Mobile Computing

Practical Assignment #1 / Design and Development

Shop and pay Android app for a supermarket

The Acme Electronic Supermarket

1. Scenario

A supermarket – call it the Acme Electronic Supermarket – intends to implement a more efficient shopping and payment system supplying an Android app to their customers.

Customers must register the products in the app, when they put them in a shop basket. When all the shopping is done, customers check out the products in the basket passing or showing the phone in a terminal. The payment is made (using the associated credit or debit card), and the gate doors are opened.

Payment terminal

Electronic supermarket

For start using the app, the customers should first make a registration (only once, after installing the app and before using it for the first time) in the Acme remote service, supplying personal data and payment card data for payments. Also, an asymmetric cryptographic key pair is generated, being the private key securely stored in the device.

At any time, the app should be capable of consulting past transactions' information, and retrieve available emitted loyalty vouchers, that can concede a discount in some future buy. A loyalty voucher is offered whenever the customer accumulated payments surpassed a multiple of some value.

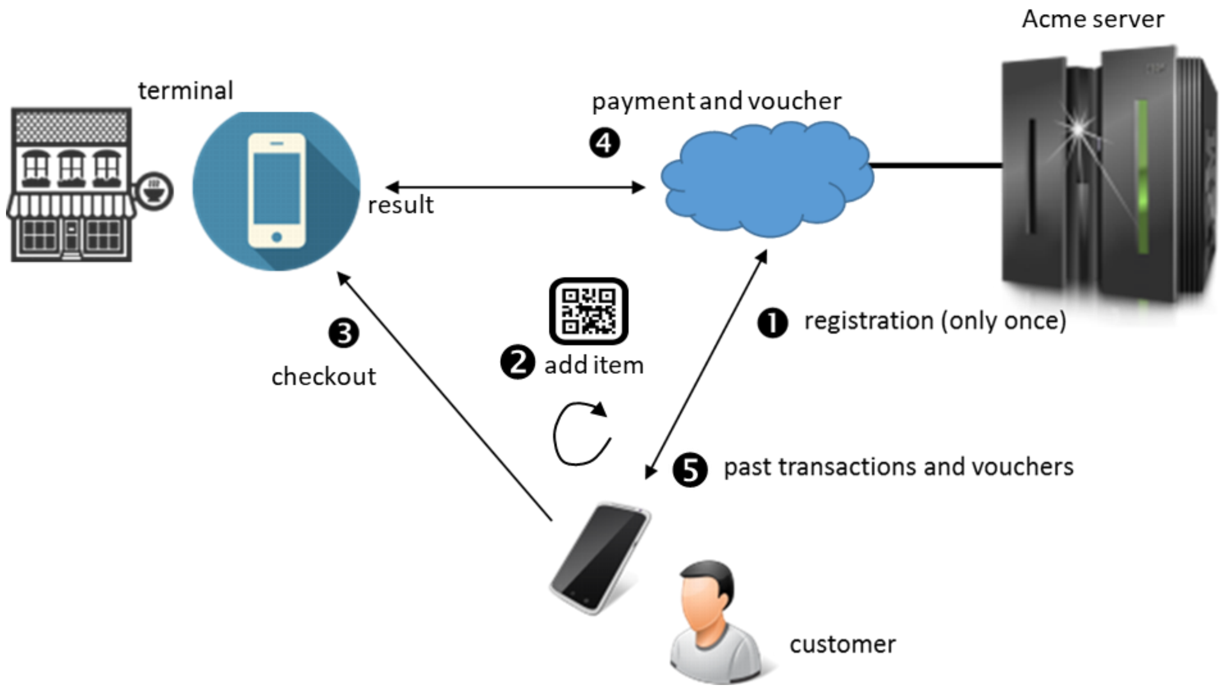
2. System applications

The shopping and payment system is composed of three different applications, namely:

1. The remote service (a REST service) located on the supermarket server (it can be divided into several groups of operations: register and validate customers, perform a checkout, consult transactions and retrieve vouchers) implementing a simple back-office with the operations needed by the main Android app and terminal.
2. The Customer App allowing him to register himself in the system, consult past transactions and retrieve vouchers, initiate and add items to the shopping basket by reading a tag or QR-code in each product, and performing the checkout, possibly adding an already retrieved voucher, and use a discount.
3. For this experience the supermarket checkout terminals run also an Android application, receiving from the customer app and phone the items list, and optionally **one** voucher. The terminal communicates this information to the server. The server executes the payment and transmits back to the terminal the operation result and the total value charged. The terminal opens the gate doors (not performed in this proof of concept).

3. Operations and interactions

The typical set of operations and interactions between these software applications are depicted in the following diagram:



They should perform, at least, the following:

1. Registration – The first operation the customer app should do is to register the customer in the supermarket service, using an internet connection. The customer should supply at least his name, a user nickname and password (used only for a local login) and payment card information. In this process, also a cryptographic key pair should be generated in the phone. The private key is stored locally and securely, and the public key is transmitted, with the other information, to the server, which takes note of all these items in its database.
   In response, the server should generate a unique user identifier in the format of a UUID (16-byte value) and transmit it back to the app, that should store it. Together with the previous **id**, the server sends also the supermarket public key, used to read the product labels.

2. Adding to the shopping basket – Whenever the customer puts something in his basket, he should read the QR-code label in the product. Those QR-codes contain a product code (unique for each product unit and being also a 16-byte UUID), the price (euros and cents), and the product name (a string), all *encrypted* with the *supermarket private key* (to prevent producing fake labels).
   These readings are accumulated in a shopping list inside the app.

3. Checkout – When the user wants to abandon the supermarket, he performs a checkout operation, that will generate a QR-code in the screen, or an NFC message, and present the phone to the terminal near the exit gates. The QR-code, or message, should contain the list of products (for the purpose of this implementation we will consider a **maximum of 10 items**) containing each item the **id** and **price** only, the user id (stored in the app during registration), optionally **one** voucher id, selected by the user, and also if he wants to discount the amount already accumulated so far (a boolean). All this information should be **signed** by *the private key of the user*, and the signature appended.

4. Payment and result – The terminal reads the QR-code (or NFC message) and passes all this information to the server for payment. The server calculates the final value to pay and subtracts the existing accumulated discount (zeroing this value), if the user asked for it. Also, if a voucher is included (whose **id** should be verified in the server as belonging to this user) a 15% discount of the total value paid is calculated and added to the accumulated discount (in the server) so far (it can accumulate to existing discounts from previous purchases, if that amount was not used in this transaction). The total accumulated actual expense of the user is updated (adding the current value paid) and for each new multiple of a fixed value reached (say €100.00 for testing) a new voucher is generated, associated with this user, and kept on the server, for possible future retrieval and use. The success of the operation, and total amount paid, are transmitted back to the terminal, displaying it in a *very visible way*. If all has gone well, the exit gate is opened.

5. At any time, the user can ask the server (using the internet channel) for past transaction data (at least a certain number of those transactions should be transmitted back) using his user identifier, **signed** with the *user private key*, and verified at the server. If available, the server also transmits back to the app all the emitted and still unused vouchers, belonging to the user, that should replace the local list (from previous retrievals) in the app.

4. Vouchers

Vouchers are used as a loyalty strategy. They are offered to the customer whenever his total accumulated value paid surpasses a new multiple of €100.00. Whenever the user presents one of these vouchers in a checkout, a 15% discount is offered in the current transaction. This value is not immediately used but instead accumulated on the server. The user can use the accumulated discount in a future transaction, if it asks to.

A voucher is simply a sequence of bytes containing a unique identifier (a 16-byte UUID). When they are emitted the server takes note of the user possessing it. When the server receives a voucher in a payment he should check if it belongs to the correct user and was not yet used. The user should retrieve and store in the phone the emitted unused vouchers, in order to be able to use them in a checkout.

5. Cryptography

Cryptography is used to assure security. The suggested operations are two of the possible.
Labels printed with a QR-code in each product are **encrypted** with a private key, preventing anyone else to produce fake ones (e.g. changing the price). The app should have stored the corresponding *public key* to read them. This key is transmitted to the app in the registration process. The algorithm used is ***RSA with 512 bits key*** for simplification.
The checkout information is also **signed** with the *user private key* (generated in the registration process and also stored securely in the app), and the signature is transmitted together with the checkout info.
To verify the signature, the server should use the corresponding *user public key* which was transmitted to the server on the registration process. For the signature use also a key length of 512 bits with the algorithm ***SHA256WithRSA***. This produces a signature of **64 bytes** which is the shortest with standard secure algorithms. Requests for previous transactions and vouchers are also signed in the same way.

6. Communications

All the communications with the server are done using the internet and the HTTP protocol (it should be the HTTPS protocol, but let's simplify) in a REST service, over Wi-Fi, or over the phone operator network. The communication between the customer app and the terminal device in the supermarket should use NFC or a QR-code and camera. The reading of a label should use the device camera (and the QR-code on the label).
If you have two physical Android phones supporting NFC you can use it, for a simplified transmission process. If not, use the QR-code technique.
The QR-code technique can also be used between a physical phone and an emulator. QR-codes can only represent a small number of bytes, so the information coded in them should be the minimum possible (with values in binary, not strings).
If you don't have any Android physical phone available, you can use a TCP/IP connection between two different emulators.

Note: if you are using only the Google emulators the channel between them has to be TCP. See https://developer.android.com/studio/run/emulator-networking . With one real phone and an emulator, the emulator should be the user phone presenting the QR-code and the real phone can capture it with the camera.

7. Design and development

You should design and implement the set of applications capable of complying with a large as possible number of the described features and functions and demonstrate its use. The applications should have a comfortable and easy to use interface. Also, try to test the security features included (faking keys, QR-codes, signatures, id's, …). You can add any other functionalities considered convenient and fill any gaps not detailed.

You should also write a report, describing the architecture, data schemas, included features, and performed tests (functional). The applications way of use, should also be included in the report, presenting screen capture sequences. Include details about the representation of the checkout information and its signature, and the key representation.

Grades will consider ease of use and interface, the completeness of the scenario, security features, architecture, and any other innovative aspects included.