Coursework 02. Rahul Balaji - 11449565. COMP61421

Introduction

The secure mobile payment system is designed for a beverage company called “Drinks4U”, which has installed advanced drinks vending machines in railway stations around the country. The machines allow transactions via mobile payment, the protocols and security of which is addressed by this document.

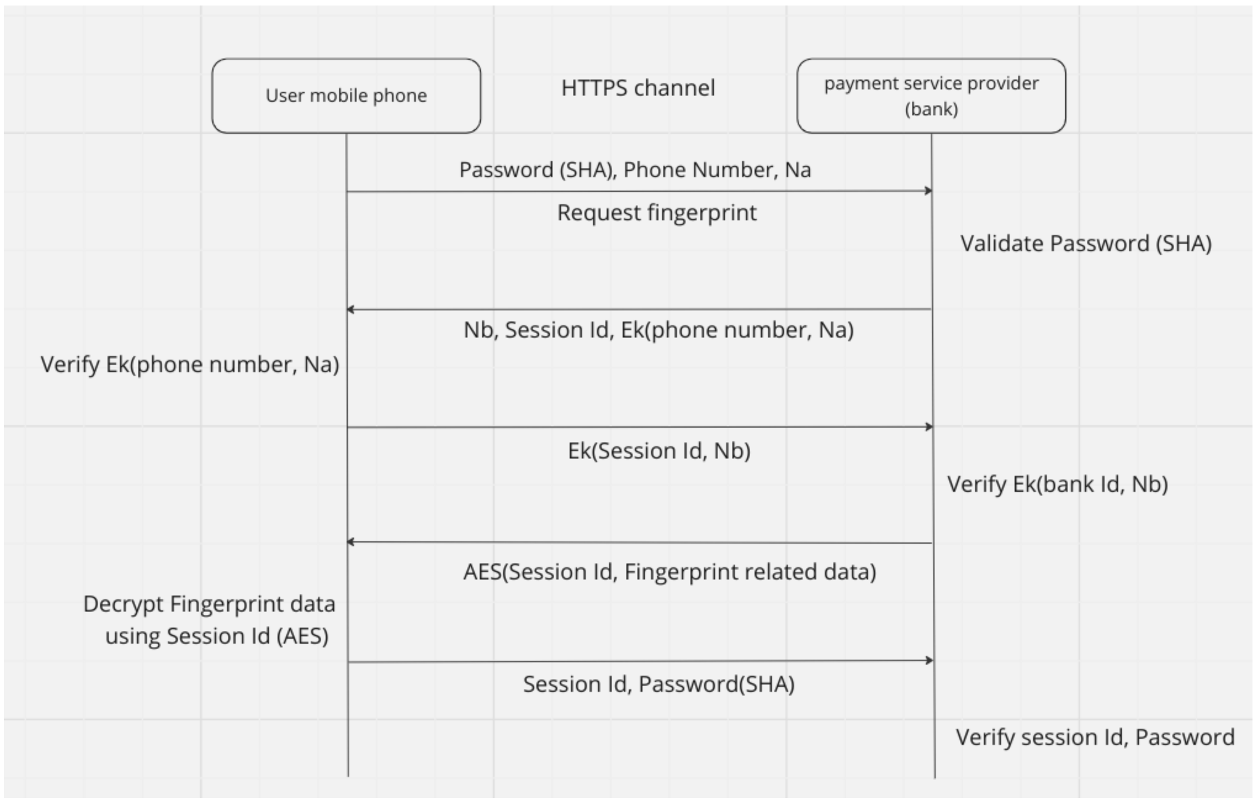
Before we proceed with the discussion on the protocols and security measures, we must first address all the assumptions made by the solutions proposed in this document. They are as listed below.

- The vending machine that the user makes a request to for a drink using the app has access to the user’s phone number when making the request.
- The phone number is associated with a bank account, which has been registered with the payment service provider and it is protected by a memorable information (password) that is known only to the bank and the user alone. There are no other shared passwords or keys between the systems.
- The phone used is assumed to have the most minimum features available for this system to work. It should support the use of AES-based symmetric cryptosystem and a secure hash function (ex: SHA-256). The mobile phone cannot run any asymmetric cryptosystems such as RSA or DH (Diffie-hellman).
- If the phone does not have a fingerprint reader to collect the fingerprint related data, then the user can download their fingerprint related data to their mobile phone from the server by authenticating their identity. We assume that the username will be the phone number in this scenario, which is linked to their bank account.
- The bank account of the user has enough money to pay for the drinks. If in case the user does not have enough money in the bank, the server will notify the vending machine, in which case the drink will not be dispensed with a suitable error message and the session terminates.

With these assumptions made, we can proceed with discussing the design of the system.

Fingerprint download service

The system allows the mobile user to download their fingerprint related data onto the mobile device when they do not have a built in fingerprint scanner device in their system. A diagram for the process of downloading the data is illustrated below.


User mobile phone
HTTPS channel
payment service provider (bank)
Password (SHA), Phone Number, Na
Request fingerprint
Validate Password (SHA)
Nb, Session Id, Ek(phone number, Na)
Verify Ek(phone number, Na)
Ek(Session Id, Nb)
Verify Ek(bank Id, Nb)
AES(Session Id, Fingerprint related data)
Decrypt Fingerprint data using Session Id (AES)
Session Id, Password(SHA)
Verify session Id, Password

This is a five step communication protocol to download the fingerprint related data. It begins with the user requesting for the fingerprint data by authenticating their session by providing their password, which is the only known secret memorable information shared between the user and the bank along with the phone number and a nonce Na as a challenge.

It is worth noting that the password shared is transformed using the SHA-256, so that in the event there is an eavesdropper that is listening on the channel, the password is protected in such a way that the original data cannot be derived from the information given.

The password is then validated by the server of the payment service provider, as a copy of the password transformed by SHA-256 is also stored in their servers. The direct password is not stored to minimise the damage of a data leak in the event of a breach in the system.

If the password is correct, the server then responds with a Session Id, a nonce Nb as a challenge to the user and the response to the challenge provided by the user. The user verifies the response to nonce Na by the server, and then responds to the challenge given by the server with Ek(session Id, Nb). If you notice previously, the response to nonce Na was Ek(phone number, Na), but the response to challenge Nb is using session Id instead of the phone number. The reason for this choice is to prevent any replay attacks. Ek itself, is a simple key based symmetric encryption algorithm called MAC (message authentication code).

The server then verifies the response to the challenge given by the user's mobile phone. If the verification is successful, the server then confirms it is the user and then responds with the fingerprint related data, which is encrypted with AES, where the key used to encrypt is derived from the session Id. It is then decrypted on the end user side using the same session Id previously shared with the user. We have now successfully downloaded the fingerprint related data. To prove to the server that the data was indeed shared with the authorised user, we once again share the session Id and password, which is verified by the server, after which the connection and session is closed.

Analysis of the protocol on satisfying conditions

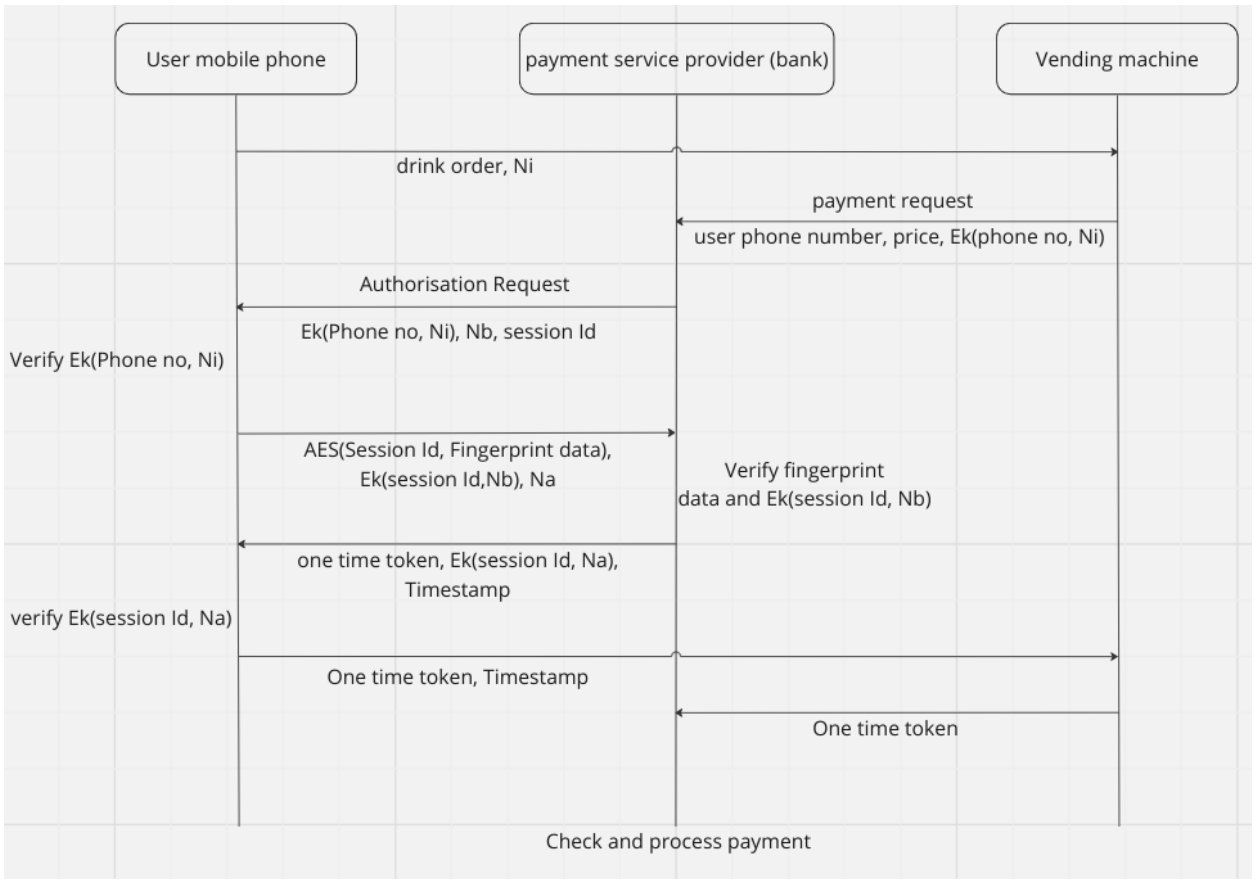
- The server transfers the fingerprint related data to the mobile user only when the server is convinced that the user is the legitimate owner of the fingerprint related data, after the 3 way handshake is established with a session Id that is randomly generated. Even after the fingerprint is shared to the user, the user confirms that the related data has been handed over to the intended person by initiating an acknowledgement.
- The confidentiality of the fingerprint related data is maintained throughout the process, as the data transferred is all encrypted with AES and the key used is also derived from a value that changes regularly (session Id). Even if there is an eavesdropper on the channel, they will not be able to get any information directly by analysing the messages that pass through the channel. The communication also takes place over an https channel, which adds another layer of security to the system.
- We can ensure that the system is safe from D.o.S attacks using a stateful firewall. It ensures that the system is much secure compared to using a stateless firewall that has no context over the protocol and bases the rules only on the port the request is sent to.

Computational and communication costs

- The computational costs are low in terms of verifying the identity of the user. The password is encrypted using a SHA-256 algorithm, while the nonces are fairly computed easily by the mobile system with a simple symmetric encryption algorithm like the MAC (message authentication code). The only computationally heavier algorithm, like the AES is used only for sharing the fingerprint related data.
- The communication costs are average as it uses a secure https channel. However, using stateful firewalls add a considerable computation cost to the system, but a necessary cost because we handle the transfer of fingerprint and identity related data that can be used to impersonate someone, which on the wrong hands can be very harmful to the user.
- Overall, it is a 5 step protocol, which has been optimised to ensure that there are no wasteful communications between the parties involved. Each step is clearly designed to respond to a previous request made and also initiate the next part of the authentication and data transfer process. Hence, it is relatively effective in communication costs.

Payment authorization service

The system allows for the user to purchase a drink by authorising the payment to “Drinks4U” company from the payment service provider using their fingerprint related data, which is either scanned using their device, or downloaded from their payment service provider. Illustrated below is the protocol for carrying out this service.


User mobile phone
payment service provider (bank)
Vending machine
drink order, Ni
payment request
user phone number, price, Ek(phone no, Ni)
Authorisation Request
Ek(Phone no, Ni), Nb, session Id
Verify Ek(Phone no, Ni)
AES(Session Id, Fingerprint data),
Ek(session Id,Nb), Na
Verify fingerprint
data and Ek(session Id, Nb)
one time token, Ek(session Id, Na),
Timestamp
verify Ek(session Id, Na)
One time token, Timestamp
One time token
Check and process payment

This is a 7 step communication protocol that occurs between three different entities in the system. It begins with the user providing a drink order and a unique token (Ni) which is kind of used as a nonce so that the user can later identify the order as their own to the vending machine. The vending machine then takes the price, the phone number and then also processes the nonce using a MAC algorithm, denoted by Ek(Phone no, Ni) and sends it to the payment service provider as a payment request.

This payment request is then processed by the server, which then sends an authorization request to the user, along with a nonce Nb as a challenge to the user’s mobile phone. The request also contains the response Ek(phone no, Ni) which was generated by the vending machine, so that the user can recognize the payment request. This request also comes with a session Id to keep all the keys used for encryption fresh and unpredictable. The session Id is randomly generated.

The user then verifies the authorization request details, after which they provide the server with the fingerprint related data and the response to nonce Nb Ek(session Id, Nb) along with a new challenge nonce Na also. It is worth noting that the fingerprint related data is not directly sent to the server, but rather encrypted with AES, where the key is derived from the session Id previously sent to the user by the server.

The server will then verify the fingerprint and the response to the nonce Nb. If it passes the verification, then it responds with a One time token which can be used to authorise the payment. It also includes the timestamp of when the token was provided. This one time token expires within 10 minutes of being issued to prevent any misuse or delayed exploit of token.

The user verifies the response to nonce Na, which is done to ensure that the OTT (one time token) received from the server is a trusted source. If successful, the token is then sent from the user to the vending machine. Along with the timestamp. Giving the user's mobile the authority to send the token to the vending machine is also important as it provides one last confirmation from the user before authorising and processing the payment.

The one Time token is then passed on to the server from the vending machine and the payment is processed. The vending machine marks off the user as paid based on the phone number from which the user sends the one time token, along with the current context of the session.

Thus, the payment details are then checked off and processed by the payment service provider at the end.

Analysis of the protocol on satisfying conditions

- The mobile user authorises the drink purchase using his fingerprint data, by sending it to the server and encrypting it with AES. The vending machine has no information about the fingerprint, and uses a one time token to verify the payment request. All messages generated have a random element of the one time token which is generated per session, and the session Id which is also randomly generated. This ensures that the messages used for authentication and authorization purposes are fresh, and cannot be replayed. It ensures security against forgery.
- The Expiry of One time token, with a strict time limit of 10 minutes from the time it is issued by the server, ensures that it cannot be misused by the vending machine. It is also worth noting that the one time token expires after being used once by the vending machine, which is discarded by the server. It cannot be replayed and exploited by the vending machine. Hence the name one time token. It is randomly generated and also unpredictable by external users who try to exploit the system. Once the payment is performed by the server, it is then logged into the user's purchase records, where they

can view the payment receipt online. This takes place in the check and process payment step at the end.

- The communication is kept to a minimum so that only the necessary information is shared between the entities. The 7 step communication protocol can be further split into three parts.

    - The order and the payment request. (Step 1 and 2)
    - The three way handshake of payment authorization. (Step 3,4 and 5)
    - The payment processing. (step 6 and 7)

  Therefore, we can agree that the communication process is highly optimised, with no unnecessary communication involved. The computation power is also well accounted for, as the major computational steps are only required during the authorisation step, where we use AES. Other than that, the nonce challenge computation using algorithms like MAC can be handled fairly easily by the mobile users and the vending machine. The use of session Ids that are randomly generated will ensure that they are secure.

Computational and communication cost

- We will use stateless firewalls here to prevent attacks, which also computationally require less power. They do not have to know the entire context of the protocol, as it will be very computationally heavy. As there will be a lot of payments to process, this tradeoff is slightly advantageous to ensure that the system is realistically possible. In terms of complexity, if we have more computational power available, we could opt for stateful firewalls here.
- There is minimal communication between the entities in the system. With the previous explanations given for the protocols, it can be concluded that it is optimised for the given scenario.
- The use of MAC and AES ensures that the computations involved can be performed by the mobile device that meets the minimal requirements.
- The maximum load handled will be the server in this scenario that processes multiple payments. Maintaining a hash table of the authorised one time tokens along with the accounts associated with it will allow it to access the details and process payment in O(1) time when it comes to managing OTT. The main bottleneck is in managing all sessions, which handle a lot of verification from the server-side. We can limit the number of sessions that come from a particular user in a set amount of time to avoid any threat actor to exploit D.O.S attacks, or to simply overload the server.
- Overall, it is a 7 step protocol.