Implementation of Smartcard Personalization Software

Hafid Mammass and Fattehallah Ghadi

MMS, Faculty of Sciences, Ibn Zohr University, Agadir - Morocco
hmammass@neuf.fr, f.ghadi@uiz.ac.ma

Abstract

We treat in this paper the implementation of a graphic tool which allows to personalize smart card (JavaCard cards), which use the RSA (Rivest Shamir Adleman) algorithm and ECDSA (Elliptic Curve Digital Algorithm) for the authentication of the card by the personalization tools and by the authorization server of the card issuer and transforming them as bank cards and securing their use during the transactions of payment, credit, withdrawal, load of the electronic purse or the payment by PME (electronic purse).

This personalization software secures the creation of a bank account by a strong authentication and realize the creation and the personalization of a bank card, the visualization of the data of the card, the consultation of the account, the putting an account or a card in opposition, the canceling of the last opposition of an account or a card, the blocking and the unblocking of a card, the blocking and the unblocking of an application as well as the consultation of the online and offline financial records and the commands sent to a card.

Keywords: *RSA, ECDSA, Monetic, Cryptography*

1. Introduction

The realization of an electronic transaction requires computing, electronic and telecommunications tools given to the cardholder and all the participants.

Our tool is a prototype of application which groups together several features , it plays the role of personalization tool, acquisition tool by performing the features offered usually by electronic payment terminal (TPE), the role of automated teller machines (ATM) and securing the transactions of payment.

The bank Cards serve to store the secret data of the cardholder and the issuer (public and private key of the card and the public key of the issuer) and are able of ciphering data and generating a RSA or ECDSA signature.

With our application, we can prove that in a few steps a blank smart card which costs 24 Euro becomes a card offering multiple banking services (payment, credit, withdrawal) and (load and use of electronic purse) with mechanisms of strong authentication so allowing to realize transactions safely and offering a level of very affordable personalization and a very high security level.

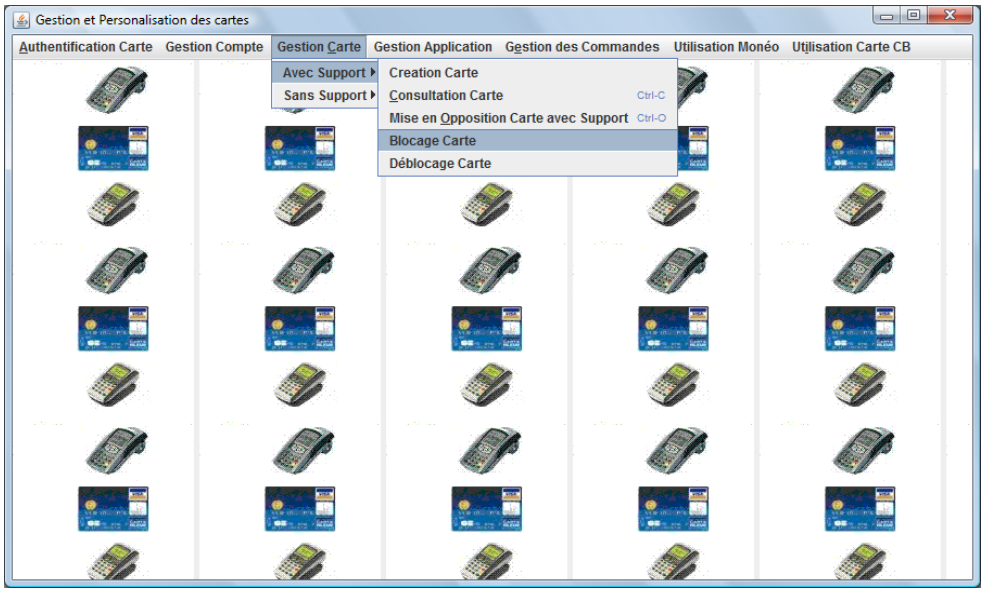

Figure 1. User Interface Overview

2. Strong Authentication

Shamir introduced in 1984 an Identity Based Signature Schemes because the prime factorization of a big integer is a hard mathematical problem and needs an infinite computing power.

2.1. RSA Signature

The smart card stores the public and private key (n,e,d) and sends the public key to the issuer with

$N = pq$, p et q are primes and N is very big (1)

$\phi = (p-1)(q-1)$ and $ed = 1 \bmod \phi$ (2)

Ciphering message $y = x^e \bmod N$ (3)

Unciphering $x = y^d \bmod N$ (4)

They tools receive the RSA Modulus (n) and the RSA exponent (e) by sending two APDU to the card.

Table 1. *request of modulus*

CLA	INS	P1	P2	LE
0x90	GET_MODULUS	0x00	0x00	Lg

The smart card sends the response:

Table 2. *reception of modulus*

DATA	SW1	SW2
MODULE (N)	90	00

The client sends the command:

Table 3. *request of exponent*

CLA	INS	P1	P2	LE
0x90	GET_EXPONENT	0x00	0x01	Lg

The smart card sends the response:

Table 4. *reception of exponent*

DATA	SW1	SW2
EXPONENT (E)	90	00

The tools send an APDU for generating an RSAWithSHA1 signature on a message generated randomly.

Table 5. *request of signature*

CLA	INS	P1	P2	LC	DATA
0x90	RSA_SIGN	0x00	0x00	lg	message

During the generation of the signature for a message m sent by the card, the card calculates an SHA(m) and sign on the SHA(m) with the private key of the card (In the JavaCard applet, we choose the algorithm ALG_RSA_SHA_PKCS1)

The smartcard send the following response APDU:

Table 6. *reception of the signature*

DATA	SW1	SW2
RSAWithSHA Signature	90	00

2.2. Verification of RSA Signature

The tool realizes verification of the signature by initializing a public key object with the public key of the card (modulus and exponent) and by using an RSAWithSHA1's signature object initialized in verification mode with the public key object and verifies that the couple (message, signature) is valid avoiding the attacks Man-In-the-Middle and absent cards.

2.3. ECDSA Authentication

RSA had been the mainstay of PKC for over a quarter-century. ECC, however, is emerging as a replacement in some environments because it provides similar levels of security compared to RSA but with significantly reduced key sizes. NIST use the following table to demonstrate the key size relationship between ECC and RSA, and the appropriate choice of AES key size:

Table 7. ECC and RSA Key Comparison

ECC Key Size	RSA Key Size	Key-Size Ratio	AES Key Size
163	1,024	1:6	n/a
256	3,072	1:12	128
384	7,680	1:20	192
512	15,360	1:30	256
Key sizes in bits.		*Source: Certicom, NIST*	

Some studies have found that ECC is faster than RSA for signing and decryption, but slower for signature verification and encryption.

We use in the second option an authentication with a schema of ECDSA (Elliptic Curve Digital Signature Algorithm) signature where the smart card stores its public key and its private key and generates a signature ECDSA at the request of the application.

The card stores the parameters of the elliptic equation:

$$y^2=x^3 + ax +b \quad (5)$$

And we work on a finished additive group which corresponds to all the points of the curve.

The smart card so parameterized can generate signatures ECDSA which the tool verifies by using to the public key published by the card.

3. Smart Card Personalization

3.1. JavaCard Applet

The first step in the pre-personalization of a smart card is to install an JavaCard applet which manages the various services offered by the card among which the generation of the RSA signature, the control of the pin code, the treatment of the Offline transactions, the load of electronic purse, the payment by electronic purse and the displaying of the balance of the electronic purse.

This Applet is coded in JavaCard and it is tuned to listening the APDU sent by the client application.

3.2. Account Creation

After authentication of the card by the personalization tool, the creation of an account can take place by entering the agency code, the name, the first name, the date of birth, the number of ID card, the address, the zip code, the city, the options of payment of the account, the amount of the useful overdraft, the Authorized maximum overdraft, the balance of the account and the data of the spouse.

The numbering of the account is realized automatically with an Oracle sequence and begins with the agency code.

The account number is registered into the card and after we can personalize the card and linking the card to the account.

3.3. Card Creation and Personalization

After authentication by the personalization tool and the creation of a bank account, a creation and a customization of a bank card can take place.

After the validation of the account number, the agent chooses the first 6 digits of the number of card among a predefined prefixes list and the tool concatenate the prefix with the first 12 digits of the bank account and propose ten cards number by varying the 19th digit from 0 to 9.

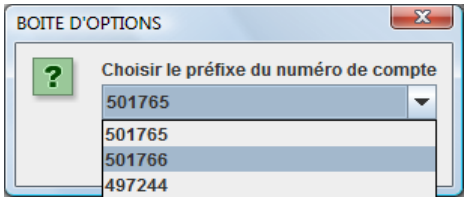

Figure 2. ***Choice of prefix***

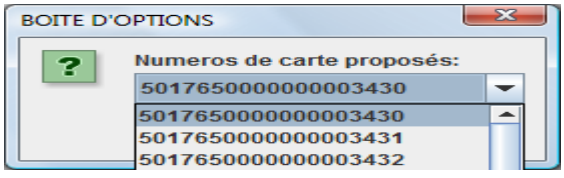

Figure 3. ***Choice of card number***

When the card number was chosen, the user has to choose the type of the support (holder, spouse or shared, the right of withdraw), the validation code of the card and the personal

identification number of the card.

The personalization tool sends two APDU to smart card to initialize the card number and the expiry date so the card becomes a bank card and knows its number, its expiry date and its account number.

Table 8. *setting the card number*

CLA	INS	P1	P2	LC	DATA
0x90	SET_NUMBER	0x00	0x00	0x13	Number

Table 9. *setting the expiry date*

CLA	INS	P1	P2	LC	DATA
0x90	SET_DATEEXP	0x00	0x00	0x08	DATE

The tool calculates an imprint SHA on personal identification number and inserts the database record of the card.

The banking account number stored in the database record of the card makes the link between the bank card and the bank account.

3.4. The Link between the Smart Card and Bank Account

The link between a bank card and a bank account is made by the account number used in the personalization of the card and which is stored in the database record of the card.

An account can contain several cards as the case of a card of the holder and of a card spouse or of a credit card and a withdrawal card associated to the same account.

4. Services Offered By The Smart Card

In the stages of every transaction of payment, credit, withdrawal, load of electronic purse, payment by electronic purse, consultation of its balance as well as all the services proposed by the application, the card is authenticated and the identity of the cardholder (card number and expiry date) are sent by the card at the beginning of every transaction.

4.1. Offline Transaction

After RSA or ECDSA authentication, the Offline transactions are authorized by the smart card if the daily number of Offline transactions is not exceeded and if the daily accumulative amount of the Offline transactions is not exceeded.

Thresholds are parameterized in the smart card via the JavaCard applet installed at the beginning of the personalization of the card.

If one of the thresholds is not respected, the transaction is rejected by the smart card then by the application.

In Order to avoid the Man-in-the-Middle attacks, the smart card calculates a RSA signature on the data of the transaction (the amount of the transaction and the timestamp of the transaction) which is verified by the software by using to the public key of the card.

In case of success, a record is generated in the table of the Offline records.

The Offline records are collected and compensations are sent to the issuer for update of the accounts and for the regulation.

4.2. Payment and Credit by Card

After RSA or ECDSA authentication, the transactions of payment or credit by card are authorized in case the card is not out-of-date, is not blocked, is not in opposition for a given motive, is valid and within the limits of the balance of the bank account for payments, in case of success, the balance is updated and a record is generated in the table of the on-line records.

4.3. Withdrawal by Card

After RSA or ECDSA authentication or ECDSA, the withdrawal transactions by card are authorized after entering and validation of the personal identification number and in the same conditions as the payments.

The goal is to transform a smart card into multi-services card with a very high level of security.

4.4. Account Visualization

After authentication, the application receives the card number and the expiry date and read the database record of the card to get the account number and then read the database record of the account so the data of the bank account are displayed via the menu and without any interactivity with the user.

The options of payment, the balance and the daily accumulative amount are displayed.

4.5. Card Visualization

After authentication, the data of the card are displayed effortlessly additional of the user, the software receives the card number and the expiry date and read the database record of the card and show the data of the card, the right for the dab, the motive opposition, date and time opposition, validation and the flag “blocked card “, etc.

The tool sends an APDU

Table 10. *asking the card number*

CLA	INS	P1	P2	LE
0x90	GET_NUMCARTE	Ox00	0x00	0x13

The smart card sends the response:

Table 11. *reception of the card number*

DATA	SW1	SW2
Card Number	0x00	0x00

The tool sends an APDU to get the expiry date

Table 12. *asking the expiry date*

CLA	INS	P1	P2	LE
0x90	GET_DATEEXPIR	Ox00	0x00	0x08

The smart card sends the response

Table 13. *reception of the expiry date*

DATA	SW1	SW2
Expiry Date(AAAAMMJJ)	0x00	0x00

When the data of the card are got back, the tool read the database record of the card and shows all the information of the card.

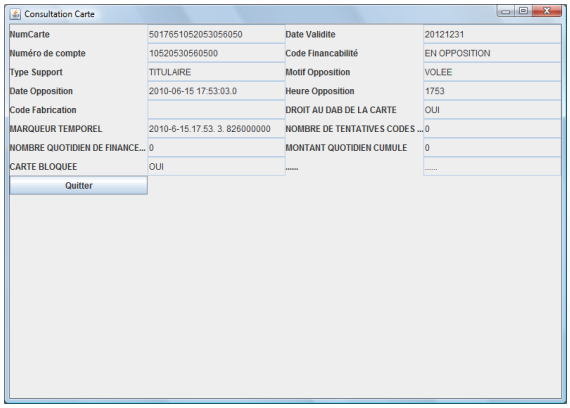

Figure 4. *Card Visualization*

5. Security Services In Case of Incident

5.1. Opposition of the Card

The personalization tool receives the card number and the expiry date, sends to the card an APDU of blocking the card and updates the database record of the card with the motive of opposition, the date and time of opposition and the flag card blocked so the card is blocked logically and physically and its use is impossible.

5.2. Opposition of the Account

The application receives the card number and the expiry date, gets back the account number and read the database record of the card to get back the account number and proposes the putting in opposition one by one every card of the account. In this case the physical blocking is impossible, only a logical blocking is made but allows rejecting all the On-line transactions on the account.

If all the cards of the account are in opposition, the account becomes in opposition.

5.3. Cancelling the Opposition of the Account

It is the inverse operation of the setting an account in opposition, the application suggests cancelling the opposition on every card attached to the account, every card becomes useful and the account becomes again useful and valid.

5.4. Blocking the Card

The application sends an APDU of blocking the card to the smart card and updates the database record of the card by setting the flag "blocked card" to 1.

A command of blocking card in the state "sent" is inserted into the commands database.

Table 14. *blocking the card*

CLA	INS	P1	P2
0x90	BLOCK_CARD	0x00	0x00

5.5. Unblocking the Card

The application sends an APDU for unblocking the card physically and updates the database record of the card by setting the flag "blocked card" to 0.

Table 15. *unblocking the card*

CLA	INS	P1	P2
0x90	UNBLOCK_CARD	0x00	0x00

5.6. Blocking the Application

The application sends an APDU for blocking an application into the card and updates the database record application/support by setting the flag "application blocked" to 1.

Table 16. *blocking the application*

CLA	INS	P1	P2	LC
0x90	BLOCK_APPLI	0x00	0x00	0x00

5.7. Unblocking the Application

The application sends an APDU for unblocking an application into the card and updates the database record application/support by setting the flag "application blocked" to 0.

Table 17. *unblocking the application*

CLA	INS	P1	P2	LC
0x90	UNBLOCK_APPLI	0x00	0x00	0x00

The smart card sends the response:

Table 18. *acknowledge*

SW1	SW2
90	00

This answer indicates to the application that the desired operation was done successfully.

6. Management of Electronic Purse

The e-money made its appearance with the electronic purses which allow to store and to manage a sum of money deposited by the card owner by a load operation.

This money which is stored in an electronic memory (EPROM) has a value because she allows realizing transaction of small amounts independently of the issuer because the transaction is entirely realized on the chip.

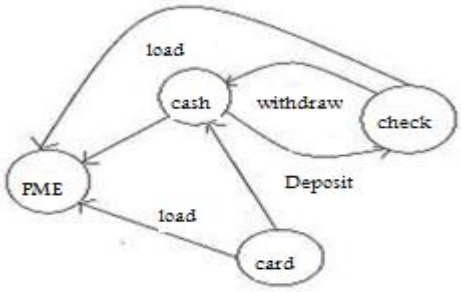

Figure 5. *Use cases of load of the PME*

After the personalization of the smart card and the authentication by the application and the entering of the pin code , the transactions of load of the electronic purse with cash, check or from the bank account attached to the card, the transaction of payment by electronic purse and the displaying of its balance are authorized by the smart card and this within the thresholds pre-personalized in the smart card, we are thinking about the maximum amount of a transaction and the maximal limit of the balance of electronic purse. As well as the technical constraints imposed by the JavaCard technology (supporting of byte and short) and not supporting of integer and float.

6.1. Load PME

After RSA or ECDSA authentication by the application and the entering of the pin code, the acquisition software suggests choosing an amount multiple of 10 Euros and the mode of load (Load CB, in cash or in check), in the case of a load CB, an authorization request is sent to authorization server of the card issuer and all the rules applied to an on-line payment is applied and in the case of a positive answer of the issuer, the software sends an APDU to the card which applies its rules with regard to the validity of the pin code and the respect for the maximum threshold of the balance, in case of success the balance of PME is updated and a record of "load PME" is inserted into the database of the On-line records.

The application sends the following APDU to the card

Table 19. *request of the load of PME*

CLA	INS	P1	P2	LC	DATA
0x90	LOAD_PME	0x00	0x00	0x01	<= 127

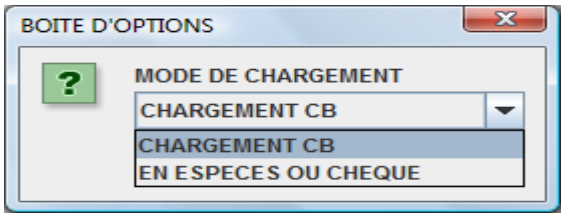

Figure 6. *choice of the load mode*

6.2. Payment by PME

The transaction of payment by PME is an Offline transaction where all the treatments are made by the smart card, we think about the control of the pin code, the respect for the maximum amount of the transaction and the affordability of PME (The PME balance > = amount of the transaction, no overdraft is tolerated in the case of a payment by PME).

The software sends an APDU of request of payment by PME to the smart card

Table 20. *request of the payment by PME*

CLA	INS	P1	P2	LC	DATA
0x90	PAYMEN_PME	0x00	0x00	0x01	<= 127

6.3. Displaying The Balance Of Electronic Purse

After authentication of the card by the software and the entering and validation of the pin code, the software sends an APDU to the smart card.

Table 21. *request of the balance*

CLA	INS	P1	P2	LE
0x90	DISPLAY_PME	0x00	0x00	0x02

The smart card sends the response:

Table 22. *reception of the balance*

DATA	SW1	SW2
BALANCE	90	00

The software read the balance in the array of bytes and calculates his decimal value.

7. Displaying The Offline Records

Every Offline transaction in success generates a record in the table of the Offline records and these records are displayed via the menu "Management Card / Displaying Offline records".

The software receives the card number and the expiry date, read the table of the Offline records and shows the history of the records in the inverse chronological order and the amount sign and the transaction amount, the date and time of the transaction, the mode of payment and the authorization number for every transaction.

8. Displaying The Online Records

Every On-line transaction generates a record in the table of the On-line records and these records are displayed via the menu "Management Card / Displaying the On-line records".

The software receives the card number and the expiry date, read the table of the On-line records and shows the history of the On-line records of the card as well as the data account number, sign, amount of the transaction, balance sign before transaction, balance before transaction, timestamp, mode of payment and authorization number for every transaction.

Table 23. ***Financial Record Description***

Field	Format
Card Number	Number(19)
Expiry Date	Date
Account Number	Number(14)
Last Name * First Name	Varchar2(255)
Amount sign	Number(1)
Amount	Number(15)
Balance sign	Number(1)
Balance	Number(15)
Payment Mode	Varchar2(4)
TimeStamp	TimeStamp
Autorisation Number	Number(15)

9. Displaying The Commands Sent To The Card

The personalization tool sends to the card commands of blocking , unblocking card, blocking, unblocking application and ask the issuer to create a command record for every command sent to the card, this feature displays the history of sent commands and the data of every command including timestamp and the type of command.

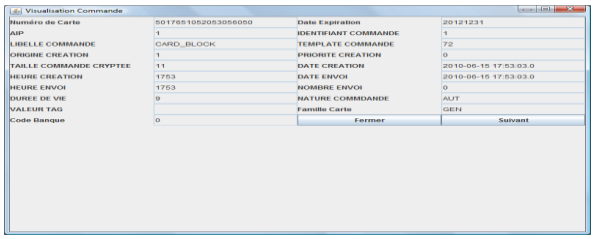

Figure 7. ***Command Visualization***

10. Application Access

The access to the application is authorized in users' list having a valid password, the password is an imprint SHA of a password known by the user.

All the secret data, the password, the personal identification numbers are stored under format imprint SHA.

To resist to every type of attacks and mainly the injections SQL, we used only prepared requests.

To use the software, it is necessary to know user's name and password and to make a successful RSA or ECDSA authentication what is very difficult being given the complexity of the RSA and the resolve the discrete logarithm in a finite group.

11. Protocols Description

11.1. APDU Description

Table 24. ***APDU Description***

Field	Length	Description	Code
Instruction	1 (0x90)	Instruction Class	CLA
Instruction code	1	Instruction Code	INS
Parameters	2	Instruction Parameters	P1-P2
Lc	0, 1 ou 3	Length of data	-
Data	Lc	Lc bytes of data	-
Le	0,1,2 ou 3	Le bytes expected	-
Response data	Lr	String of Lr bytes	-
Status bytes	2	Commande processing status	SW1-SW2

12. Implementation of the Applet

```
protected TheApplet() {

publicRSAKey = privateRSAKey = publicIssuerRSAKey = privateIssuerRSAKey = null;

cRSA_NO_PAD = null;

cipherRSAKeyLength = KeyBuilder.LENGTH_RSA_1024;

// build RSA pattern keys

publicRSAKey = KeyBuilder.buildKey(KeyBuilder.TYPE_RSA_PUBLIC, cipherRSAKeyLength,
false);

privateRSAKey = KeyBuilder.buildKey(KeyBuilder.TYPE_RSA_PRIVATE, cipherRSAKeyLength,
false);

// initialize RSA public key

((RSAPublicKey)publicRSAKey).setModulus(N, (short)0, (short)N.length);

((RSAPublicKey)publicRSAKey).setExponent(e, (short)0, (short)e.length);

// initialize RSA private key

(RSAPrivateKey)privateRSAKey).setModulus(N, (short)0, (short)N.length);

((RSAPrivateKey)privateRSAKey).setExponent(D, (short)0, (short)D.length);

// get cipher RSA instance

cRSA_NO_PAD = Cipher.getInstance((byte)0x0C, false );

signatureRSA=Signature.getInstance(Signature.ALG_RSA_SHA_PKCS1 ,false);

signatureRSA.init(privateRSAKey,Signature.MODE_SIGN);

publicIssuerRSAKey = KeyBuilder.buildKey(KeyBuilder.TYPE_RSA_PUBLIC,
cipherRSAKeyLength, false);

privateIssuerRSAKey = KeyBuilder.buildKey(KeyBuilder.TYPE_RSA_PRIVATE,
cipherRSAKeyLength, false);

((RSAPublicKey)publicIssuerRSAKey).setModulus(N_ISSUER, (short)0, (short)N_ISSUER.length);

((RSAPublicKey)publicIssuerRSAKey).setExponent(e, (short)0, (short)e.length);

((RSAPrivateKey)privateIssuerRSAKey).setModulus(N_ISSUER, (short)0, (short)N_ISSUER.length);

((RSAPrivateKey)privateIssuerRSAKey).setExponent(D_ISSUER, (short)0,
(short)D_ISSUER.length);

signatureIssuerRSA=Signature.getInstance(Signature.ALG_RSA_SHA_PKCS1 ,false);

register();

}
```

13. Verification of the RSA Signature by the Client

```
private boolean verifySignRSA(String mod, String exp,String plain,byte [] empreinte)
{
try{
        Signature signature=Signature.getInstance("SHA1withRSA"); RSAPublicKey
        pkey=(RSAPublicKey)KeyFactory.getInstance("RSA").generatePublic(new
        RSAPublicKeySpec(new BigInteger(mod,16),new BigInteger(exp,16)));
        signature.initVerify(pkey);
        signature.update(plain.getBytes());
        byte [] t=new byte[128];
        System.arraycopy(empreinte,(short)0,t,(short)0,(short)128); // la partie utile de l'empreinte est
        de taille 128
        boolean res=signature.verify(t);
        return res;
}catch (Exception e){
JOptionPane.showMessageDialog(this,"Probleme dans la vérification de la signature RSA!!"+e);
        return false;}
}
```

14. Conclusions

Bank cards are more and more secure and support the most modern algorithms of ciphering and generating signature as RSA and ECDSA and we have proved thanks to this software that we can personalize quickly smart cards so that they become bank cards offering several services with a very low cost and with a very high level of security.

The incidents of theft and loss of bank cards are rising and with the internet and mobiles media it is very easy for a holder of a card to realize fraudulent transactions putting in financial trouble the real holder so this software allows putting in opposition a card with or without physical support

The standard EMV was created to counter a galloping rise of the fraud by card further to incidents of theft, loss of bank card, absent cards and our software respects this standard because it sets up a multi-application personalization and allows sending commands to smart cards further to an incident of this type.

Our software is very similar to software deployed in a bank branch to realize the operations of creation of accounts, of cards and personalization of card and after-sales services with a very high security level and can serve as prototype of an application of future that the banking agent can participate on all the cycle of manufacturing and personalization of bank cards because at present several services providers participate in this process making high the cost of manufacturing one card.

References

[1] D. Pointcheval and P. Nguyen, 1st Edition., **(2010)**, XIII, pp. 519, Softcover, ISBN: 978-3-642-13012-0, 13th International Conference on Pratice and theory in Public Key Cryptography, PKC 2010 May 26-28 in Paris.
[2] M. N. Nabi, S. Mahmud and M. L. Rahma, "Implementation and performance analysis of elliptic curve digital signature algorithm", Bangladesh.
[3] H. Mammass, "Sécurisation des cartes bancaires par les scripts automatiques EMV", SETIT 2007, **(2007)** March, Hammamet-Tunisia.
[4] Jean-Sebastien Coron, "Cryptanalyses and Security Proofs of Public-Key Schemes", PhD Thesis 2001 at Ecole Normale Supérieure in Paris, **(2001)**.
[5] A. Menezes, P. van Oorschot and S. Vanstone, "Handbook of Applied Cryptography", CRC Press, **(1996)**.
[6] C. Delannoy (EYROLLES), Programmer en Java 5ème édition Java 5 et 6, pp. 787.
[7] M. H. Sherif , (EYROLLES), La Monnaie électronique, Systèmes de paiement sécurisé

Authors

Hafid Mammass

1991: Engineer of computer science (institut national polytechnique de Toulouse)

1993-2012: Engineer in several big companies in France

2010: Specialized Master security of information and systems (ESIEA, Paris)

2011: Specialized Master Architect of computers and systems (Rabat, Morocco)

2012: Phd Student at University Ibn Zohr Agadir (Morocco)

Fattehallah Ghadi

Professor Fattehallah Ghadi is a professor of higher education and is currently the Vice President of Ibn Zohr University in Agadir, Morocco. His research investigates two main areas. The first is the analysis and research of optimal finite elements and optimal algorithms applied to problems of fluid mechanics. The other area of his research focuses in finding optimal algorithms in graph theory applied to several fundamental issues such as connectivity, routing and mobility.