Not So Gassed
Hardware Security 2018

Team Zes
Bauke Brenninkmeijer - s4366298
Wesley van Hoorn - s4018044
Ties Robroek - s4388615
Mathijs Sonnemans - s4738799
Aniek den Teuling - s1010747
All from RU

June 11, 2018

Contents

1 Introduction **4**

2 Use Cases **4**
2.1 Initialization . . . 4
2.2 Charging allowance . . . 4
2.3 Gas transaction . . . 4
2.4 Lost or stolen . . . 5
2.5 Decommissioned . . . 5

3 Assets **5**
3.1 Smart cards . . . 5
3.2 Charging terminal . . . 6
3.3 Pump terminal . . . 6
3.4 Server . . . 6
3.5 Cryptographic keys . . . 6

4 Stakeholders **6**
4.1 Government . . . 6
4.2 Motorist . . . 6
4.3 Gas station owners . . . 6
4.4 Manufacturer . . . 6

5 Attacker Model **6**
5.1 Assumptions . . . 7
5.2 Gas station owner . . . 7
5.3 Motorist . . . 7

6 Security Requirements **7**
6.1 Smart card . . . 7
6.2 Charging terminal . . . 8
6.3 Pump terminal . . . 8

7 Overall Design **8**
7.1 Data overview . . . 8
7.2 Card initialization protocol . . . 9
7.3 Pump terminal initialization protocol . . . 9
7.4 Charging protocol . . . 10
7.5 Pumping protocol . . . 11
7.6 Card decommissioning protocol . . . 11

8 Implementation **11**
8.1 Transient state . . . 11
8.2 Persistent state . . . 12
8.3 Design choices . . . 12
8.3.1 Data types . . . 12

9 Further Improvements **12**

List of Figures

1 Use case diagram . . . 4
2 Gas transaction activity diagram . . . 5
3 Card initialization protocol . . . 9

4 Pump terminal initialization protocol 9
5 Charging protocol 10
6 Pumping protocol 11
7 Transient state diagram 11
8 Persistent state diagram 12

List of Tables

1 Values stored in the system and their notation 8

1 Introduction

We implemented a smart card system to maintain gas rationing. Motorists are assigned a smart card. This card holds a balance for assigned gas. Users are able to charge the card once a month. This allowance is determined by the government. Gas can only be pumped following presentation of the smart card.

2 Use Cases

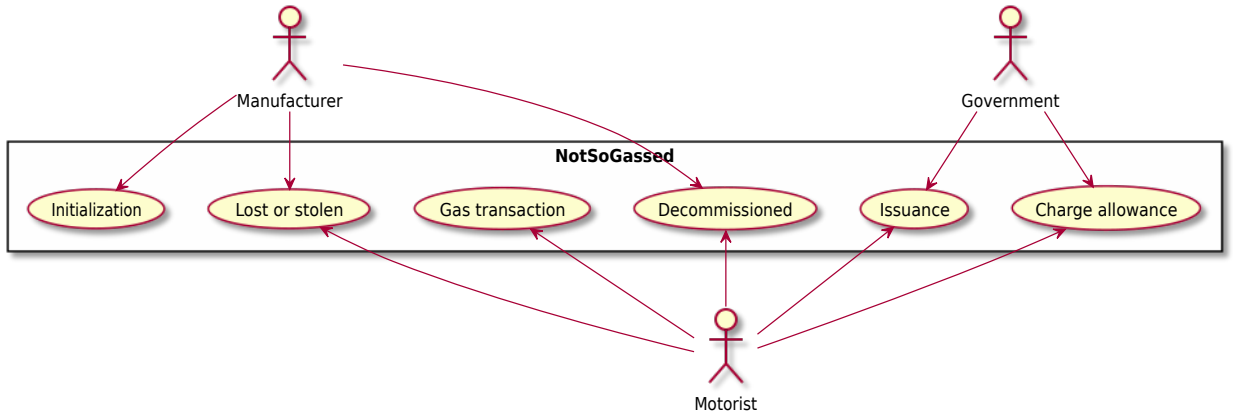

Figure 1: Use case diagram

2.1 Initialization

The smart card manufacturer is responsible for the initialization process of the cards. Every card has a unique key pair, it is the manufacturer's responsibility to provide these keys on the cards. During the initialization, the server sends a command that blocks personalization, as described in 7.2. The government keeps a sufficient stock of smart cards. This allows them to issue a card to a motorist within a short period of time.

A motorist is eligible for the smart card when he or she is in the possession of one or more vehicle registration certificates (VRC). The DMV carries out the smart card delivery similar to how it carries out VRCs. The Department of Motor Vehicles (DMV) authenticates the individual and supplies a factory-new smart card. An ID number is assigned to the card and paired with the motorist in the database. The public key is appended to the list of authorized cards in the server.

2.2 Charging allowance

A motorist in possession of the smart card is entitled to a fixed amount of gas each month. This amount is specified by the government. The amount of gas allowed per month is termed "allowance" and is expressed in liters. The remaining allowance for a certain month is kept track of by the server. This means that when an amount lower than the monthly maximum allowance is charged, the remaining amount is stored by the server. Allowance can only be charged to authenticated cards at verified charging terminals.

Changes proposed by the government are invoked in the new month.

2.3 Gas transaction

Pump terminals works in a fairly simple manner. They are not connected to the Internet. Pump terminals accept smart cards, which will stay in the machine during the encounter. The total balance of the smart card will be deducted upon insertion of the card, to discourage early retraction. After pumping gas, the card will be recharged with the original amount minus the amount that was pumped. The pump terminal will have a screen which displays the amount pumped and the remaining balance.

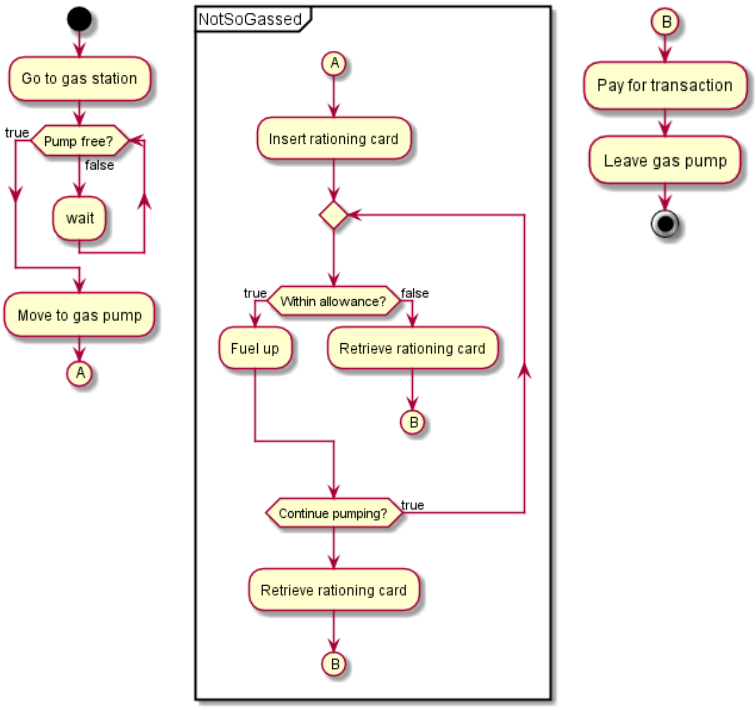

Figure 2: Gas transaction activity diagram

The pumping will stop when either it is manually stopped or the the balance minus the amount pumped becomes zero. When the balance becomes zero, nothing is recharged to the card.

It should be noted that the Not So Gassed system functions independently from the usual gas transaction. The system does not interact with the operator; it merely limits the maximum size of possible transactions. An example of a complete gas transaction activity from the motorists perspective is presented in figure 2.

2.4 Lost or stolen

In case of loss or theft the motorist should contact the DMV to report the card missing. The DMV authenticates the motorist and updates the concerning card's ID number in the server to inactive. A new smart card can be requested; The card ID linked to the motorist in the database will be altered. See subsection 2.1.

2.5 Decommissioned

A smart card that is broken or not used is up for decommissioning. The motorist should contact the DMV and request a return authorization (RA). The DMV authenticates the motorist. If the RA is approved, the concerning card record is removed from the server. The remains of the smart card have to be sent to the DMV. They will recycle or destroy the card. In case of malfunction, the user can request a new smart card; see subsection 2.1.

3 Assets

3.1 Smart cards

Every motorist holds a smart card that is running our applet. Motorists may use these cards at both the charging and the pump terminals. The motorist is responsible for his or her own card.

3.2 Charging terminal

Motorists can retrieve their monthly gas allowance by presenting their smart card to a charging terminal. These terminals are connected to the Internet as a way to communicate with the server. Charging terminals are stationed throughout the country.

3.3 Pump terminal

The pump terminals are similar to gas pumps currently in operation at gas stations. They will be modified to have a terminal that accepts these smart cards.

3.4 Server

The system uses a server that is accessible for different steps in the protocol. The issuance, charging and decommissioning procedures approach the server through an Internet connection. The server has information of every card stored. It is assumed that the communication between the charging terminals and the server is secure. The same applies for the communication regarding card initialization and issuance.

3.5 Cryptographic keys

All smart cards and pump terminals have an RSA key pair consisting of a private and a public key. In addition, there is a global key pair which is stored on the server. The global public key is also stored on the cards and pump terminals; see table 1.

4 Stakeholders

4.1 Government

The government issues the cards and is administrator of the server. This makes the government one of the major stakeholders in this new system.

4.2 Motorist

The motorist is the card holder. Motorists can acquire a card from the government. The only way to retrieve gas at gas stations is to use this smart card. Their dependency on the card for gas makes them a major stakeholder.

4.3 Gas station owners

Gas station owners collect the rations and will have to prove to the government that this has been done. These rations are stored in a log on the pump terminal, and have to be extracted by the owner. The government will sample a number of gas stations every month to verify the integrity of the gas stations.

4.4 Manufacturer

The card manufacturer is the final stakeholder in the system. It is the organization producing the cards. If the cards themselves fail to fulfill the government's needs in the long run or have some defects, the manufacturer might lose money, public trust and corporate trust.

5 Attacker Model

We split this section into the gas station owner and everyone else. The owner is different in the attack perspective, since they have increased physical access to the pumps. Firstly, we list our assumptions about the system since that is what our protocols are based upon.

5.1 Assumptions

We assume a server network that is accessible over the Internet by charging terminals, for which the security is out of the scope of this project. We assume that both the smart cards and the pump terminals are tamper resistant but it would be possible that fake cards and or terminals are used. We will consider man-in-the-middle attacks and card tears on the transactions as this is possible through skimming. Vandalism, such as physically blocking the card entrance, is also outside the scope of the attacker model. In the protocols we assumed that a global decrease of total allowance is favorable over a possible increase as the thought behind the system is rationing.

5.2 Gas station owner

Attack: tamper with allowance
Motive: economic benefit

The operator benefits from having their customers exceed their allowance. This will allow customers to purchase more gas from the operator, increasing sales. Possible attacks include reducing or removing the allowance cost at the pump terminal. Another attack would be to tamper with a charging terminal allowing customers to repeatedly charge their card. Additionally, the owner may be motivated to sell gas without charging the customers allowance. for example on the black market. The gas station operators may also be motivated to sell gas without collecting rationing allowance, i.e. on the black market.

5.3 Motorist

Attack: add allowance
Motive: economic benefit

A motorist poses a threat by aiming for retrieving more gas than the monthly allowance. Example attacks are removing the gas restriction from the smart card, charging the card multiple times a month or fooling the pump terminal into decreasing the balance with a smaller amount than pumped. Since all car owners (could be anyone over 18) are motorists, the attacker's capabilities in terms of knowledge, skills and expertise vary enormously. For all motorists holds that they have free physical access to pump and charging terminals.

Motorists may attempt to following attacks:

- Malicious card injection; A malicious smart card can be presented to a charging terminal in an attempt to gain control over the transaction.
- Skimming; A fake charging terminal is able to trick potential users into presenting their smart card. This means that all transaction data is essentially prone to be intercepted.

6 Security Requirements

Note: We assume that the individual smart cards and the server are tamper proof. Therefore the integrity of all private keys is assumed.

6.1 Smart card

1. Authentication of the charging terminal to the smart card.
2. Authentication of the pump terminal to the smart card.
3. Confidentiality of the private key on the smart card.
4. Integrity of the card's ID number.
5. Integrity of the allowance.

6.2 Charging terminal

6. Authentication of the smart card to the charging terminal.
7. Integrity of the session.
8. Non-repudiation; the user is unable to deny that the charging terminal has increased the balance, i.e. once the monthly allowance is retrieved, one is unable to claim that one has not received allowance.

6.3 Pump terminal

9. Authentication of the smart card to the pump terminal.
10. Integrity of the gas transaction.
11. Non-repudiation; the user is unable to deny that the pump terminal has reduced the balance, i.e. once the balance has decreased, one is unable to claim that one has not made a gas transaction. The pump terminal must also be able to prove that the transactions have taken place via a log, and moreover, not introduce fake transactions.

7 Overall Design

7.1 Data overview

The data overview is stated in Table 1. The smart card has six essential values stored: the card private key, the card public key, the card id, the allowance, the global public key and its card certificate. The pump terminal contains the terminal private key, the terminal public key, the global public key and its terminal certificate. The server has the global private and public key stored. It also contains the IDs of all cards, all card public keys and the allowance of each card.

Table 1: Values stored in the system and their notation

Item	Value	Notation
Card	card_private_key	skC
	card_public_key	pkC
	card_id	ID
	allowance	A
	global_public_key	pkG
	certificate	cC = sign(pkC, skG)
Pump Terminal	terminal_private_key	skT
	terminal_public_key	pkT
	global_public_key	pkG
	certificate	cT = sign(pkT, skG)
Server	global_private_key	skG
	global_public_key	pkG
	for every card:	
	card_id	ID
	card_public_key	pkC
	allowance	A

7.2 Card initialization protocol

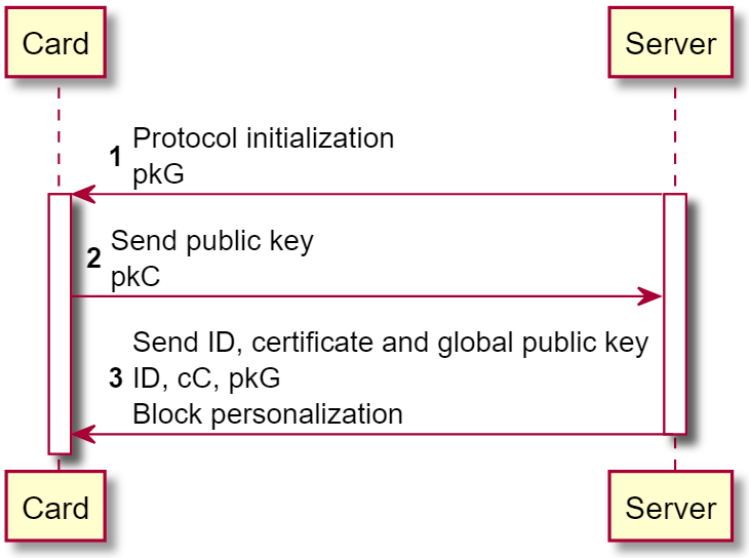

Figure 3: Card initialization protocol

During the initialization, a factory-new card is added to the system and made ready for issuance. In order to do this it has to verify that the card presented is untampered and authentic. Then all required variables are initialized. This process happens in a secured environment.

1. The server initializes the protocol and sends the *pkG* to the card.
2. The card saves the global public key and sends the *pkC* to the server, which then adds it to the record of authorized cards.
3. The server sends the card *ID* and the card certificate to the card. The certificate is generated by the server by doing cC = sign(pkC, skG). It also sends a command to the card to block personalization.

7.3 Pump terminal initialization protocol

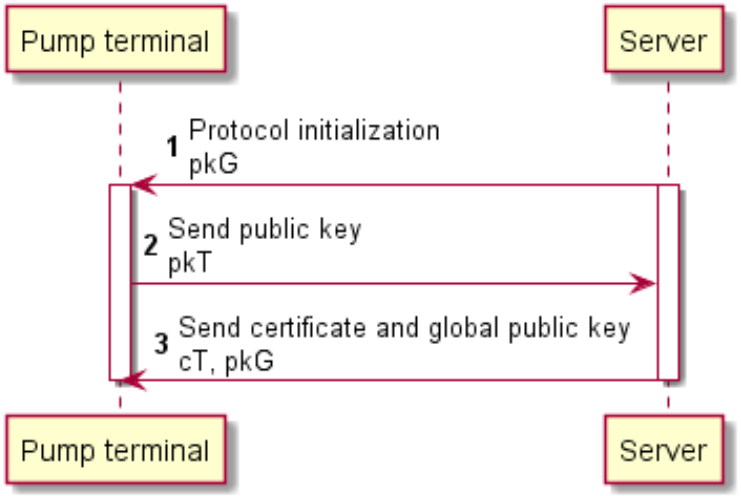

Figure 4: Pump terminal initialization protocol

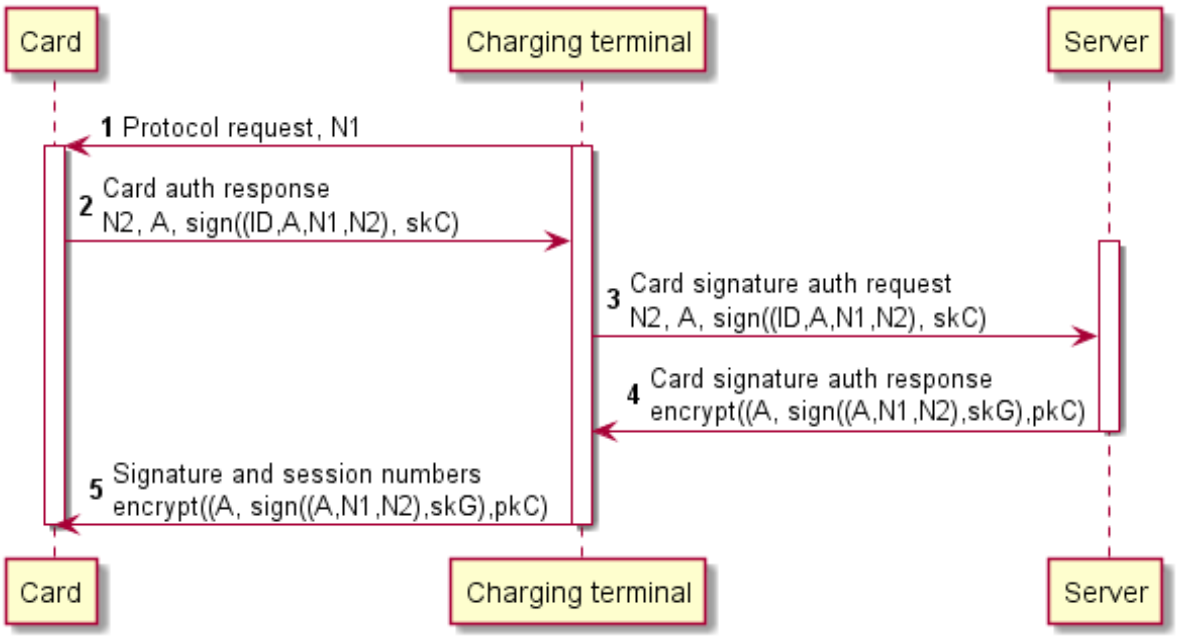

Figure 5: Charging protocol

Each pump terminal has to be initialized in order to be able to handle allowance transactions in an offline environment. The terminal will obtain its own key pair in the factory or any other secured environment. Then the terminal has to be certified.

1. The server initializes the protocol and sends the *pkG* to the terminal.
2. The terminal saves the global public key and sends the *pkT* to the server system.
3. The server sends the terminal certificate: the terminal's public key signed by the server cT = sign(pkT, skG) to the terminal for it to store it.

7.4 Charging protocol

This protocol is performed to update the monthly allowance of a person on their card. The server verifies that the provided card is valid and the card has to verify that the server attached ot the terminal is valid. It does this with the response, signed with the global secret key.

1. The protocol starts with an authentication request to the card by the terminal. This includes a description of which protocol currently is being started. The terminal also sends a generated nonce N_1
2. The card responds with sending its own nonce N_2, the allowance stored on the card, and a signature of the *ID* appended with the allowance A, N_1 and N_2, signed by the *skC*.
3. The terminal passes the information to the server. It will verify if the signature sent by the card is valid by using the *pkC* stored on the server. If this is the case it will proceed with the protocol.
4. The server sends the A stored on the server and a signature of the A appended with the two nonces signed with the skG. This is encrypted with the *pkC*. Finally, the server sets the A of the card on the server to 0.
5. The terminal passes the server data to the card. The card decrypts the message and verifies the authenticity of the message by verifying if the signature is decryptable with pkG and the content is correct. It then stores the new allowance.

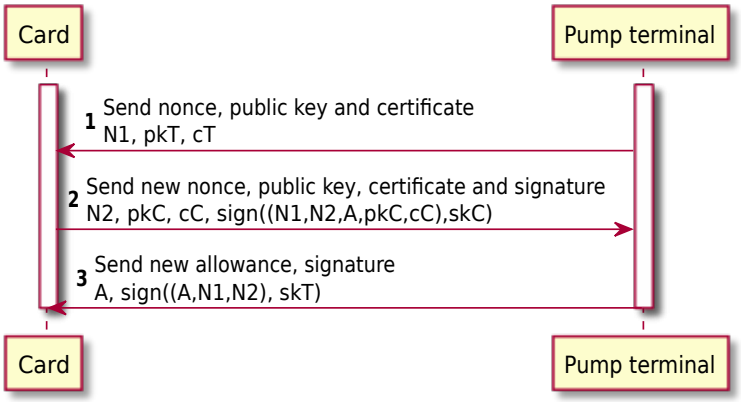

Figure 6: Pumping protocol

7.5 Pumping protocol

The pumping protocol ensures that the correct amount of gas pumped is subtracted from the allowance stored on the card. It also ensures that no more than the remaining allowance can be pumped. The authenticity of the card by the terminal is determined but also the authenticity of the terminal by the card.

1. The terminal sends a nonce N_1, the pump terminal's public key pkT and the terminal certificate cT to the card.
2. The card verifies the terminal's key certificate. It generates a new nonce N_2 and then sends this nonce, the pkC and the card certificate cC to the terminal while also sending an signature of this whole message plus the new nonce and the amount: N_1 and N_2, A, pkC and cC. It also sets the amount on the card to 0.
3. The terminal verifies the card by verifying the certificate with pkG. It then verifies the message and checks the nonces. It will then start pumping until the allowance is pumped or the user stopped pumping. Once this is done it will send the new allowance and a signature of the nonces N_1 and N_2 and the new allowance signed with skT.
4. The card will verify the signature and update the allowance.

7.6 Card decommissioning protocol

The decommissioning protocol ensures a card is safely removed from the system by removing it from the database.

8 Implementation

8.1 Transient state

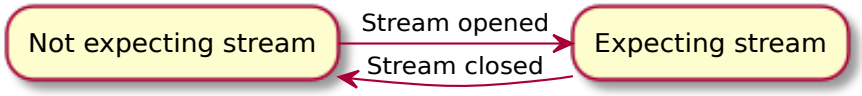

Figure 7: Transient state diagram

Our implementation does not keep track of a general transient state variable. The only real state that is kept is when a stream is initialized, which is visualized in figure 7. The implementation will then wait a certain

amount of packets worth of data before being able to accept new requests. Packages sent in this expected data window are lost when they do not belong to the expected data.

8.2 Persistent state

Figure 8: Persistent state diagram

The persistent state is not implemented by using a variable with a certain value. Rather, we have chosen for an implicit state that is defined by the values of certain variables. For example, a card that is not issued will not work in the system, and neither will a decommissioned card. As explained earlier, this is done by adding and removing the card ID from the server. In some sense, this could be regarded as an implicit state variable, but still is not an explicit one.

8.3 Design choices

In our implementation numeral decisions were made with regard to certain details. The options for encryption and hashing are kind of restricted in JavaCard 2.2.1. The hashing options are quite limited and do not support SHA2, so we use SHA1. For encryption, we needed an asymmetric algorithm. We chose RSA 1024 for this, due to the good balance between size of keys and security. Our system is implemented in such a way that limited adaptations will enable the usage of RSA 2048. This includes the changing of some messages to streaming messages. We padded all our data with pkcs1 to prevent weaknesses with RSA on unpadded data.

Due to the limited support for different algorithms, some specific functions exist in JavaCard to allow easy usage of encryption/hashing algorithms. This made implementation considerably easier.

8.3.1 Data types

The most used data type in our system are shorts. When data is larger than this, it is mostly represented as byte-array. This is always the case on the card, since the card does not support big-integers. However, due to the requirement of Bouncy Castle to use big-integers, sometimes these byte-arrays are converted to big-integers in the terminals. The data that doesn't fit in shorts are the keys for encryption and related modulo calculations. The keys that are saved on the server are also represented as byte-arrays, allowing for easy sending.

9 Further Improvements

RSA 2048 is more robust and future-proof than RSA 1024. Implementing this may impact the performance though. The change will increase the size of the blocks which may result in more and longer byte transfers. It will be simple to adapt RSA to 2048 bits, although the impact on the performance should be tested as it might perform significantly slower. On this version of JavaCard, only SHA1 and MD5 available although both are not considered to be secure anymore. We recommend that in the future, a different JavaCard version is used in order to support atleast SHA2.