Sauveron, “Secure collaborative and distributed services in the java card grid platform,” p. 8, 2006.

[14] J. Andronicj and Q.-H. Nguyen, “Certifying an embedded remote method invocation protocol,” p. 8, 2008.

 ISBN: 978-1-61208-164-9

[15] N. Aini, “Joomla authentication using smart card web server,” p. 48, 2008.

[16] K. Holtzblatt, “Customer-centered design for mobile applications,” p. 11, 2005.

[17] Z. Chen, *Technology for Smart Cards: Architecture and Programmer’s Guide*. Prentice Hall, 2000.

[18] Gemalto, “Java card & stk applet development guidelines.” [Online]. Available: http://developer.gemalto.com/fileadmin/contrib/downloads/pdf/Java_Card_STK_Applet_Development_Guidelines.pdf

[19] J.-F. Dhem and N. Feyt, “Hardware and software symbiosis helps smart card evolution,” p. 19, 2001.

[20] GlobalPlatform, “Globalplatform card specification 2.1.1,” [accessed 10-July-2011]. [Online]. Available: http://www.globalplatform.org/specificationscard.asp

 ISBN: 978-1-61208-164-9