TECHNOLOGY PARK MALAYSIA

CT124-3-3-BCD

BLOCKCHAIN DEVELOPMENT

UC3F2108CS

HAND IN DATE: 29 NOVEMBER 2021

WEIGHTAGE: 60%

---

|  |  |  |  |
|---|---|---|---|
| STUDENT NAME & ID | : | LEE ZHE JIAN | TP051128 |
|  |  | HENRY KHOO SHIEN CHEN | TP051334 |
| INTAKE CODE | : | UC3F2108CS |  |
| MODULE CODE | : | CT124-3-3-BCD |  |
| ASSIGNMENT | : | GROUP ASSIGNMENT |  |
| LECTURER | : | MR. LEE KIM KEONG |  |

# TABLE OF CONTENTS

**TABLE OF CONTENTS** .......... 2

1.0 Introduction of Blockchain System for Car Rental Management .......... 4

2.0 Findings .......... 6

2.1 System Architecture .......... 6

2.2 Data Management .......... 10

2.2.1 User Login Credential .......... 10

2.2.2 User Record .......... 11

2.2.3 Booking Record .......... 11

3.0 Solution model of blockchain system in car rental business operation .......... 14

4.0 Solution design and implementation of blockchain algorithms .......... 22

4.1 Hashing Algorithm .......... 22

4.1.1 Overview .......... 22

4.1.2 Benefits of hashing .......... 22

4.1.3 Types of hashing .......... 23

4.1.4 Source code and implementation of hashing in business .......... 24

4.2 Cryptography Algorithm .......... 30

4.2.1 Overview .......... 30

4.2.2 Benefits of cryptography .......... 31

4.2.3 Types of cryptography .......... 32

4.2.4 Source code and implementation of cryptography in business .......... 33

4.3 Digital Signature .......... 39

4.3.1 Overview .......... 39

4.3.2 Benefits of digital signature .......... 40

4.3.3 Source code and implementation of digital signature in business .......... 41

5.0 Evaluation of developed solution .......... 45

5.1 Security .......... 45

5.2 Transparency ........................................................................................................ 46

5.3 Accessibility ........................................................................................................ 47

6.0 Conclusion ........................................................................................................ 48

Reference ........................................................................................................ 49

## 1.0 Introduction of Blockchain System for Car Rental Management

Blockchain can be broken down into chain of blocks, which is list of records that are linked together with hashes of previous block, timestamp and the data represented as Merkle tree. The blockchain technology has characteristics of decentralization, distributed and public ledger that cannot be altered. The term blockchain has been gaining popularity from Bitcoin, work of pseudonym named Satoshi Nakamoto. The technology is originated by the work of three authors Dave Bayer, Stuart Haber, and W. Scott Stornetta on their work about improving the reliability and efficiency of digital timestamping. There are two proposed solutions mentioned in their study, linking of hashed documents into linear list and randomized signature chosen from random validators. The solutions are implemented in Bitcoin by using hashes and digital signature with public ledger that created the fundamental characteristics of blockchain.

The integration of blockchain technologies have been emerging over the years where the use cases are implemented across various sectors. The blockchain technology have started emerging form the cryptocurrencies to financial services like banking and lending services. As the technology becoming more mature, blockchain creates a rippling effect globally from supply chain, healthcare, retail, and automotive industries. The promise of transparency, trust, and security in recording the transaction have met the demand to counter cybersecurity risks in the network. The blockchain technology is expected and on track for delivery around US $ 1.76 trillion to the economy by increasing the efficiency, building customer loyalty and trust between the customers, business partners and the management staff (Davies, 2020).

According to PwC (2020), the blockchain is capable for fulfilling the needs and demand to add value to business as an effective solution when these situations are applicable on these situations. The blockchain technology can allow multiple parties from cross-functional teams to view the data for common information and update the recorded data. High confidence and trust is required on the validity of the recorded data. Intermediaries or third-party involvement can be removed to save cost and time, while reducing the complexity on system processes. The Interactions between different business parties are time sensitive and have to be dependent on each other. The proposed system of the car rental management system is suitable for blockchain technology as these requirements are essential and needed for the system. The following section will discuss about the different types of blockchain technology and the chosen ones for as discussed in the solution design and implementation sections in detail.

The public and private blockchain have many similarities on their characteristics. They are both designed as decentralized peer-to-peer network that shared an append only ledger with signed digital signature. Consensus protocol is achieved in both blockchain types as they are maintained in sync with all other replicas. The immutability of the ledger provides guarantees, and it is tamper-proof even with the existence of malicious nodes in the network. The difference between the public and private blockchain are discussed below.

The primary distinct difference between the public and private blockchain is determined by the accessibility of the participants in the network, by the maintenance of the shared ledger and the execution of the consensus protocol. The public blockchain is open and free for any participants in the network to join. The Bitcoin cryptocurrency is the most well-known public blockchain created. The public blockchain largest setbacks is the amount of computational power required for maintaining the public ledger in large scale. The high computational power and resource intensive for solving the cryptographic problems that is complex uses the consensus mechanism called Proof-of-Work (PoW). The lack of privacy in the public blockchain create more loophole for security aspects as the public ledger is openly available anyone.

The private blockchain network in comparison will require invitation by the network administrator with set of rules applied to the participants in the network. A permissioned network is commonly implemented in the private blockchain with access control and invitation to join into the network. The added security mechanism with authorization access are more comprehensive and designed for security against potential malicious attackers. The Hyperledger Fabric from Linux Foundation is a permissioned blockchain that is developed and designed for enterprise use cases. The digital identity stressed in the private blockchain will become the fundamental on solving data sensitive use cases like transactions in financial industry, data exchange in healthcare between the patients, and traceability in the supply chain.

The proposed blockchain type for the car rental management is the hybrid blockchain system that has both private and public blockchain characteristics. The design of the hybrid approach is to limit the disadvantages. The proposed system allows the ledger to be viewed by the selected actors allowed in the system, while only the management staff with granted permission can verify and approve the transactions to be added into the ledger.

## 2.0 Findings

Based on the analysis of business use case of blockchain in car rental industry, the system architecture and data management of the proposed system is based on the assumptions supported by the business use case of automotive industry.

### 2.1 System Architecture

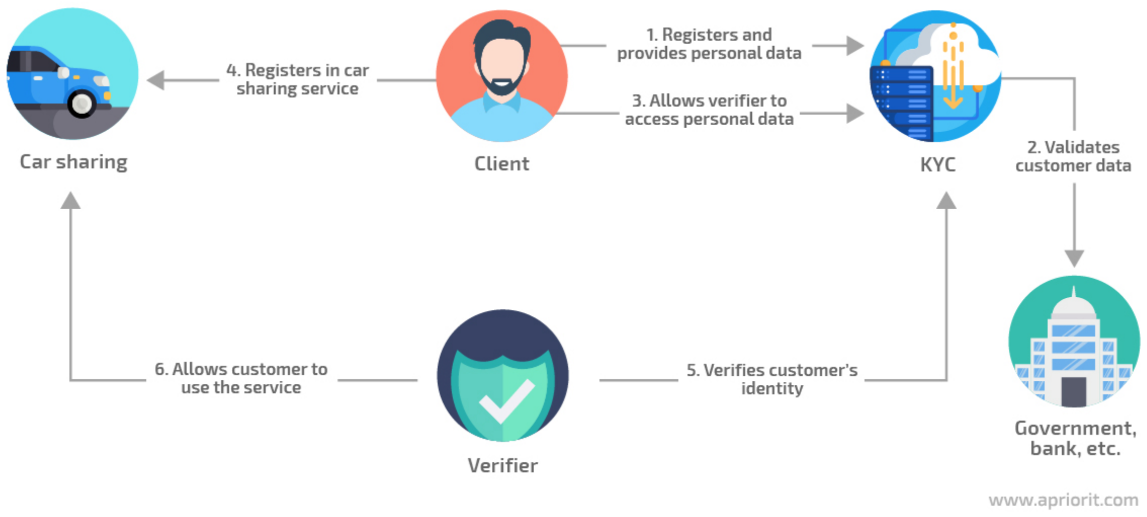

Car sharing
4. Registers in car sharing service
Client
1. Registers and provides personal data
3. Allows verifier to access personal data
KYC
2. Validates customer data
Government, bank, etc.
6. Allows customer to use the service
Verifier
5. Verifies customer's identity
www.apriorit.com

*Figure 1: Existing Car Rental System Flow*

Based on the findings from existing car rental system in the industry, the proposed blockchain car rental system is designed and developed with a similar flow with these existing systems. In general, the car rental system will be performed in the following stages. Firstly, clients will provide their personal information to be registered into the system. This information will be used by the verifier or management to verify the customer's identity whenever required. For example, when client registers for a car rental service, the client will allow the verifier to access their personal data and verify their identity to ensure the legitimacy of the client. To do so, a form of smart contract are required to be created and accepted by all parties involved.

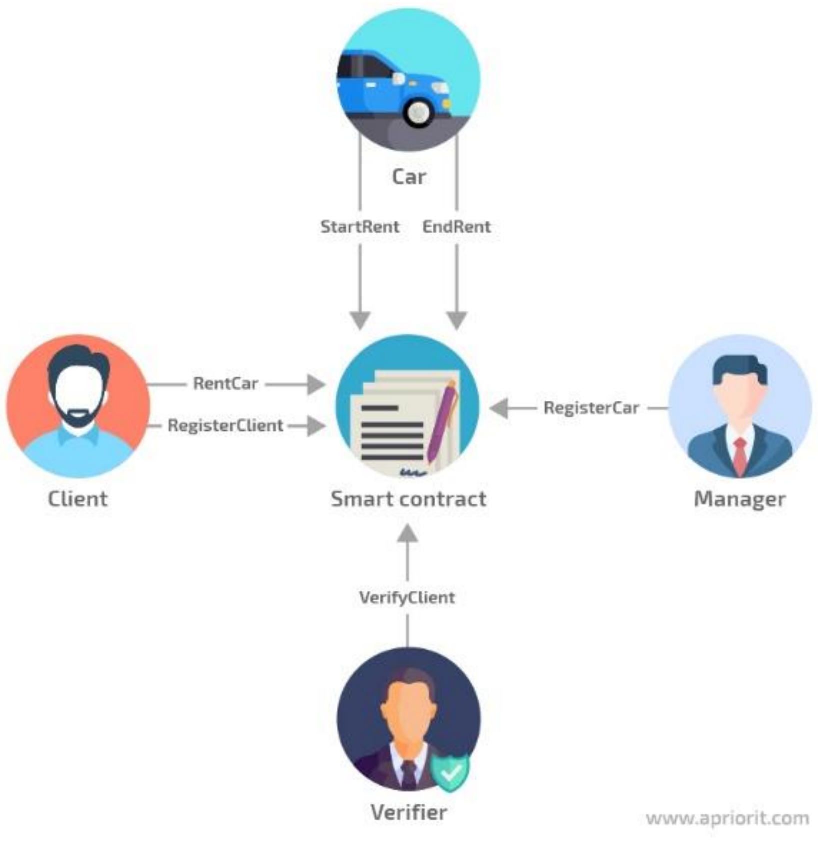


*Figure 2: Existing Car Rental System Process*

In general, existing car rental systems incorporate some form of smart contract to handle the car rental process. Clients who are registered into the system will provide details to rent a car, in which the data and time as well as any other relevant information will be included based on the car model. The car information will be handled by the manager. After renting a car, a verifier will verify and authenticate the user's legitimacy to use the renting service, in which the contract will then be either approved or denied. The contract will involve confirmation from all parties to ensure the legitimacy of the transaction, as such the data used for the smart contract is highly sensitive and should be secured

In the case of the proposed car rental system, blockchain technology will be implemented for the car rental process, in which the booking record or contract will be applied with blockchain. Blockchain can provides a much more secure handling of information and data for the parties involved. The management of the car rental business can have better visibility of the vehicles as every booking record will be stored in a digital ledger. As such, management can easily access the necessary information to track the vehicle and performed management tasks such as repairs, mileage, fuel changing and maintenance without having the risk of fraud such as mileage fraud or issue such as rental terms disputes. Implementation of blockchain technology also ensure that the rental records are not modified, thus preventing malicious activities.

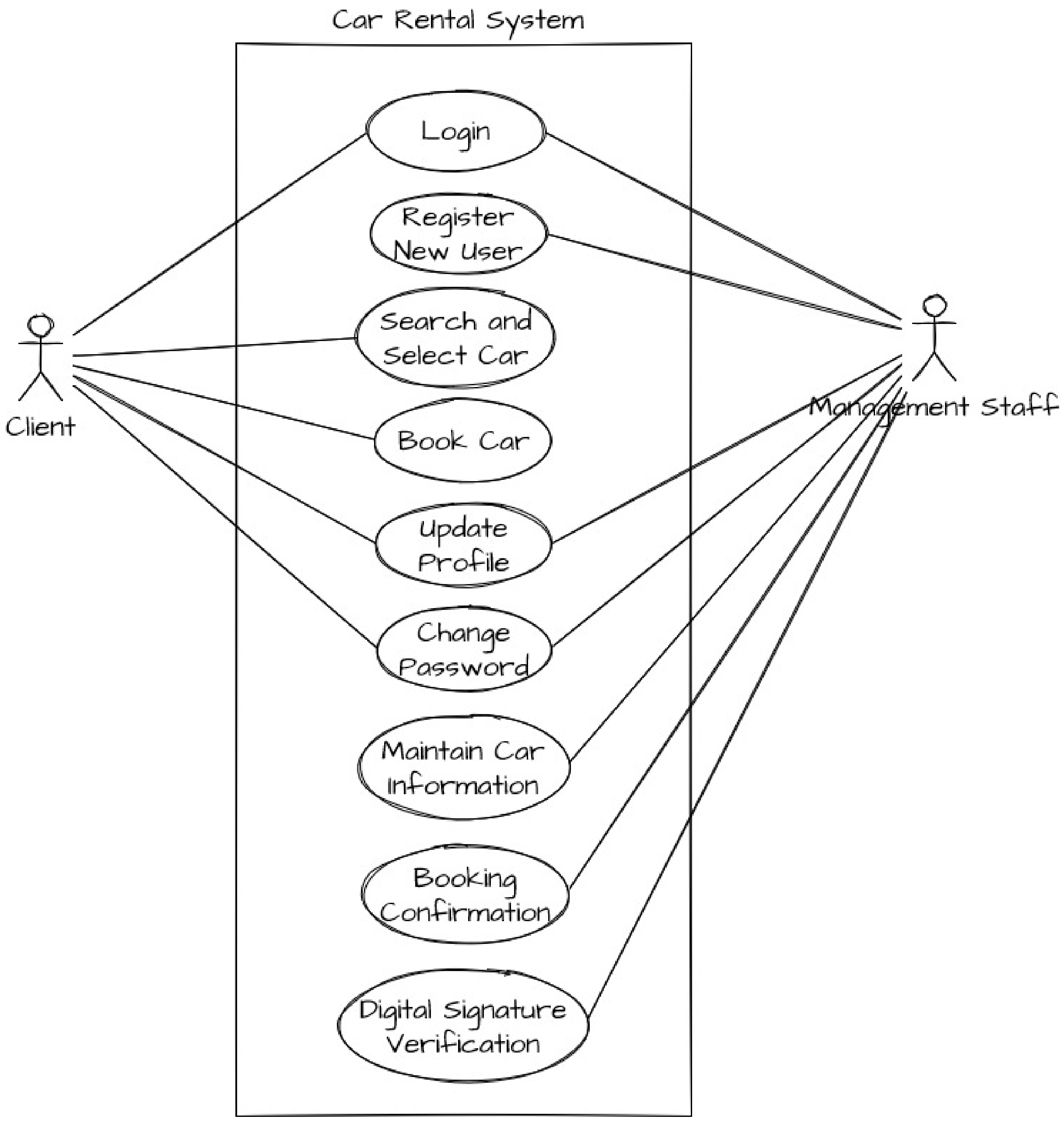


*Figure 3: Use Case for Proposed Car Rental System*

The main users of the proposed car rental system include the admins, management staffs and clients. Management staffs are in charge of the registration of new users into the system. Every relevant information will first be filled out by the management staff to be registered as a user.

An identification (ID) and digital signature will be generated for each user upon registration into the system. The ID will be used internally by the system to identify the user credentials and the password, while the digital signature is the unique electronic fingerprint associated with each user. The password for the login credentials will be initialized using the user's username, in which they can update it after the first time they logged on to the system. The password will be hashed using hashing algorithm to encrypt the password and prevent unauthorized access.

Once registered in the system, the registered users can login and access the relevant and appropriate functions based on their role in the system. For the clients, they are able to search and look up the car records in the system and obtain the relevant car information. They can then book the vehicle that they want to rent by filling in relevant information and selecting the booking start and end date. After confirming their booking, their unique digital signature will be used to sign the booking record. On top of that, the clients can also update their own profile and change their password, in which the updated records will be changed and reflected on the relevant data in the system.

On the other hand, for the management staffs, they are able to create new user and add their record into the system. They can also edit any user's record but not their password. They also can update their own profile and change their password. As for the vehicle's information, the management staffs have the authority to maintain them. Most importantly, the management staffs have the authority to view the booking records and verify the digital signature based on the user's digital signature to ensure the authenticity and integrity of the booking data. They can then proceed to either approve or deny the client's request to book the vehicle.

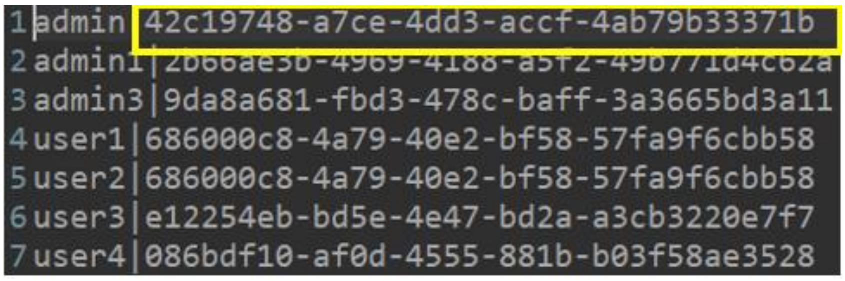
On top of that, to ensure the security as well as verification and traceability of the records within the system, blockchain algorithms are implemented in the system. The user's passwords are hashed and store, thus no unauthorized users can access another user's password. For the booking information, it is cryptographed to hide the sensitive information of the booking data. The confirmed booking will then be added into the ledger using blockchain to ensure the traceability.

## 2.2 Data Management

Different sources for both sensitive and non-sensitive data were identified based on analyzation of car rental industry. The data source was based on existing fleet management system and car rental system used in the market. Based on the analyzation, the data to be used in the proposed blockchain car rental system includes user login credentials and data, vehicle data and booking records.

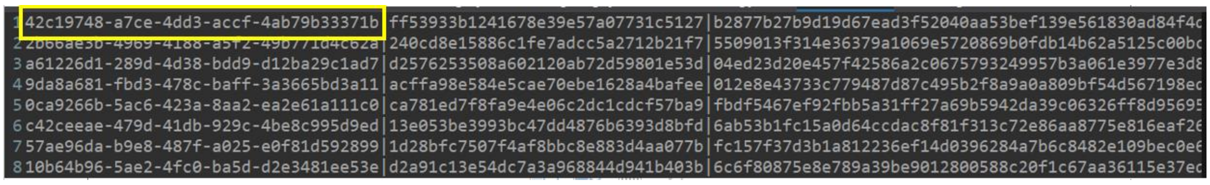
### 2.2.1 User Login Credential

The user login credentials are consisting of the user's ID and secret key, which is used to hash the password and map it to the actual password input. Due to the one-way nature of hashing, the hashed data is irreversible, thus it is suitable to be used for authentication process such as login functions. For the login function in the proposed system, the password will be hashed using a hashing algorithm, in which it will output the hashed password as well as a salt value that acts as additional layer of security for the hashing process. These values are stored as following:



```
admin|42c19748-a7ce-4dd3-accf-4ab79b33371b
admin1|2b66ae3b-4969-4188-a5f2-49b771d4c62a
admin3|9da8a681-fbd3-478c-baff-3a3665bd3a11
user1|686000c8-4a79-40e2-bf58-57fa9f6cbb58
user2|686000c8-4a79-40e2-bf58-57fa9f6cbb58
user3|e12254eb-bd5e-4e47-bd2a-a3cb3220e7f7
user4|086bdf10-af0d-4555-881b-b03f58ae3528
```

*Figure 4: Login Username and ID*

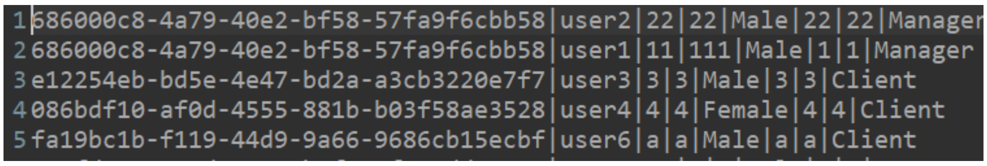


```
42c19748-a7ce-4dd3-accf-4ab79b33371b|ff53933b1241678e39e57a07731c5127|b2877b27b9d19d67ead3f52040aa53bef139e561830ad84f4c
2b66ae3b-4969-4188-a5f2-49b771d4c62a|240cd8e15886c1fe7adcc5a2712b21f7|5509013f314e36379a1069e5720869b0fdb14b62a5125c00bc
a61226d1-289d-4d38-bdd9-d12ba29c1ad7|d2576253508a602120ab72d59801e53d|04ed23d20e457f42586a2c0675793249957b3a061e3977e3d8
9da8a681-fbd3-478c-baff-3a3665bd3a11|acffa98e584e5cae70ebe1628a4bafee|012e8e43733c779487d87c495b2f8a9a0a809bf54d567198ec
0ca9266b-5ac6-423a-8aa2-ea2e61a111c0|ca781ed7f8fa9e4e06c2dc1cdcf57ba9|fbdf5467ef92fbb5a31ff27a69b5942da39c06326ff8d95695
c42ceeae-479d-41db-929c-4be8c995d9ed|13e053be3993bc47dd4876b6393d8bfd|6ab53b1fc15a0d64ccdac8f81f313c72e86aa8775e816eaf26
57ae96da-b9e8-487f-a025-e0f81d592899|1d28bfc7507f4af8bbc8e883d4aa077b|fc157f37d3b1a812236ef14d0396284a7b6c8482e109bec0e6
10b64b96-5ae2-4fc0-ba5d-d2e3481ee53e|d2a91c13e54dc7a3a968844d941b403b|6c6f80875e8e789a39be9012800588c20f1c67aa36115e37ec
```

*Figure 5: ID and Secret (Salt and Hashed Password)*

The login credentials will store the user's username and an automatically generated ID, which is used for referencing with the secret file that contains the user's hashed password and salt. When logging in, the user's input will be match with their username and ID to find out the relevant salt and hashed password based on their ID, in which the obtained result will be used to authenticate user's password input.

### 2.2.2 User Record

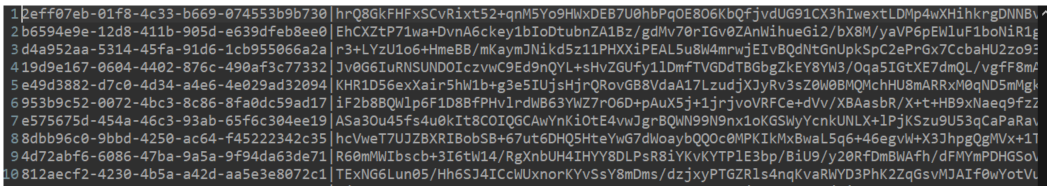


```
686000c8-4a79-40e2-bf58-57fa9f6cbb58|user2|22|22|Male|22|22|Manager
686000c8-4a79-40e2-bf58-57fa9f6cbb58|user1|11|111|Male|1|1|Manager
e12254eb-bd5e-4e47-bd2a-a3cb3220e7f7|user3|3|3|Male|3|3|Client
086bdf10-af0d-4555-881b-b03f58ae3528|user4|4|4|Female|4|4|Client
fa19bc1b-f119-44d9-9a66-9686cb15ecbf|user6|a|a|Male|a|a|Client
```

*Figure 6: User Record Stored*

User records will store the information regarding the users within the car rental institute. For the proposed car rental system, only the relevant data that contain attributes necessary for the car rental process is stored. The attributes of user record include userID, username, firstName, lastName, gender, email, phoneNumber and position. The userID refers to the randomly generated string of number that uniquely identifies the user within the system. The ID is not editable and will only be used internally by the system to perform certain functions that require user identification. The ID is used as a linkage and referrer to the user's login credentials and secret key to login into the system. The username contains the unique username that is used by user to log in into the system. The firstName and lastName field are used for storing the name of the users. Gender field is used for storing information regarding the user's gender identity. Email field and phoneNumber field are used to store the user's contacts, which are their email and phone number data respectively. Lastly, the position field will store the user's roles within the system such as Admin, Manager or Client. These roles will be used to determine the user's accessibility throughout the system.

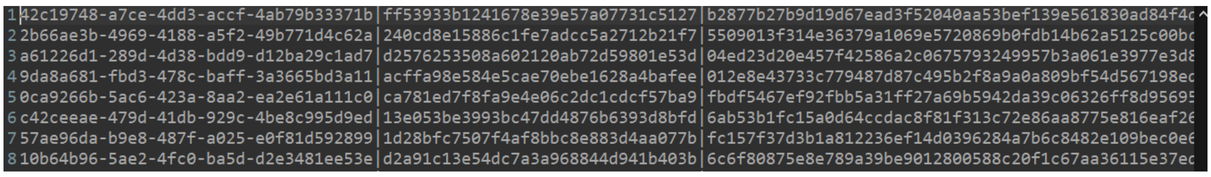
### 2.2.3 Booking Record



```
2eff07eb-01f8-4c33-b669-074553b9b730|hrQ8GkFHFxSCvRixt52+qnM5Yo9HWxDEB7U0hbPqOE8O6KbQfjvdUG91CX3hIwextLDMp4wXHihkrgDNNBv
b6594e9e-12d8-411b-905d-e639dfeb8ee0|EhCXZtP71wa+DvnA6ckey1bIoDtubnZA1Bz/gdMv70rIGv0ZAnWihueGi2/bX8M/yaVP6pEWluF1boNiR1g
d4a952aa-5314-45fa-91d6-1cb955066a2a|r3+LYzU1o6+HmeBB/mKaymJNikd5z11PHXXiPEAL5u8W4mrwjEIvBQdNtGnUpkSpC2ePrGx7CcbaHU2zo93
19d9e167-0604-4402-876c-490af3c77332|Jv0G6IuRNSUNDOIczvwC9Ed9nQYL+sHvZGUfy1lDmfTVGDdTBGbgZkEY8YW3/Oqa5IGtXE7dmQL/vgfF8mA
e49d3882-d7c0-4d34-a4e6-4e029ad32094|KHR1D56exXair5hW1b+g3e5IUjsHjrQRovGB8VdaA17LzudjXJyRv3sZ0W0BMQMchHU8mARRxM0qND5mMgk
953b9c52-0072-4bc3-8c86-8fa0dc59ad17|iF2b8BQWlp6F1D8BfPHvlrdWB63YWZ7rO6D+pAuX5j+1jrjvoVRFCe+dVv/XBAasbR/X+t+HB9xNaeq9fzZ
e575675d-454a-46c3-93ab-65f6c304ee19|ASa3Ou45fs4u0kIt8COIQGCAwYnKiOtE4vwJgrBQWN99N9nx1oKGSWyYcnkUNLX+lPjKSzu9U53qCaPaRav
8dbb96c0-9bbd-4250-ac64-f45222342c35|hcVweT7UJZBXRIBobSB+67ut6DHQ5HteYwG7dWoaybQQOc0MPKIkMxBwaL5q6+46egvW+X3JhpgQgMVx+1T
4d72abf6-6086-47ba-9a5a-9f94da63de71|R60mMWIbscb+3I6tW14/RgXnbUH4IHYY8DLPsR8iYKvKYTPlE3bp/BiU9/y20RfDmBWAfh/dFMYmPDHGSoV
812aecf2-4230-4b5a-a42d-aa5e3e8072c1|TExNG6Lun05/Hh6SJ4ICcWUxnorKYvSsY8mDms/dzjxyPTGZRls4nqKvaRWYD3PhK2ZqGsvMJAIf0wYotVu
```

*Figure 7: Cryptographed Booking Record*

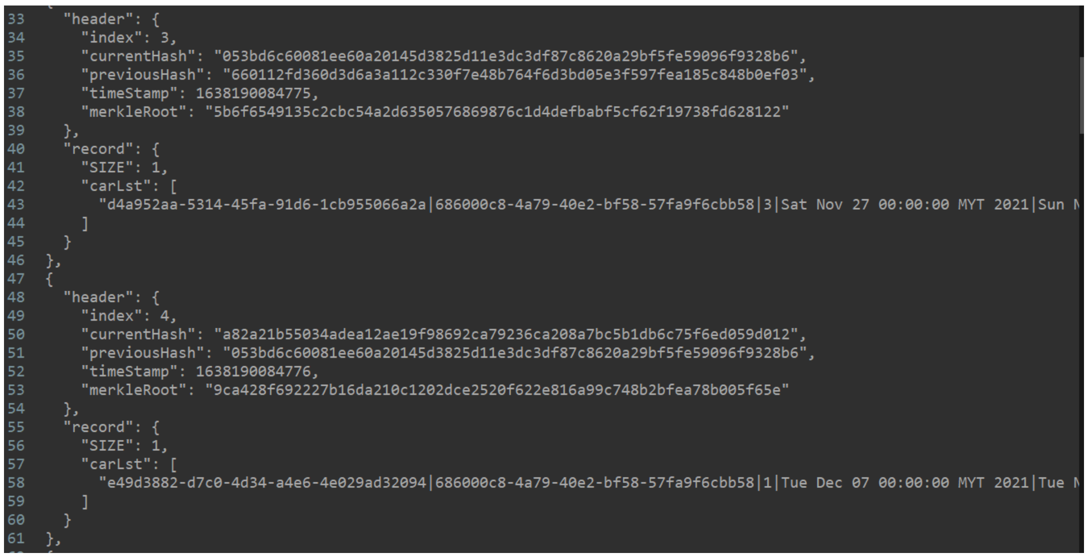
Booking record will store the data in relation to the arrangement of car rental. The attributes include the bookingID, userID, carID, startDate and endDate. The bookingID is a unique ID that is generated for each booking made by clients, in which it identifies the specific booking request. The userID is obtained from the user record for the user that made the booking request. The carID attribute is used for identifying the vehicle selected in the booking request. The startDate and endDate attributes are used for storing the booking date and time for starting date and ending date respectively. For security purposes, the data for these attributes except for the bookingID will all be cryptographed to secure all the sensitive information regarding the booking request made by the clients.



```
42c19748-a7ce-4dd3-accf-4ab79b33371b|ff53933b1241678e39e57a07731c5127|b2877b27b9d19d67ead3f52040aa53bef139e561830ad84f4
2b66ae3b-4969-4188-a5f2-49b771d4c62a|240cd8e15886c1fe7adcc5a2712b21f7|5509013f314e36379a1069e5720869b0fdb14b62a5125c00b
a61226d1-289d-4d38-bdd9-d12ba29c1ad7|d2576253508a602120ab72d59801e53d|04ed23d20e457f42586a2c0675793249957b3a061e3977e3d
9da8a681-fbd3-478c-baff-3a3665bd3a11|acffa98e584e5cae70ebe1628a4bafee|012e8e43733c779487d87c495b2f8a9a0a809bf54d567198e
0ca9266b-5ac6-423a-8aa2-ea2e61a111c0|ca781ed7f8fa9e4e06c2dc1cdcf57ba9|fbdf5467ef92fbb5a31ff27a69b5942da39c06326ff8d9569
c42ceeae-479d-41db-929c-4be8c995d9ed|13e053be3993bc47dd4876b6393d8bfd|6ab53b1fc15a0d64ccdac8f81f313c72e86aa8775e816eaf2
57ae96da-b9e8-487f-a025-e0f81d592899|1d28bfc7507f4af8bbc8e883d4aa077b|fc157f37d3b1a812236ef14d0396284a7b6c8482e109bec0e
10b64b96-5ae2-4fc0-ba5d-d2e3481ee53e|d2a91c13e54dc7a3a968844d941b403b|6c6f80875e8e789a39be9012800588c20f1c67aa36115e37e
```

*Figure 8: Digital Signature of Booking Request*

After the booking is made and recorded, it will be stored and used by management for further confirmation and verification. The verification process will be performed using the digital signature generated when the booking was made by the client. The digital signature will ensure that the booking details are not modified or altered from in between the moment it was booked by the clients to the moment it was checked by the management. This is to make sure that the client confirms their booking request and has provided a signature for it.



```
  "header": {
    "index": 3,
    "currentHash": "053bd6c60081ee60a20145d3825d11e3dc3df87c8620a29bf5fe59096f9328b6",
    "previousHash": "660112fd360d3d6a3a112c330f7e48b764f6d3bd05e3f597fea185c848b0ef03",
    "timeStamp": 1638190084775,
    "merkleRoot": "5b6f6549135c2cbc54a2d6350576869876c1d4defbabf5cf62f19738fd628122"
  },
  "record": {
    "SIZE": 1,
    "carLst": [
      "d4a952aa-5314-45fa-91d6-1cb955066a2a|686000c8-4a79-40e2-bf58-57fa9f6cbb58|3|Sat Nov 27 00:00:00 MYT 2021|Sun
    ]
  }
},
{
  "header": {
    "index": 4,
    "currentHash": "a82a21b55034adea12ae19f98692ca79236ca208a7bc5b1db6c75f6ed059d012",
    "previousHash": "053bd6c60081ee60a20145d3825d11e3dc3df87c8620a29bf5fe59096f9328b6",
    "timeStamp": 1638190084776,
    "merkleRoot": "9ca428f692227b16da210c1202dce2520f622e816a99c748b2bfea78b005f65e"
  },
  "record": {
    "SIZE": 1,
    "carLst": [
      "e49d3882-d7c0-4d34-a4e6-4e029ad32094|686000c8-4a79-40e2-bf58-57fa9f6cbb58|1|Tue Dec 07 00:00:00 MYT 2021|Tue
    ]
  }
},
```

*Figure 9: Ledger of Approved Booking Request Blockchain*

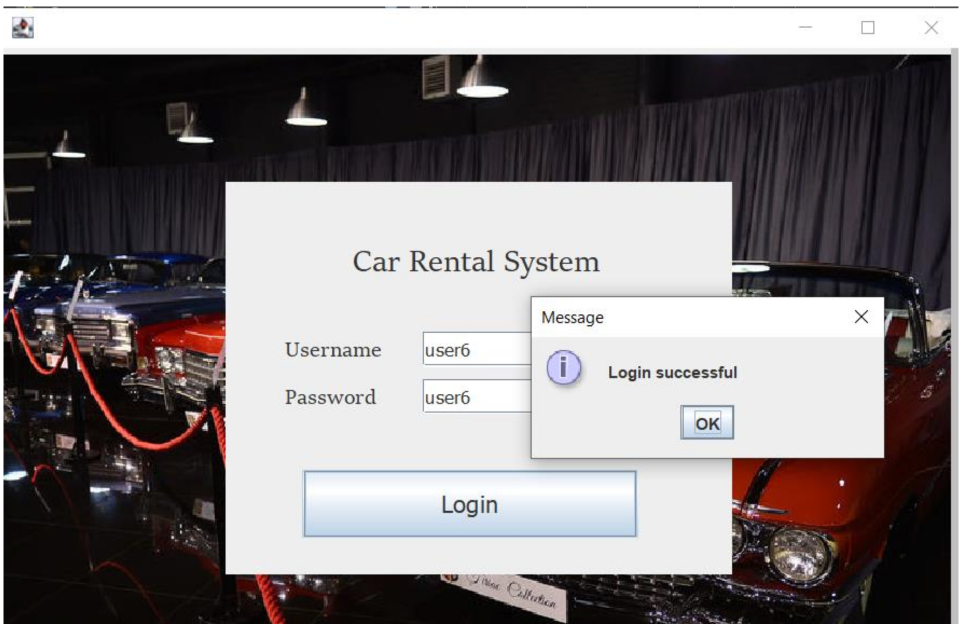
After the booking has been approved, it will be stored in the ledger file as one of the blocks in the blockchain. The blockchain contains each and every booking request that is approved by the management. Using blockchain to store the approved booking records can ensure that the booking made is immutable and cannot be further modified since it was already confirmed. It also provides traceability for the management of the car rental institute to manage their vehicle properly.

## 3.0 Solution model of blockchain system in car rental business operation

The blockchain technology is implemented in the car rental system developed based on the proposal documentation and the findings of the report. This section will walk through the car rental system by demonstrating the solution model which is broken down into the functionality and application of blockchain technology in each page.

There are three actors in the system which are the management staff, client, and administrator. The client and management staff has access restriction based on their roles assigned, while the administrator has full access privilege. Clients have access to book cars by selecting the start and end data. The booking status will be pending for verification of the booking information. Management staff have access to verify the pending review booking and either accepting or declining the booking. The manager will show the validation process and approval of the booking. The clients and managers can modify their personal information like password and other user details. The demonstration below will provide the functionality and explanation on each page with attachments of screenshots provided below.

**Client - Login**


Car Rental System

Username: user6

Password: user6

Message

Login successful

*Figure 10: Login page in car rental system (Client login).*

In the login page, the client is required to enter the username and password, and the system will perform validation to identify the user credentials in the system. The user is directed to main menu page.

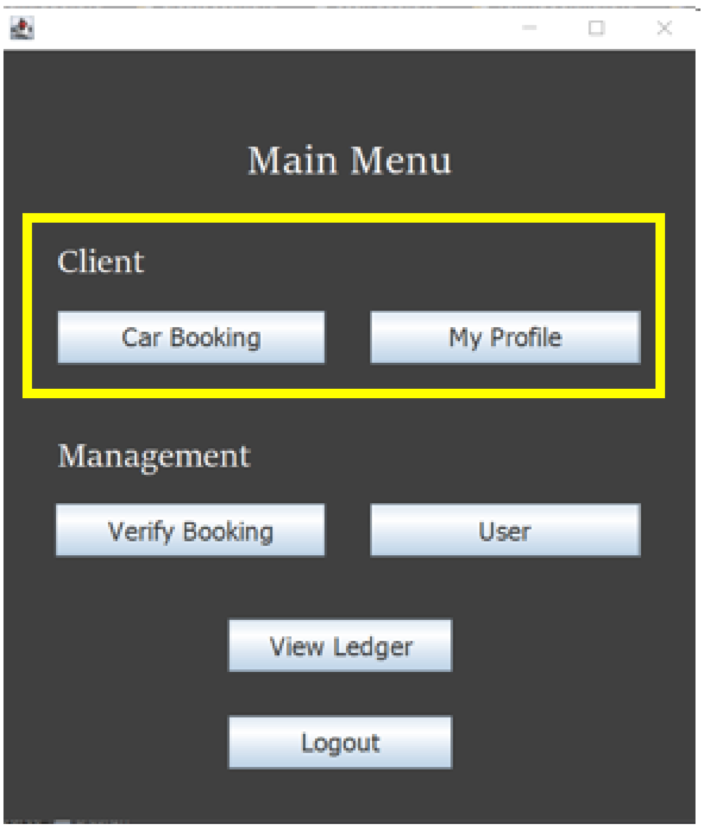
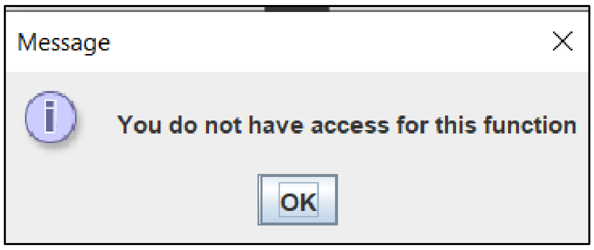


*Figure 11: Main menu page in car rental system s(Client's access).*

The client is granted access for the car booking and my profile functions, other features are restricted to the management staff. The client can then select car booking button and he will be directed to the car selection page.

**Client - Access Restriction**



You do not have access for this function

*Figure 12: Access restriction in car rental system.*

The system will display an alert message showing no access granted when the client attempts to access the verify booking function and user function.

**Client - Car selection and booking**

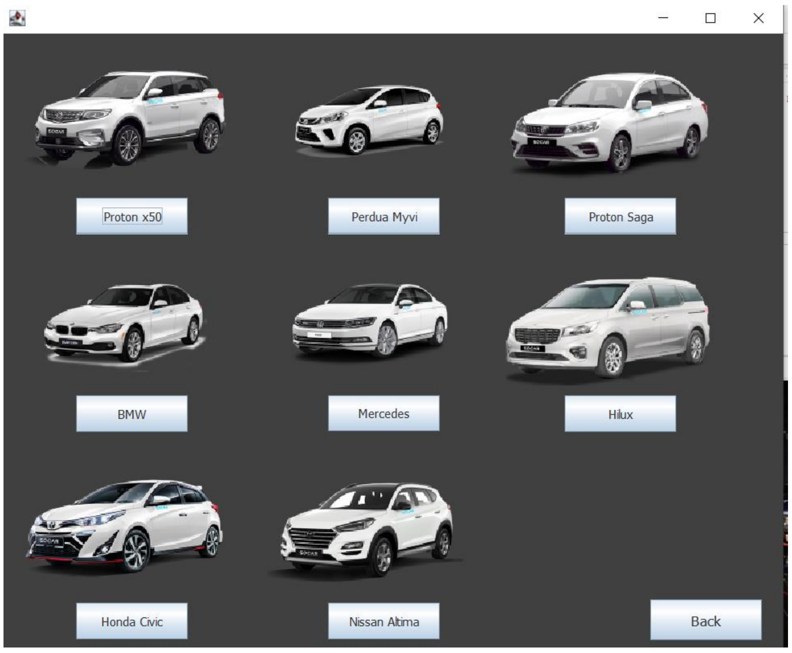
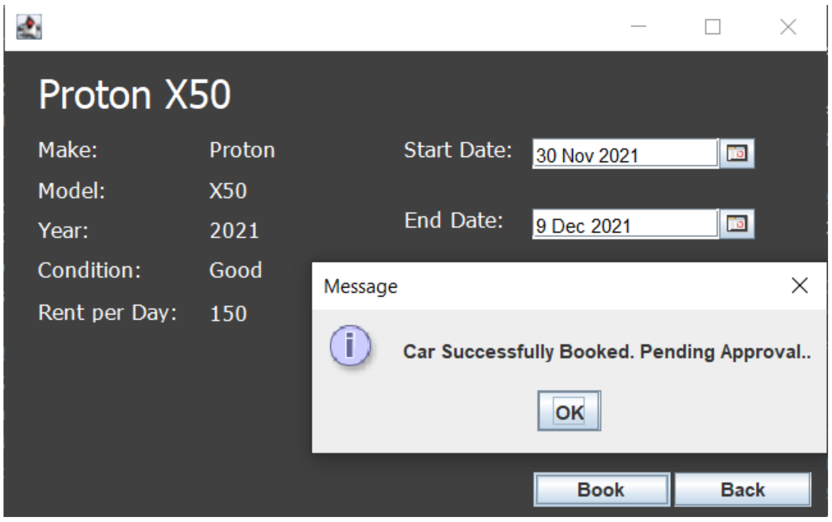


*Figure 13: Car selection page in car rental system.*

The client can then select the desired car in the car selection page, and he will be redirected to the car booking page.



Proton X50

Make: Proton
Model: X50
Year: 2021
Condition: Good
Rent per Day: 150

Start Date: 30 Nov 2021
End Date: 9 Dec 2021

Message
Car Successfully Booked. Pending Approval..

*Figure 14: Car booking page in car rental system.*

In the car booking page, the client can view the make, model, year, condition, rent per day and the car name displayed at the left side of the page. The user can select the date ranging starting from today onwards. The date verification will check if the end date is greater than start date. After the client entered the corresponding dates, the system will prompt the booking is successful and pending approval from the management team.

### Client - Editing user details and changing password

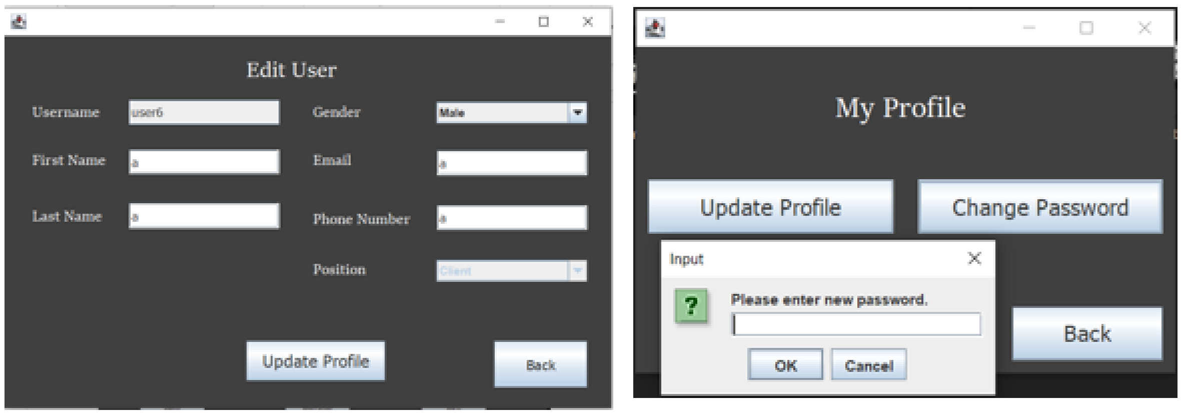


*Figure 15: Editing user information page in car rental system.*

The client can edit their user information and password in my profile page The system will update the user database on the updated client’s details.

### Management Staff - Login

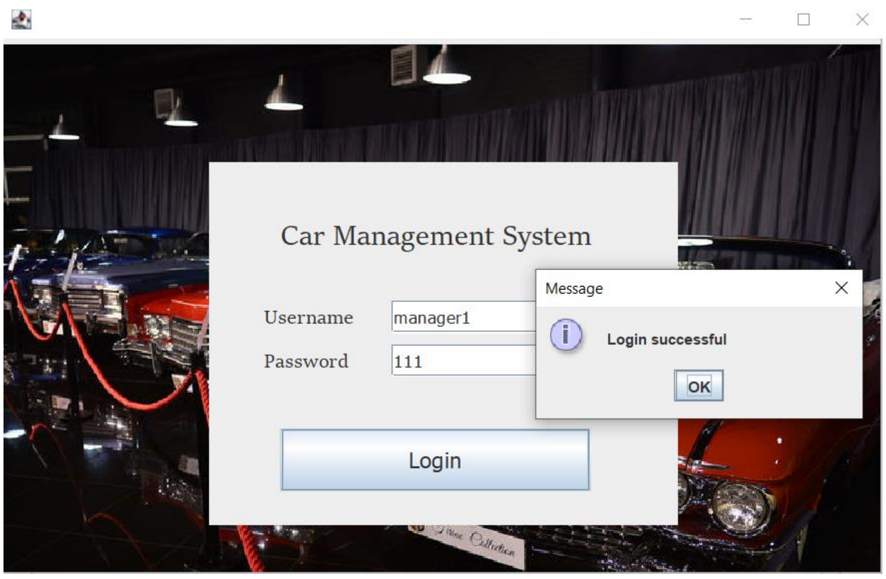


*Figure 16: Login page in car rental system (Management staff login).*

In this section of the demonstration, the management staff will proceed to the login page. In this example, the management staff called manager1 will login into the system.

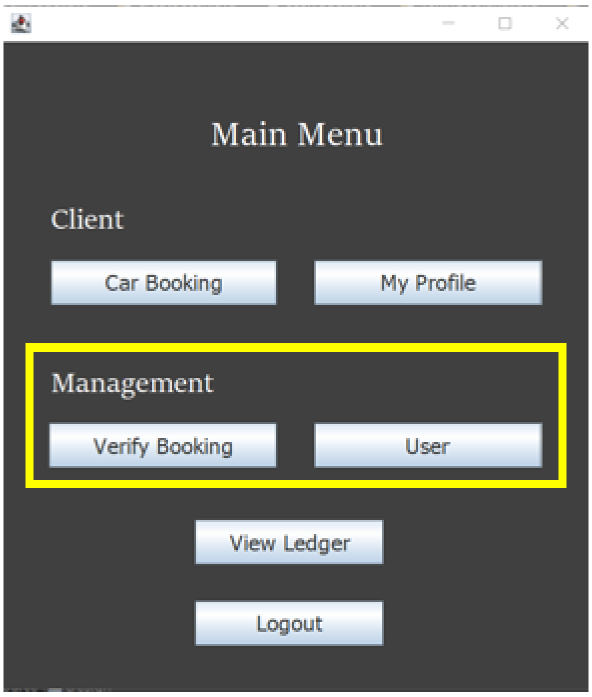
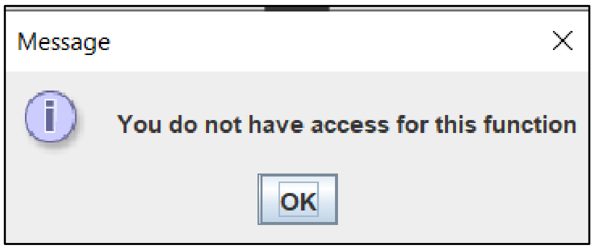


*Figure 17: Main menu page in car rental system (Manager's access).*

The manager can access the verify booking, modifying user details and creating new users feature in the system.

**Management Staff - Access restriction**

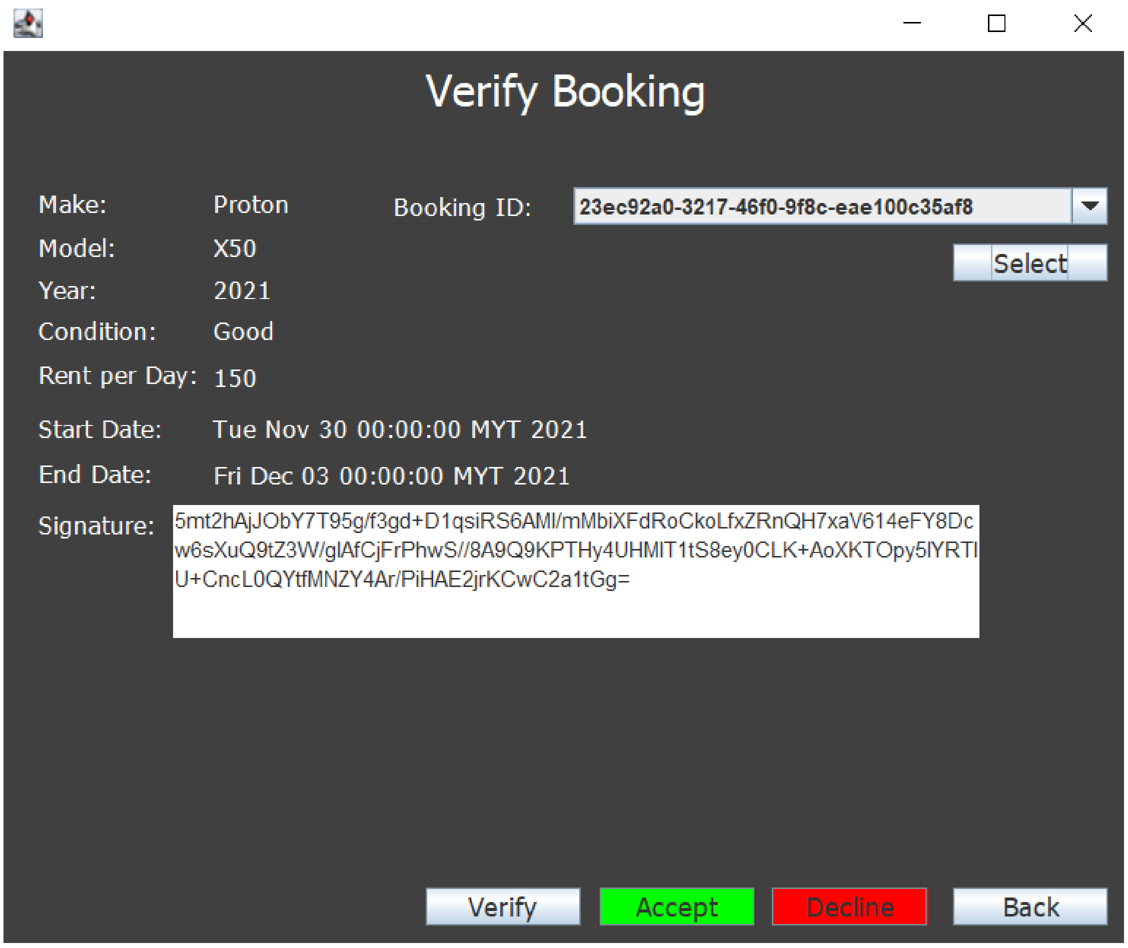


You do not have access for this function

*Figure 18: Access restriction in car rental system.*

The system will display similar alert message showing no access granted when the management staff attempt to access the car booking and my profile functions.

**Management Staff: Booking verification and validation**

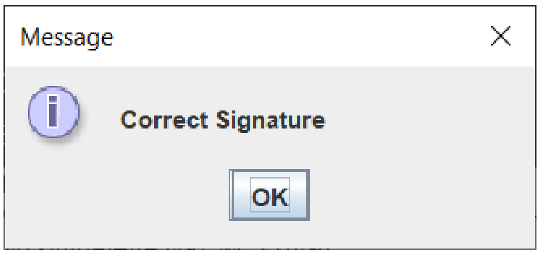

Verify Booking

Make: Proton
Model: X50
Year: 2021
Condition: Good
Rent per Day: 150
Start Date: Tue Nov 30 00:00:00 MYT 2021
End Date: Fri Dec 03 00:00:00 MYT 2021
Signature:
5mt2hAjJObY7T95g/f3gd+D1qsiRS6AMl/mMbiXFdRoCkoLfxZRnQH7xaV614eFY8Dc
w6sXuQ9tZ3W/glAfCjFrPhwS//8A9Q9KPTHy4UHMIT1tS8ey0CLK+AoXKTOpy5lYRTl
U+CncL0QYtfMNZY4Ar/PiHAE2jrKCwC2a1tGg=

Booking ID: 23ec92a0-3217-46f0-9f8c-eae100c35af8

*Figure 19: Verification of digital signature in car rental system.*

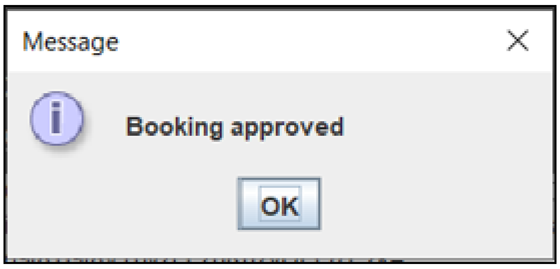
The management is directed to this verify booking page to verify, accept and decline the bookings from the client. The management staff can select the booking ID from the dropdown box at the top right corner. Once a booking id is selected, the car details and the digital signature will be showed.


Message

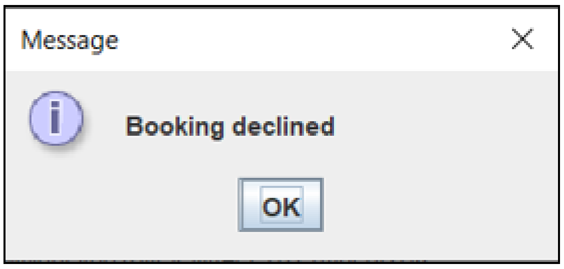
Correct Signature

*Figure 20: Prompt message showing current digital signature.*

The verify button will compare the digital signature and the data hashed using the same algorithm to match both hashed values. The prompt message will show if the value matches.


Message

Booking approved

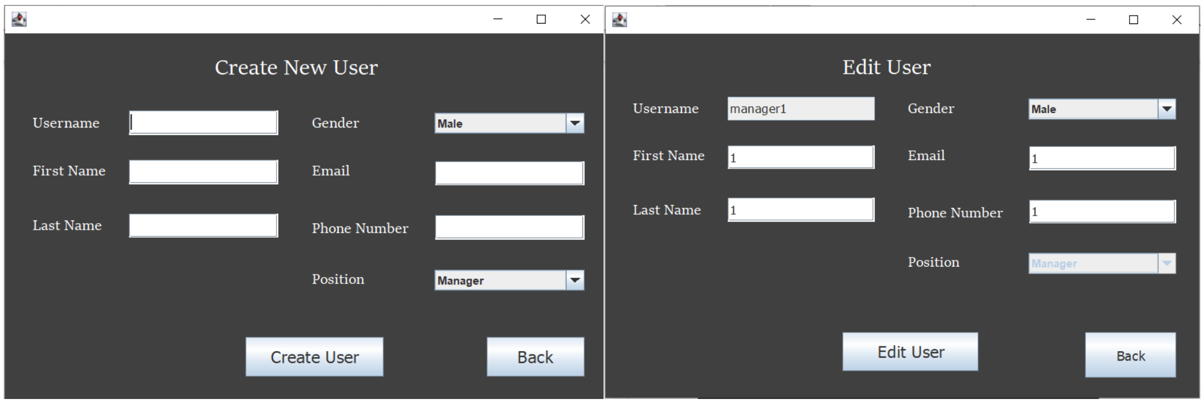

Message

Booking declined

*Figure 21: Alert messages of approving the booking (left) and declining the booking (right).*

The management staff can select either the accept or decline buttons at the bottom of the verify booking page. If the booking is approved, the accepted booking will convert into a block and form the blockchain. The ledger will be updated as more blocks are added into the blockchain. The booking which was declined will not be added into the blockchain and therefore not shown in the ledger.  al

**Management Staff: Creating new user and editing existing users**


Create New User

| | | | |
|---|---|---|---|
| Username | | Gender | Male |
| First Name | | Email | |
| Last Name | | Phone Number | |
| | | Position | Manager |

Edit User

| | | | |
|---|---|---|---|
| Username | manager1 | Gender | Male |
| First Name | 1 | Email | 1 |
| Last Name | 1 | Phone Number | 1 |
| | | Position | Manager |

*Figure 22 & 23: Creating new user and editing existing user in car rental system.*

The management staff can create new user and edit the details of the existing users. The figures above show the form for both of these functions in the car rental system.

## Reading the ledger

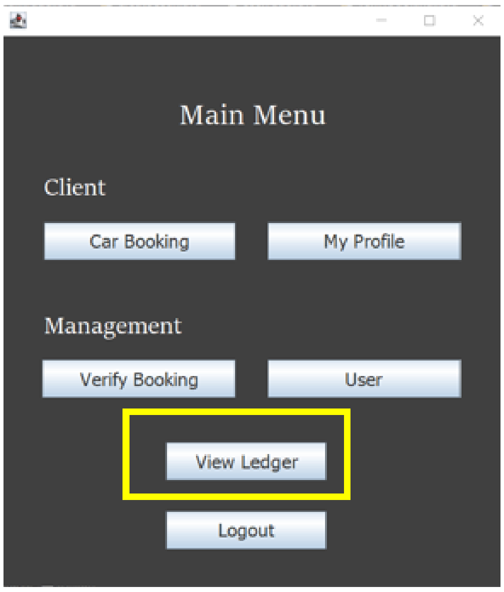
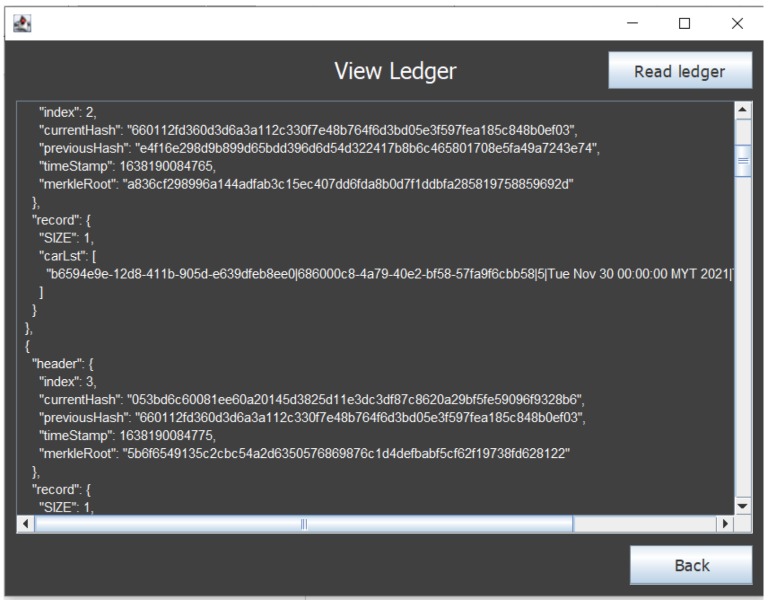


*Figure 24:* View ledger selection in the main menu page.

The client and the management staff can select the view ledger button in the main menu will show the accepted booking appended into the ledger.



```
    "index": 2,
    "currentHash": "660112fd360d3d6a3a112c330f7e48b764f6d3bd05e3f597fea185c848b0ef03",
    "previousHash": "e4f16e298d9b899d65bdd396d6d54d322417b8b6c465801708e5fa49a7243e74",
    "timeStamp": 1638190084765,
    "merkleRoot": "a836cf298996a144adfab3c15ec407dd6fda8b0d7f1ddbfa285819758859692d"
  },
  "record": {
    "SIZE": 1,
    "carLst": [
      "b6594e9e-12d8-411b-905d-e639dfeb8ee0|686000c8-4a79-40e2-bf58-57fa9f6cbb58|5|Tue Nov 30 00:00:00 MYT 2021|
    ]
  }
},
{
  "header": {
    "index": 3,
    "currentHash": "053bd6c60081ee60a20145d3825d11e3dc3df87c8620a29bf5fe59096f9328b6",
    "previousHash": "660112fd360d3d6a3a112c330f7e48b764f6d3bd05e3f597fea185c848b0ef03",
    "timeStamp": 1638190084775,
    "merkleRoot": "5b6f6549135c2cbc54a2d6350576869876c1d4defbabf5cf62f19738fd628122"
  },
  "record": {
    "SIZE": 1,
```

*Figure 25:* View ledger page with scroll bar.

The view ledger page will show the bookings added into the ledger by selecting the read ledger button. All the contents of the ledger can be view by using the scroll bar at bottom and right.

# 4.0 Solution design and implementation of blockchain algorithms

## 4.1 Hashing Algorithm

### 4.1.1 Overview

The term of hash represents the action of chopping and mixing. The hashing concept begin in 1953 after the first invention is implemented in electronic computer in 1950 (Rojasree et al., 2020). Hashing algorithm is used for generating fixed-sized string value (messages) to fixed and shorter length (message digest) from file which contains the data blocks. The original string is transformed into a shorter string with fixed length that represents the summary of the file.

A good hashing algorithm will use hashing that operates in unidirectional cannot be converted back to the original data. The hash value generated should change accordingly called the avalanche effect when even one byte in the data changed. Besides, the hash collisions are unavoidable when working with large data set known as the pigeonhole principle. The functions or algorithms to minimize on probability of collision are open addressing and separate chaining techniques (Dhar et al., 2018; Yusuf et al., 2021). Open addressing is storing the all the value in the available empty slot in the hash table. Sperate chaining is the process of storing the linked listed structured by mapping the data entry to the hash value. According to Liu & Xu (2015), the research conducted shows that close addressing have better stability than the open addressing in large set of data.

### 4.1.2 Benefits of hashing

The hashing technique is used as verification method to compare the validity between the copied file and the original file when transferring files among computers. According to Du et al. (2020), the hashing algorithm can be implemented for images authentication in the database. The proposed car rental system utilized the hashing algorithm to compare the users' booking details during verification to ensure no tampering or corruption of the data. The application of hashing is message digest hashing and password verification in the proposed system. The message digest hashing is to verify the data integrity by generating unique signature and compared by the validators. The hashing algorithm is also implemented in password verification to prevent identity theft and unauthorized access of account by hashing the password of the users.

### 4.1.3 Types of hashing

The hashing algorithms are categorized into cryptographic hash (keyed and unkeyed), cyclic redundancy checks and checksum functions. The commonly used hashing algorithms are the SHA-2 and MD-5 from cryptographic hash, and CRC-32 from cyclic redundancy checks. The message digest algorithm known as MD-5 produces 128-bit hash value. The MD5 algorithm is vulnerable but it is used for checksum on verifying data integrity on unintentional corruption of data. The security of the MD-5 algorithm can be improved by adding salt value or hashing the value multiple times. The salt value is randomly generated to create unique hashes even with similar input data. The salting technique can prevent the system from rainbow table attacks and pre-computed dictionary attacks (Rathod et al., 2020).

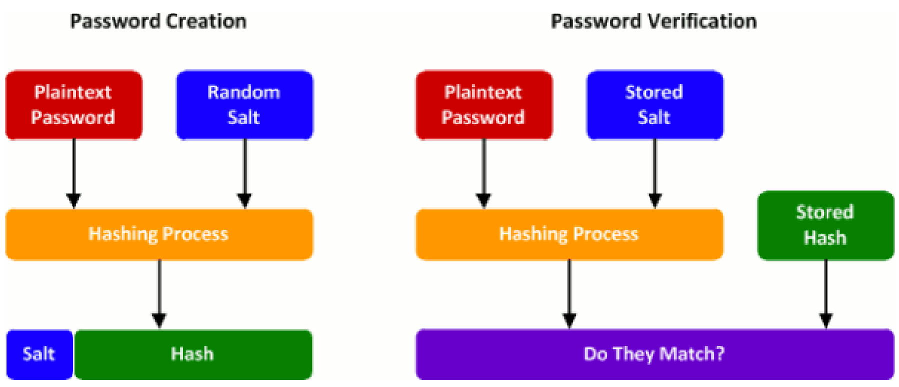


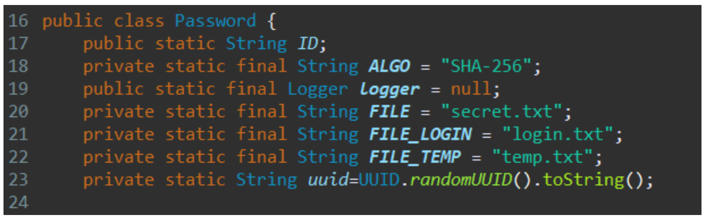
*Figure 26: Example of password hashing and verification process (Mostafa, 2019).*

The Secure Hash Algorithm 2 (SHA-2) is cryptographic hash functions that are created by the National Security Agency (NSA) in United States (Rachmawati et al., 2018). The SHA-2 is an improved version of the predecessor which is the SHA-1. The SHA-1 will produce 160 bits of hash value while the SHA-2 will the family of SHA-2 are consisting of SHA-224 (224 bits), SHA-256 (256 bits), SHA-384 (384 bits) and SHA-512 (512 bits). SHA-256 algorithm is the most secure hashing function that are required by the US government agencies for protecting sensitive information. The SHA-256 is secure against brute force and dictionary attacks where $2^{256}$ possibilities are required for generating the original data (Cortez et al., 2020). The selected algorithm for hashing is SHA-256 algorithm.

### 4.1.4 Source code and implementation of hashing in business

This section will discuss about the implementation of the hashing algorithm into the car rental blockchain system with source code provided as reference. The main hashing methods implemented in the car rental system is the password hashing.

**Initialization of Password**

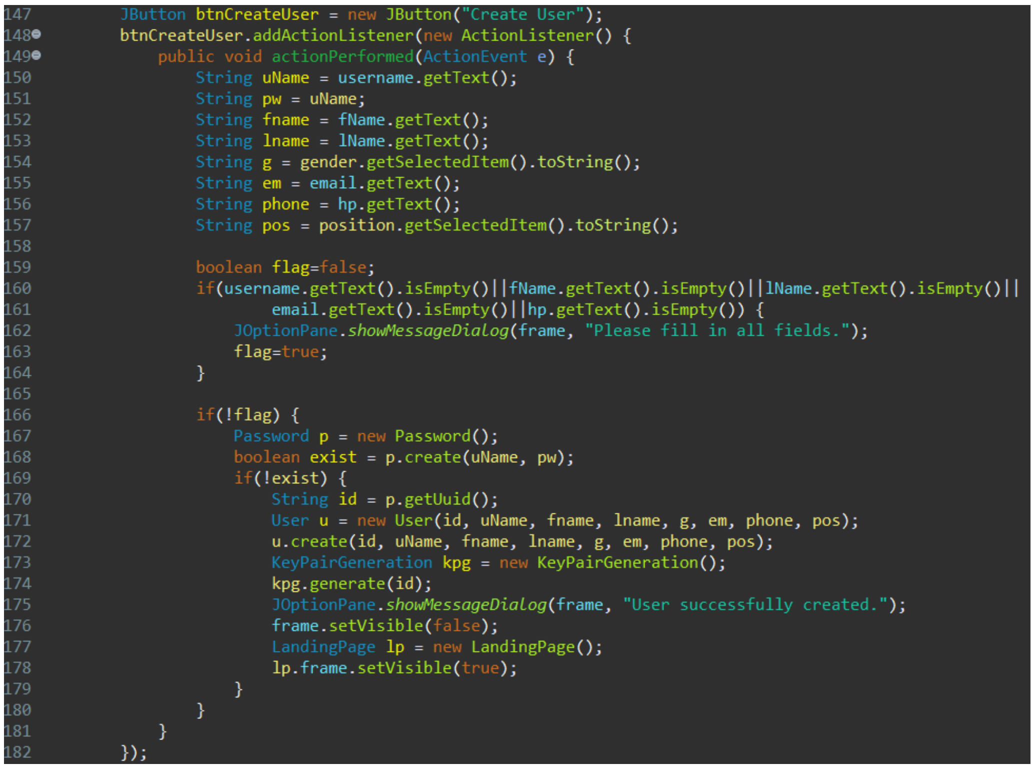


```
16 public class Password {
17     public static String ID;
18     private static final String ALGO = "SHA-256";
19     public static final Logger logger = null;
20     private static final String FILE = "secret.txt";
21     private static final String FILE_LOGIN = "login.txt";
22     private static final String FILE_TEMP = "temp.txt";
23     private static String uuid=UUID.randomUUID().toString();
24
```

*Figure 27: Password.java.*

The algorithm used for the password hashing is SHA-256 that is initialized in line 18. The other variables are also declared from line 17-23.

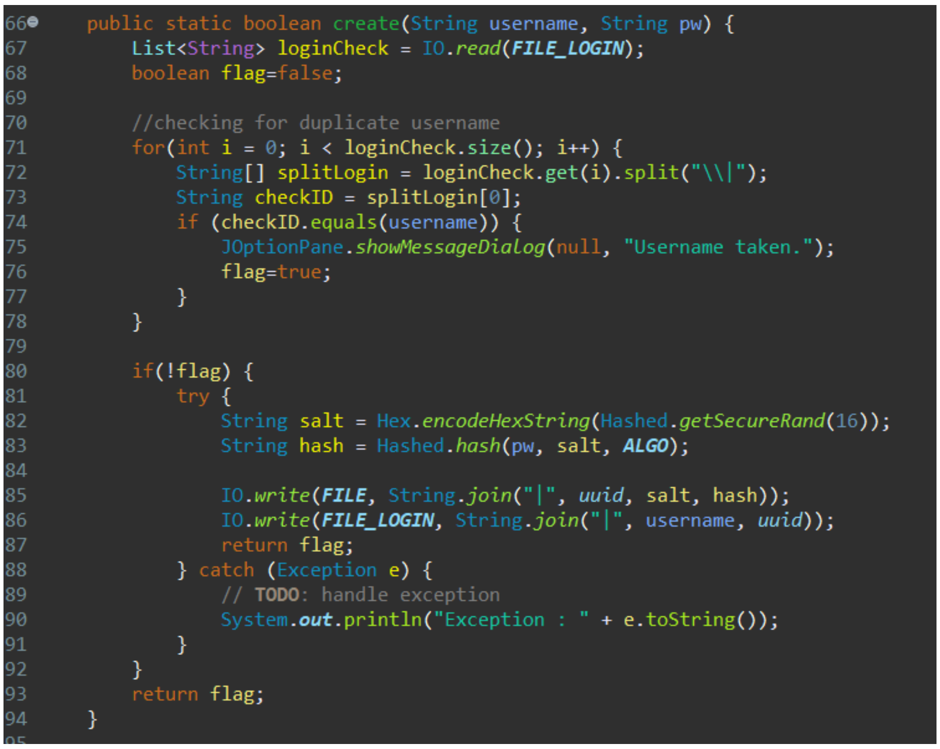
**Password creation**



```
        JButton btnCreateUser = new JButton("Create User");
        btnCreateUser.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String uName = username.getText();
                String pw = uName;
                String fname = fName.getText();
                String lname = lName.getText();
                String g = gender.getSelectedItem().toString();
                String em = email.getText();
                String phone = hp.getText();
                String pos = position.getSelectedItem().toString();

                boolean flag=false;
                if(username.getText().isEmpty()||fName.getText().isEmpty()||lName.getText().isEmpty()||
                        email.getText().isEmpty()||hp.getText().isEmpty()) {
                    JOptionPane.showMessageDialog(frame, "Please fill in all fields.");
                    flag=true;
                }

                if(!flag) {
                    Password p = new Password();
                    boolean exist = p.create(uName, pw);
                    if(!exist) {
                        String id = p.getUuid();
                        User u = new User(id, uName, fname, lname, g, em, phone, pos);
                        u.create(id, uName, fname, lname, g, em, phone, pos);
                        KeyPairGeneration kpg = new KeyPairGeneration();
                        kpg.generate(id);
                        JOptionPane.showMessageDialog(frame, "User successfully created.");
                        frame.setVisible(false);
                        LandingPage lp = new LandingPage();
                        lp.frame.setVisible(true);
                    }
                }
            }
        });
```

*Figure 28: CreateUser.java.*

The CreateUser class will users to access in the system for car rental features. Line 150-157 will require user details to be filled for creating a new user. The line 159-164 will check if any fields have not been entered and an alert message will show. The user password is generated same as the username when the user is created in line 168.

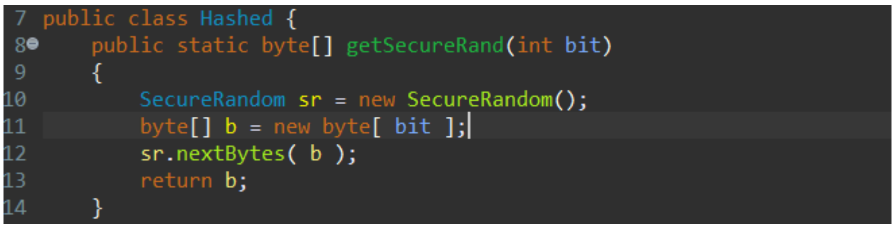


```
66     public static boolean create(String username, String pw) {
67         List<String> loginCheck = IO.read(FILE_LOGIN);
68         boolean flag=false;
69 
70         //checking for duplicate username
71         for(int i = 0; i < loginCheck.size(); i++) {
72             String[] splitLogin = loginCheck.get(i).split("\\|");
73             String checkID = splitLogin[0];
74             if (checkID.equals(username)) {
75                 JOptionPane.showMessageDialog(null, "Username taken.");
76                 flag=true;
77             }
78         }
79 
80         if(!flag) {
81             try {
82                 String salt = Hex.encodeHexString(Hashed.getSecureRand(16));
83                 String hash = Hashed.hash(pw, salt, ALGO);
84 
85                 IO.write(FILE, String.join("|", uuid, salt, hash));
86                 IO.write(FILE_LOGIN, String.join("|", username, uuid));
87                 return flag;
88             } catch (Exception e) {
89                 // TODO: handle exception
90                 System.out.println("Exception : " + e.toString());
91             }
92         }
93         return flag;
94     }
```

*Figure 29: Password.java.*

The create method is called in the Password class. The duplication of username is checked from line 74 to 77 and the system will prompt the user that username is taken. Line 82 will generate the salt by retrieving 16 bits from the getSecuredRand method and the byte is converted to hexadecimal characters in String format. Line 83 receive the parameters of the password, salt generated and the SHA-256 algorithm to generate the password hash.

## Creating salt

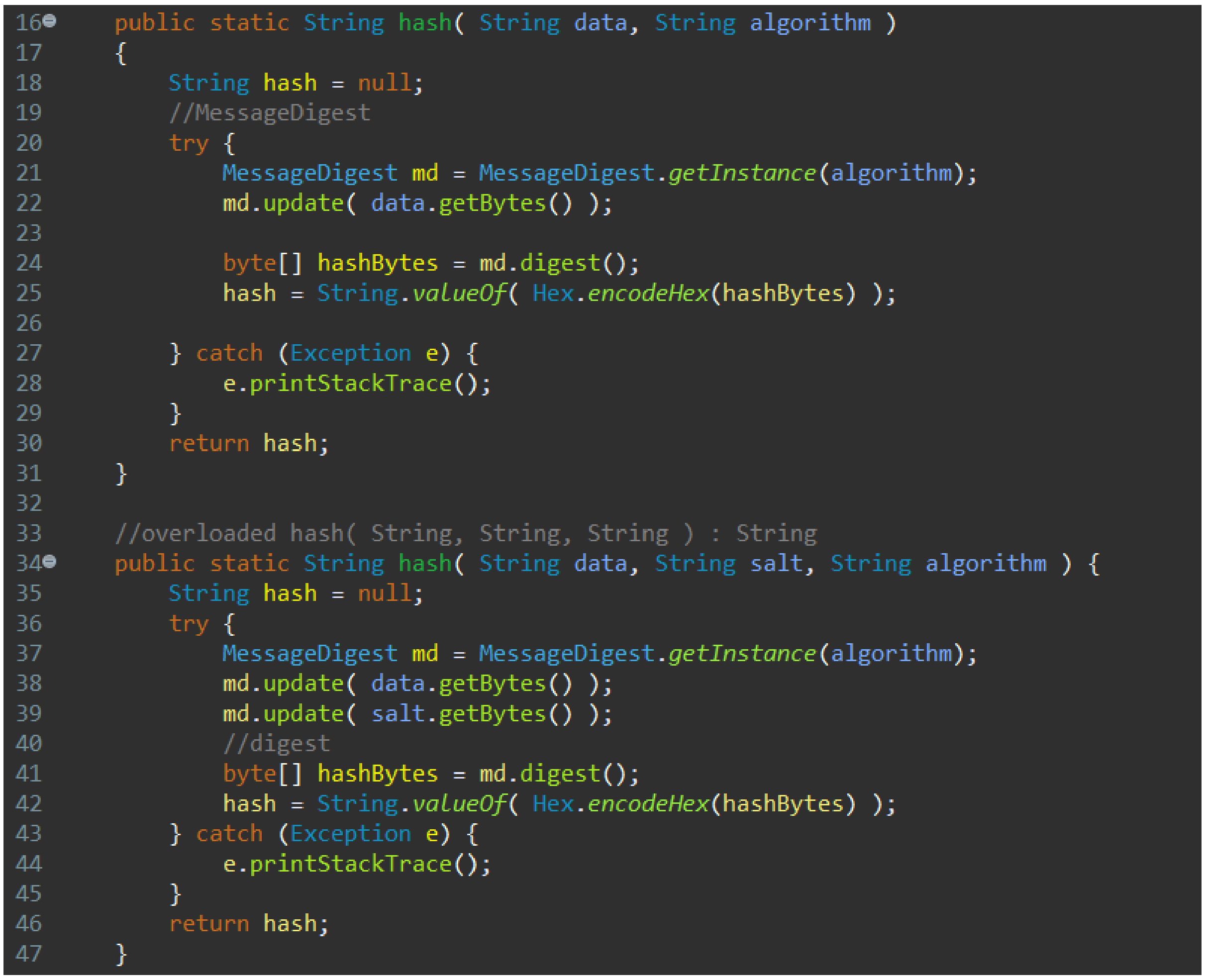


```
public class Hashed {
    public static byte[] getSecureRand(int bit)
    {
        SecureRandom sr = new SecureRandom();
        byte[] b = new byte[ bit ];
        sr.nextBytes( b );
        return b;
    }
```

*Figure 30: Hashed.java.*

Salt refers to a series of random data that is used as the additional input to secure the one-way hashing. It is generated using SecureRandom method in java.security library, contains a class that applies generate a cryptographic number using random number generator from line 8 to line 12. The salt generated is used as the input for hashing values.

### Hashing with salt

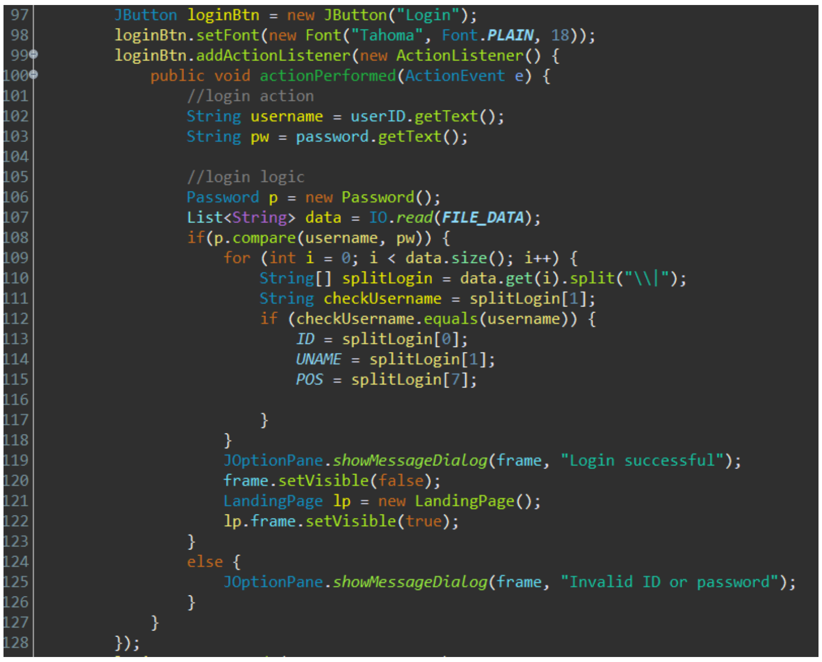


```
16      public static String hash( String data, String algorithm )
17      {
18          String hash = null;
19          //MessageDigest
20          try {
21              MessageDigest md = MessageDigest.getInstance(algorithm);
22              md.update( data.getBytes() );
23
24              byte[] hashBytes = md.digest();
25              hash = String.valueOf( Hex.encodeHex(hashBytes) );
26
27          } catch (Exception e) {
28              e.printStackTrace();
29          }
30          return hash;
31      }
32
33      //overloaded hash( String, String, String ) : String
34      public static String hash( String data, String salt, String algorithm ) {
35          String hash = null;
36          try {
37              MessageDigest md = MessageDigest.getInstance(algorithm);
38              md.update( data.getBytes() );
39              md.update( salt.getBytes() );
40              //digest
41              byte[] hashBytes = md.digest();
42              hash = String.valueOf( Hex.encodeHex(hashBytes) );
43          } catch (Exception e) {
44              e.printStackTrace();
45          }
46          return hash;
47      }
```

*Figure 31: Hashed.java*

Hashing method is used for hashing a string of data based on the predefined algorithm. The method will take the string of data to be hashed as well as the algorithm as the input, which is then used to output and return the string of the hashed data in line 16-30. Line 34-47 shows an overloaded method is also used for hashing in the case that the Salt value is provided.

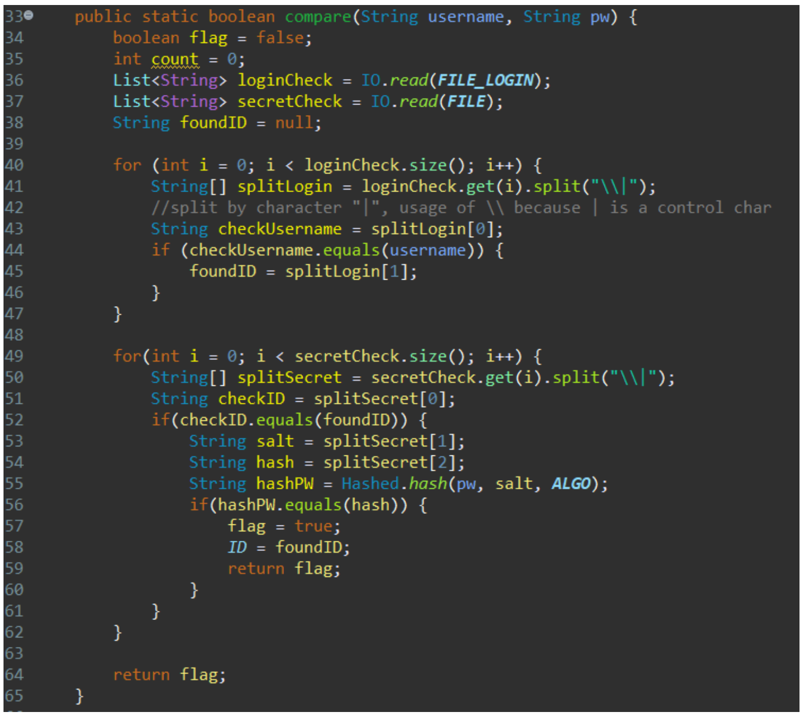
## Comparing hashed password



```java
 97        JButton loginBtn = new JButton("Login");
 98        loginBtn.setFont(new Font("Tahoma", Font.PLAIN, 18));
 99        loginBtn.addActionListener(new ActionListener() {
100            public void actionPerformed(ActionEvent e) {
101                //login action
102                String username = userID.getText();
103                String pw = password.getText();
104
105                //login logic
106                Password p = new Password();
107                List<String> data = IO.read(FILE_DATA);
108                if(p.compare(username, pw)) {
109                    for (int i = 0; i < data.size(); i++) {
110                        String[] splitLogin = data.get(i).split("\\|");
111                        String checkUsername = splitLogin[1];
112                        if (checkUsername.equals(username)) {
113                            ID = splitLogin[0];
114                            UNAME = splitLogin[1];
115                            POS = splitLogin[7];
116
117                        }
118                    }
119                    JOptionPane.showMessageDialog(frame, "Login successful");
120                    frame.setVisible(false);
121                    LandingPage lp = new LandingPage();
122                    lp.frame.setVisible(true);
123                }
124                else {
125                    JOptionPane.showMessageDialog(frame, "Invalid ID or password");
126                }
127            }
128        });
```

*Figure 32: LoginPage.java*

The username and password are obtained when the user attempted to login in the LoginPage class. The password entered is compared in line 108 to verify the user authorization. The user will be successfully login when password is correct, while user is required to re-enter the password if invalid username or password is entered.

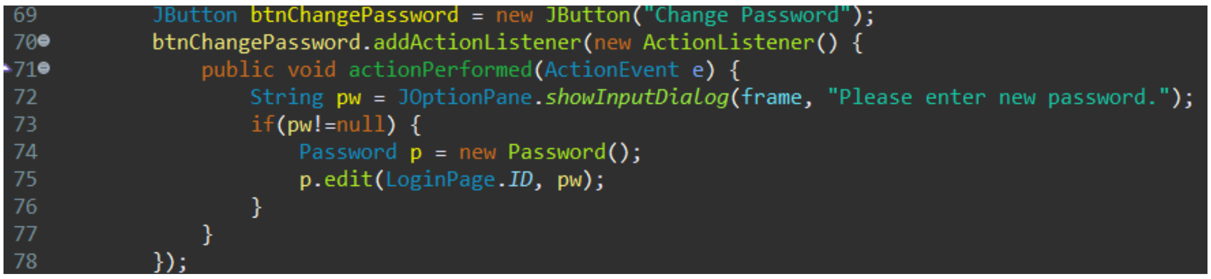


```java
    public static boolean compare(String username, String pw) {
        boolean flag = false;
        int count = 0;
        List<String> loginCheck = IO.read(FILE_LOGIN);
        List<String> secretCheck = IO.read(FILE);
        String foundID = null;

        for (int i = 0; i < loginCheck.size(); i++) {
            String[] splitLogin = loginCheck.get(i).split("\\|");
            //split by character "|", usage of \\ because | is a control char
            String checkUsername = splitLogin[0];
            if (checkUsername.equals(username)) {
                foundID = splitLogin[1];
            }
        }

        for(int i = 0; i < secretCheck.size(); i++) {
            String[] splitSecret = secretCheck.get(i).split("\\|");
            String checkID = splitSecret[0];
            if(checkID.equals(foundID)) {
                String salt = splitSecret[1];
                String hash = splitSecret[2];
                String hashPW = Hashed.hash(pw, salt, ALGO);
                if(hashPW.equals(hash)) {
                    flag = true;
                    ID = foundID;
                    return flag;
                }
            }
        }

        return flag;
    }
```

*Figure 33: Password.java*

The compare method in Password class is called with the parameters of username and password from the LoginPage class. The username is validated in line 40-47, and the username found in the database is used to validate the password in line 49 to 62. The password entered is hashed using password, salt generated and SHA-256 to match with the password hash stored in the database from line 57-60.

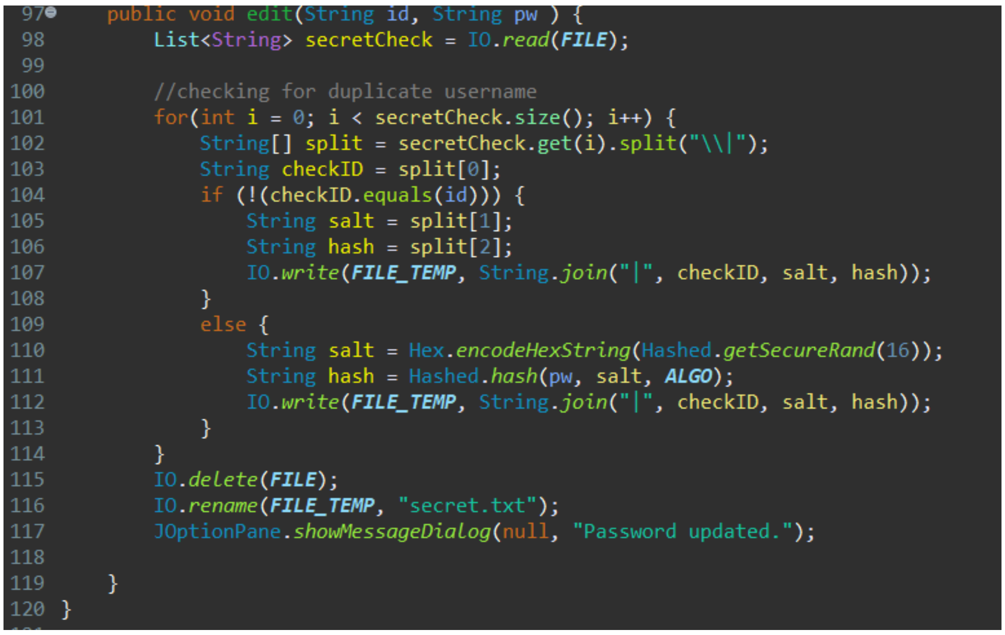
### Editing hashed password



```
        JButton btnChangePassword = new JButton("Change Password");
        btnChangePassword.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String pw = JOptionPane.showInputDialog(frame, "Please enter new password.");
                if(pw!=null) {
                    Password p = new Password();
                    p.edit(LoginPage.ID, pw);
                }
            }
        });
```

*Figure 34: MyProfile.java.*

The user can change their original password by entering the new password in the MyProfile class. The edit method will be called to replace the new password with the old password.

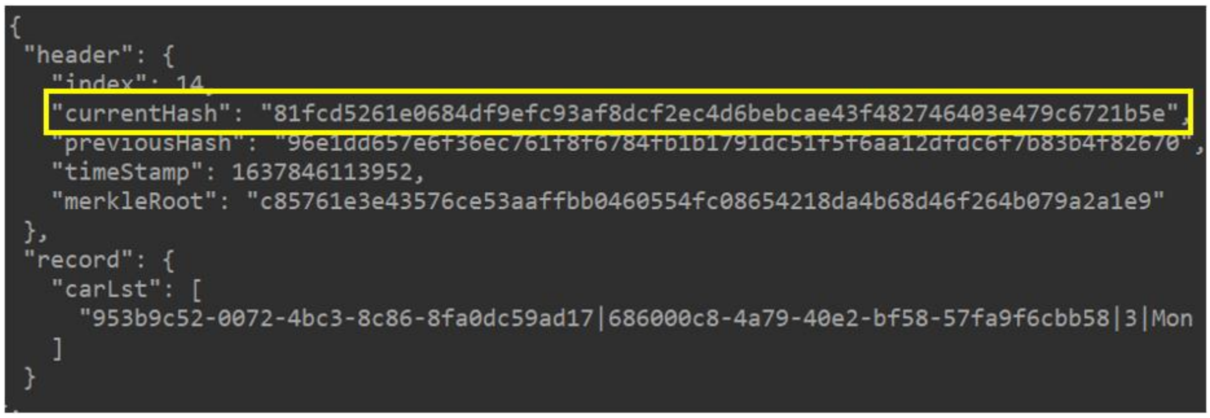


```
97      public void edit(String id, String pw ) {
98          List<String> secretCheck = IO.read(FILE);
99
100         //checking for duplicate username
101         for(int i = 0; i < secretCheck.size(); i++) {
102             String[] split = secretCheck.get(i).split("\\|");
103             String checkID = split[0];
104             if (!(checkID.equals(id))) {
105                 String salt = split[1];
106                 String hash = split[2];
107                 IO.write(FILE_TEMP, String.join("|", checkID, salt, hash));
108             }
109             else {
110                 String salt = Hex.encodeHexString(Hashed.getSecureRand(16));
111                 String hash = Hashed.hash(pw, salt, ALGO);
112                 IO.write(FILE_TEMP, String.join("|", checkID, salt, hash));
113             }
114         }
115         IO.delete(FILE);
116         IO.rename(FILE_TEMP, "secret.txt");
117         JOptionPane.showMessageDialog(null, "Password updated.");
118
119     }
120 }
```

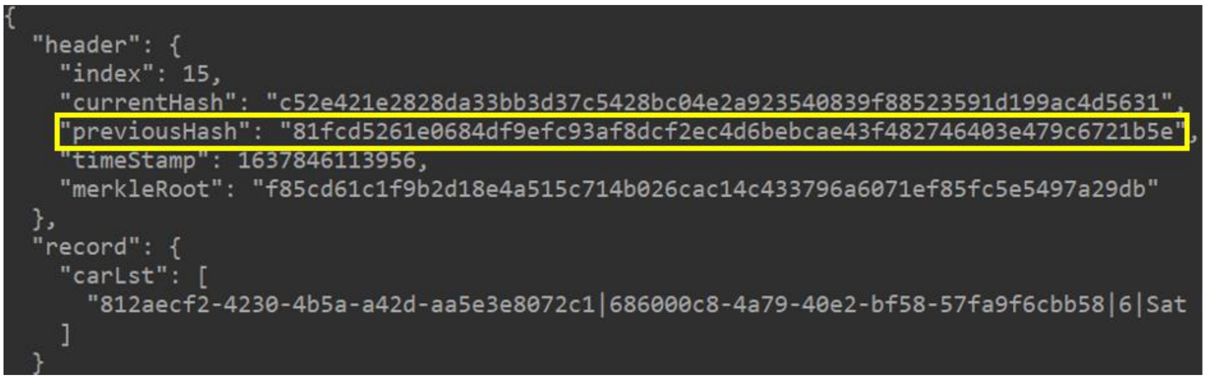
*Figure 35: Password.java.*

The new password will be hashed with same parameters from line 110 to line 112. The user will be updated with the new password with the salted value in the database.

### Hashing in ledger



```
{
 "header": {
   "index": 14,
   "currentHash": "81fcd5261e0684df9efc93af8dcf2ec4d6bebcae43f482746403e479c6721b5e",
   "previousHash": "96e1dd657e6f36ec761f8f6784fb1b1791dc51f5f6aa12dfdc6f7b83b4f82670",
   "timeStamp": 1637846113952,
   "merkleRoot": "c85761e3e43576ce53aaffbb0460554fc08654218da4b68d46f264b079a2a1e9"
 },
 "record": {
   "carLst": [
     "953b9c52-0072-4bc3-8c86-8fa0dc59ad17|686000c8-4a79-40e2-bf58-57fa9f6cbb58|3|Mon
   ]
 }
```

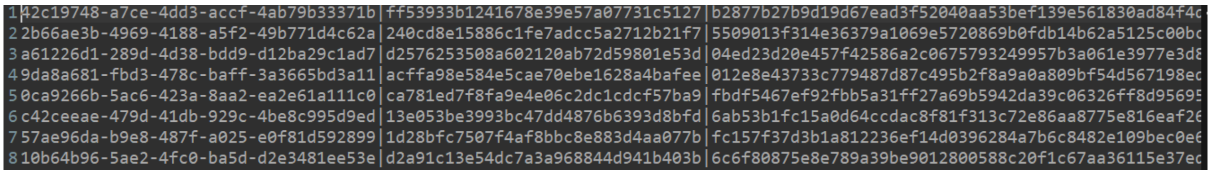


```
{
  "header": {
    "index": 15,
    "currentHash": "c52e421e2828da33bb3d37c5428bc04e2a923540839f88523591d199ac4d5631",
    "previousHash": "81fcd5261e0684df9efc93af8dcf2ec4d6bebcae43f482746403e479c6721b5e",
    "timeStamp": 1637846113956,
    "merkleRoot": "f85cd61c1f9b2d18e4a515c714b026cac14c433796a6071ef85fc5e5497a29db"
  },
  "record": {
    "carLst": [
      "812aecf2-4230-4b5a-a42d-aa5e3e8072c1|686000c8-4a79-40e2-bf58-57fa9f6cbb58|6|Sat
    ]
  }
```

*Figure 36: Comparison of the hash value of the corresponding blocks.*

A unique hashed value is generated for the current block and the value of the previous hash will be display as more values are added into the ledger.

### Hashing of user password



```
42c19748-a7ce-4dd3-accf-4ab79b33371b|ff53933b1241678e39e57a07731c5127|b2877b27b9d19d67ead3f52040aa53bef139e561830ad84f4c
2b66ae3b-4969-4188-a5f2-49b771d4c62a|240cd8e15886c1fe7adcc5a2712b21f7|5509013f314e36379a1069e5720869b0fdb14b62a5125c00bc
a61226d1-289d-4d38-bdd9-d12ba29c1ad7|d2576253508a602120ab72d59801e53d|04ed23d20e457f42586a2c0675793249957b3a061e3977e3d8
9da8a681-fbd3-478c-baff-3a3665bd3a11|acffa98e584e5cae70ebe1628a4bafee|012e8e43733c779487d87c495b2f8a9a0a809bf54d567198ec
0ca9266b-5ac6-423a-8aa2-ea2e61a111c0|ca781ed7f8fa9e4e06c2dc1cdcf57ba9|fbdf5467ef92fbb5a31ff27a69b5942da39c06326ff8d95695
c42ceeae-479d-41db-929c-4be8c995d9ed|13e053be3993bc47dd4876b6393d8bfd|6ab53b1fc15a0d64ccdac8f81f313c72e86aa8775e816eaf26
57ae96da-b9e8-487f-a025-e0f81d592899|1d28bfc7507f4af8bbc8e883d4aa077b|fc157f37d3b1a812236ef14d0396284a7b6c8482e109bec0e6
10b64b96-5ae2-4fc0-ba5d-d2e3481ee53e|d2a91c13e54dc7a3a968844d941b403b|6c6f80875e8e789a39be9012800588c20f1c67aa36115e37ec
```

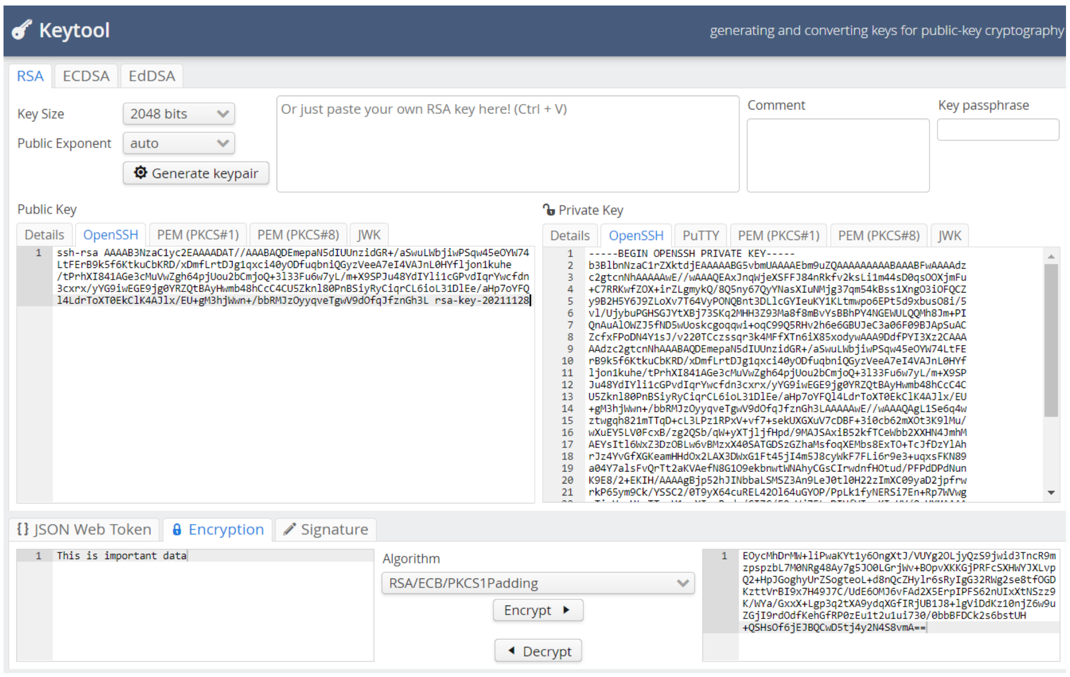
*Figure 37: Hashing with salting of user password in SHA-256 algorithm.*

The text file showed is the hashing algorithm implemented with added salt value using the SHA-256 algorithm. The file is stored in corresponding order of the randomized user id, salt value and the hashed password.

## 4.2 Cryptography Algorithm

### 4.2.1 Overview

Cryptography is the study of technique to secure the communications channels between the sender and receiver on preventing malicious person or third parties to access the data. The term cryptography means hidden, which is kryptos in Greek language. The origin of cryptography is dated back in 19th century for converting information known as plaintext into intelligible word known as ciphertext. The cryptography techniques are becoming increasingly complex since the development of the rotor cipher machine in World War 1 and computers in World War 2. The most used algorithm is called the RSA algorithm using asymmetric cryptography developed from Ron Rivest, Adi Shamir and Leonard Adleman in 1977.



Key Size: 2048 bits
Public Exponent: auto
Algorithm: RSA/ECB/PKCS1Padding

Public Key

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAT//AAABAQDEmepaN5dIUUnzidGR+/aSwuLWbjiwPSqw45eOYW74
LtFErB9k5f6KtkuCbKRD/xDmfLrtDJg1qxci40yODfuqbniQGyzVeeA7eI4VAJnL0HYfljon1kuhe
/tPrhXI841AGe3cMuVwZgh64pjUou2bCmjoQ+3l33Fu6w7yL/m+X9SPJu48YdIYli1cGPvdIqrYwcfdn
3cxrx/yYG9iwEGE9jg0YRZQtBAyHwmb48hCcC4CU5Zknl80PnBSiyRyCiqrCL6ioL31DlEe/aHp7oYFQ
l4LdrToXT0EkClK4AJlx/EU+gM3hjWwn+/bbRMJzOyyqveTgwV9dOfqJfznGh3L rsa-key-20211128
```

Private Key

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABFwAAAAdz
c2gtcnNhAAAAAwE//wAAAQEAxJnqWjeXSFFJ84nRkfv2ksLi1m44sD0qsOOXjmFu
+C7RRKwfZOX+irZLgmykQ/8Q5ny67QyYNasXIuNMjg37qm54kBss1XngO3iOFQCZ
y9B2H5Y6J9ZLoXv7T64VyPONQBnt3DLlcGYIeuKY1KLtmwpo6EPt5d9xbusO8i/5
vl/UjybuPGHSGJYtXBj73SKq2MHH3Z93Ma8f8mBvYsBBhPY4NGEWULQQMh8Jm+PI
QnAuAlOWZJ5fND5wUoskcgoqqwi+oqC99Q5RHv2h6e6GBUJeC3a06F09BJApSuAC
ZcfxFPoDN4Y1sJ/v22OTCczssqr3k4MFfXTn6iX85xodywAAA9DdfPYI3Xz2CAAA
AAdzc2gtcnNhAAABAQDEmepaN5dIUUnzidGR+/aSwuLWbjiwPSqw45eOYW74LtFE
rB9k5f6KtkuCbKRD/xDmfLrtDJg1qxci40yODfuqbniQGyzVeeA7eI4VAJnL0HYf
ljon1kuhe/tPrhXI841AGe3cMuVwZgh64pjUou2bCmjoQ+3l33Fu6w7yL/m+X9SP
Ju48YdIYli1cGPvdIqrYwcfdn3cxrx/yYG9iwEGE9jg0YRZQtBAyHwmb48hCcC4C
U5Zknl80PnBSiyRyCiqrCL6ioL31DlEe/aHp7oYFQl4LdrToXT0EkClK4AJlx/EU
+gM3hjWwn+/bbRMJzOyyqveTgwV9dOfqJfznGh3LAAAAAwE//wAAAQAgL1Se6q4w
ztwgqh821mTTqD+cL3LPz1RPxV+vf7+sekUXGXuV7cDBF+3i0cb62mXOt3K9lMu/
wXuEY5LV0FcxB/zg2QSb/qW+yXTjljfHpd/9MAJSAxiB52kfTCeWbb2XXHN4JmhM
AEYsItl6WxZ3DzOBLw6vBMzxX40SATGDSzGZhaMsfoqXEMbs8ExTO+TcJfDzYlAh
rJz4YvGfXGKeamHHdOx2LAX3DWxG1Ft45jI4m5J8cyWkF7FLi6r9e3+uqxsFKN89
a04Y7alsFvQrTt2aKVAefN8G1O9ekbnwtWNAhyCGsCIrwdnfHOtud/PFPdDPdNun
K9E8/2+EKIH/AAAAgBjp52hJINbbaLSMSZ3An9LeJ0tl0H22zImXC09yaD2jpfrw
rkP65ym9Ck/YSSC2/0T9yX64cuREL42Ol64uGYOP/PpLk1fyNERSi7En+Rp7WVwg
```

```
This is important data
```

```
EOycMhDrMW+liPwaKYt1y6OngXtJ/VUYg2OLjyQzS9jwid3TncR9m
zpspzbL7M0NRg48Ay7g5JO0LGrjWv+BOpvXKKGjPRFcSXHWYJXLvp
Q2+HpJGoghyUrZSogteoL+d8nQcZHylr6sRyIgG32RWg2se8tfOGD
KzttVrBI9x7H49J7C/UdE6OMJ6vFAd2X5ErpIPFS62nUIxXtNSzz9
K/WYa/GxxX+Lgp3q2tXA9ydqXGfIRjUB1J8+lgViDdKz10njZ6w9u
ZGjI9rdOdfKehGfRP0zEu1t2u1ui730/0bbBFDCk2s6bstUH
+QSHsOf6jEJBQCwD5tj4y2N4S8vmA==
```

*Figure 38: Example of public and private key generated for encryption and decryption (KeyTool, 2021).*

The figure shows the generating public and private key pairs using RSA algorithm for encryption and decryption of data. The encryption will generate string of bits calculated from the plaintext. The larger the number of bits will generate larger key and requires more combinations for breaking and unscrambling the data.

### 4.2.2 Benefits of cryptography

The benefits of implementing cryptography are broken down into 4 categories which are highly secure, provides data integrity, authentication of users and non-repudiation for assurance.

Confidentiality is enhancing the set of rules for limiting the access and placing restrictions towards the private information. In the context of cryptography, confidentiality can be achieved by securing the network from unauthorize users towards classified or sensitive information. The data can be encryption by using encryption standards like Advanced Encryption Standard (AES) and Data Encryption Standard (DES). These algorithms are using the computational-complexity approach where the leaked data cannot be efficiently exploited by the hackers to retrieve the plaintext from the ciphertext (Ferradi, 2016).

The security functions of integrity, authentication and fairness can be achieved the asymmetric and symmetric cryptography. The application of digital signature allows the receiver to authenticate and verify the origin of  the message and checks that information is intext by data integrity (Ferradi, 2016). Another feature of the digital signature in cryptography is non-repudiation that provides proof of delivery from the sender. The digital signature is becoming the building block of the cryptography operations and they are applied in digital currencies, financial transaction, supply chain management and smart contract management. The following section on the digital signature will discuss in-depth about the digital signature.

The security features of integrity and authentication are often overlapped as integrity also provides authentication. The algorithm used for the integrity is the SHA and MD5. For maintaining the integrity of the data, the data from the sender is hashed using the same hash function discussed in the previous section on hashing. The original hashed data will be compared with the received data that is hashed to ensure the data is not modified, corrupted, and tampered by malicious attackers or unintentionally.

The fairness of the digital signature protocol can be achieved with majority of the party in agreement after verification of the data. The data is secure as long there is more than fifty percent of honest nodes in the network. The implementation of digital signature can allow the system to achieve guaranteed data integrity and fairness.

### 4.2.3 Types of cryptography

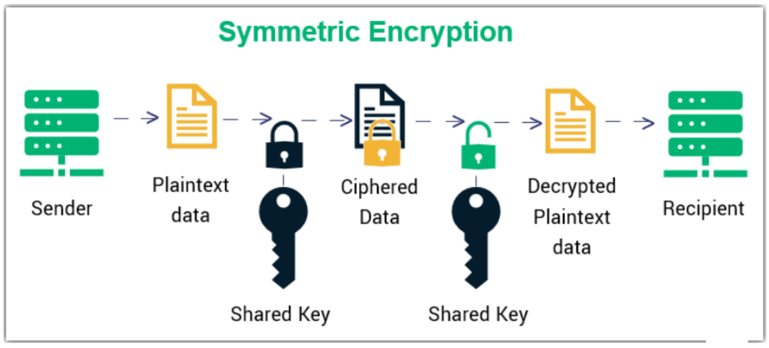


*Figure 39: Illustration of symmetric encryption using shared key (InfoSec, 2020).*

The cryptography is categorized into two categories which are asymmetric and symmetric encryption. The symmetric encryption uses one key for encryption and decryption, while the asymmetric key encrypt data using public key and decrypt data using private key. The symmetric encryption is less complex and allows faster execution when transferring large amount of data (Simmons, 2019). The key which is used for encryption of the plaintext will also be used for decryption of the ciphertext. The algorithms that are widely known for symmetric encryption is AES-128, AES-192 and AES-256. AES algorithm is applied in wireless security, SSL/TLS protocols, VPN and many more.

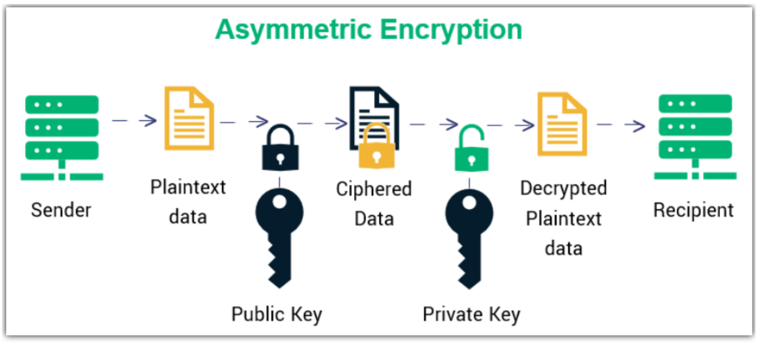


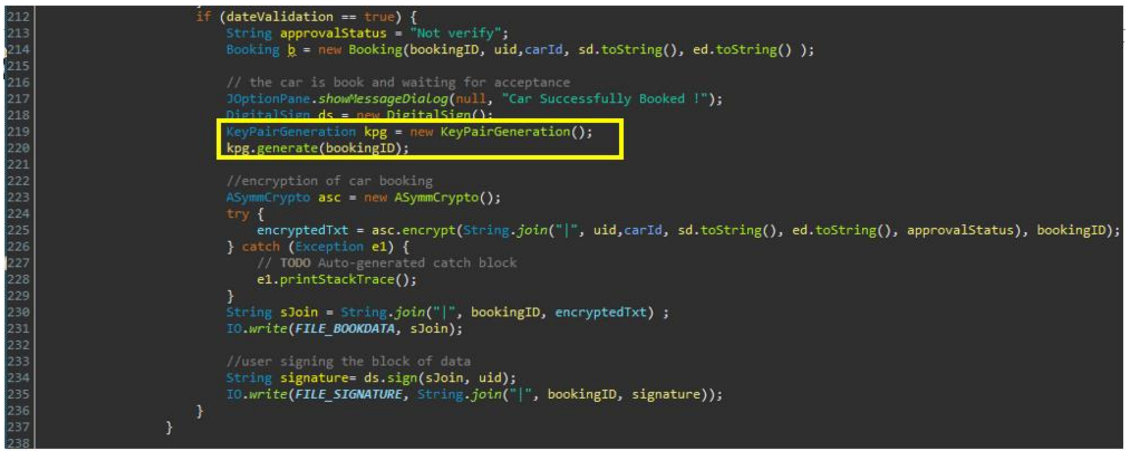
*Figure 40: Illustration of asymmetric encryption using key pairs (InfoSec, 2020).*

The asymmetric encryption is more secure than the symmetric encryption as both public and private keys are used. The public key used for encryption can only be decrypted using the private key, and vice versa, if the private key used for encryption the public key is used for decryption (Simmons, 2019). The asymmetric encryption uses in communication between client and server model where an encrypted certificate can send to another party. The common use of asymmetric encryption is blockchain implementation, public key infrastructure and digital signature.

### 4.2.4 Source code and implementation of cryptography in business

This section will discuss the implementation of cryptography in the car rental using asymmetric encryption by generating private and public keys. The generation of key pairs, encryption and decryption are briefly explained with source code attached below.

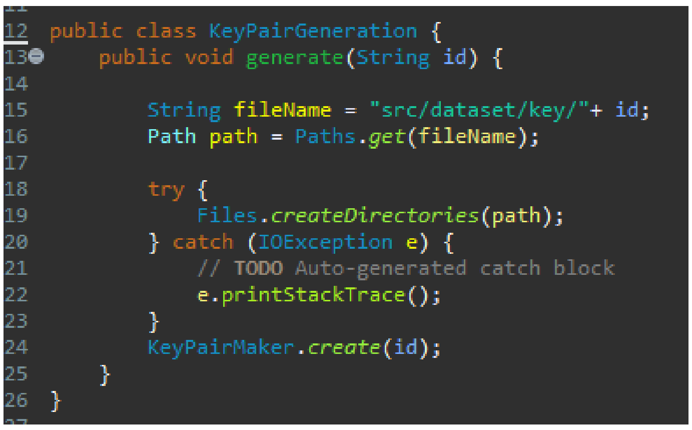
**Generating public and private key pairs**



```
212                     if (dateValidation == true) {
213                         String approvalStatus = "Not verify";
214                         Booking b = new Booking(bookingID, uid,carId, sd.toString(), ed.toString() );
215 
216                         // the car is book and waiting for acceptance
217                         JOptionPane.showMessageDialog(null, "Car Successfully Booked !");
218                         DigitalSign ds = new DigitalSign();
219                         KeyPairGeneration kpg = new KeyPairGeneration();
220                         kpg.generate(bookingID);
221 
222                         //encryption of car booking
223                         ASymmCrypto asc = new ASymmCrypto();
224                         try {
225                             encryptedTxt = asc.encrypt(String.join("|", uid,carId, sd.toString(), ed.toString(), approvalStatus), bookingID);
226                         } catch (Exception e1) {
227                             // TODO Auto-generated catch block
228                             e1.printStackTrace();
229                         }
230                         String sJoin = String.join("|", bookingID, encryptedTxt) ;
231                         IO.write(FILE_BOOKDATA, sJoin);
232 
233                         //user signing the block of data
234                         String signature= ds.sign(sJoin, uid);
235                         IO.write(FILE_SIGNATURE, String.join("|", bookingID, signature));
236                     }
237                 }
238 
```

*Figure 41: CarBooking.java.*

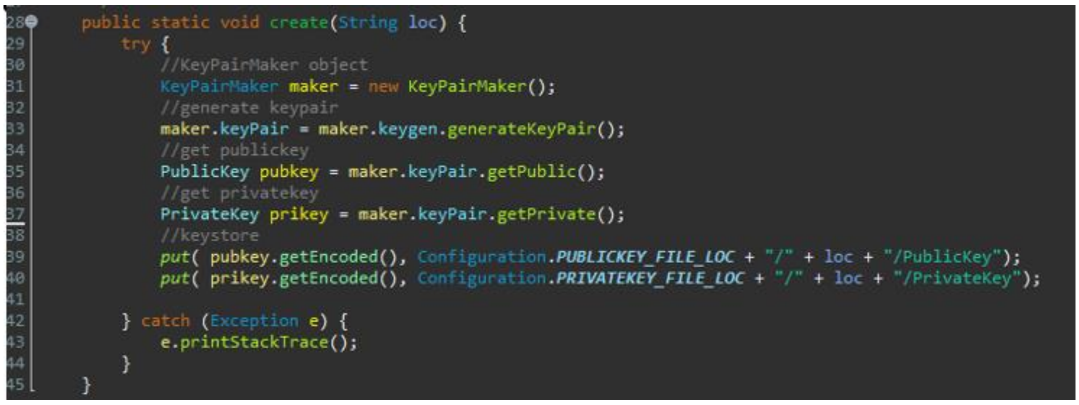
After the users entered car booking details and waiting for acceptance in the CarBooking class, the generate method from the KeyPairGeneration class is called to create new directory using the randomized booking id in line 219 and 220.



```
12 public class KeyPairGeneration {
13     public void generate(String id) {
14 
15         String fileName = "src/dataset/key/"+ id;
16         Path path = Paths.get(fileName);
17 
18         try {
19             Files.createDirectories(path);
20         } catch (IOException e) {
21             // TODO Auto-generated catch block
22             e.printStackTrace();
23         }
24         KeyPairMaker.create(id);
25     }
26 }
```

*Figure 42: KeyPairGeneration.java.*

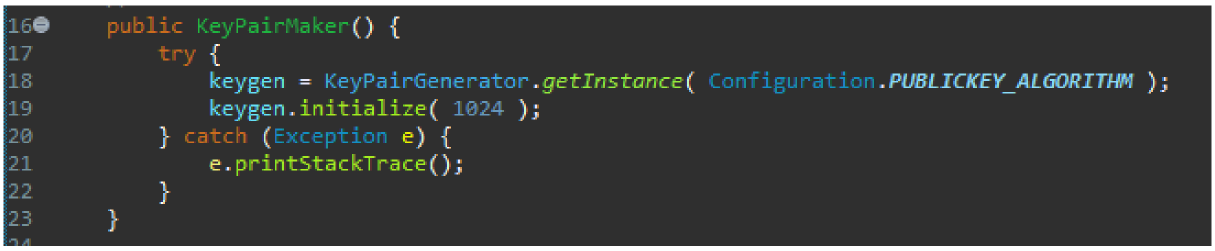
The generate method in KeyPairGeneration class will create a new directory based the randomized id from the randomUUID method in the UUID class in line 15. Line 24 will call the create method from the KeyPairMaker class will create the private and public key pairs by referring to the booking id.



```
28      public static void create(String loc) {
29          try {
30              //KeyPairMaker object
31              KeyPairMaker maker = new KeyPairMaker();
32              //generate keypair
33              maker.keyPair = maker.keygen.generateKeyPair();
34              //get publickey
35              PublicKey pubkey = maker.keyPair.getPublic();
36              //get privatekey
37              PrivateKey prikey = maker.keyPair.getPrivate();
38              //keystore
39              put( pubkey.getEncoded(), Configuration.PUBLICKEY_FILE_LOC + "/" + loc + "/PublicKey");
40              put( prikey.getEncoded(), Configuration.PRIVATEKEY_FILE_LOC + "/" + loc + "/PrivateKey");
41
42          } catch (Exception e) {
43              e.printStackTrace();
44          }
45      }
```

*Figure 43: KeyPairMaker.java.*

The create method form the KeyPairMaker class will generate the private and public key pair using the generateKeyPair method from the *java.security.KeyPairGenerator* library in line 33. The public key is retrieve using the getPublic method, while the private key is retrieved using the getPrivate method in line 35 and 37. Line 39 and 40 will call the put method with the parameters of encoded private and public key separately with their respective path directory.

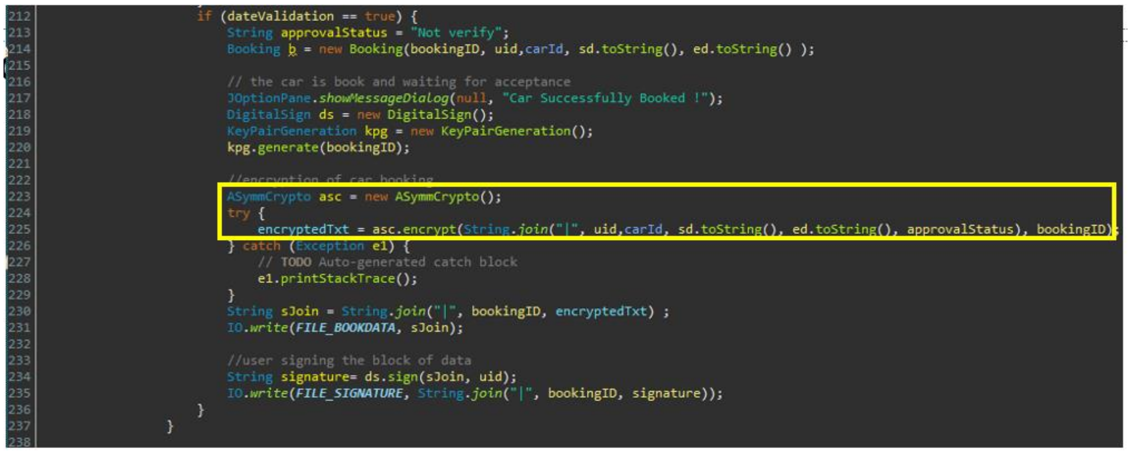


```
16      public KeyPairMaker() {
17          try {
18              keygen = KeyPairGenerator.getInstance( Configuration.PUBLICKEY_ALGORITHM );
19              keygen.initialize( 1024 );
20          } catch (Exception e) {
21              e.printStackTrace();
22          }
23      }
```

*Figure 44: KeyPairMaker.java.*

The KeyPairMaker constructor from the KeyPairMaker class will return a new KeyPairGenerator object from the *java.security.KeyPairGenerator* library in line 18. Line 19 will generate random key pair with the specified 1024 bits in key size.

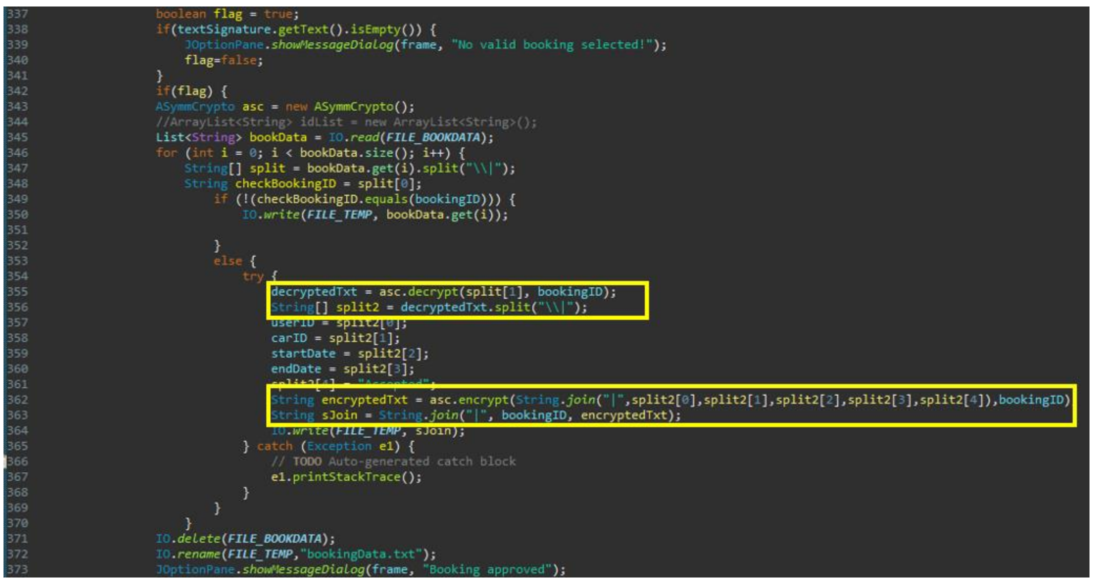
**Encryption and decryption of data**



```
212             if (dateValidation == true) {
213                 String approvalStatus = "Not verify";
214                 Booking b = new Booking(bookingID, uid,carId, sd.toString(), ed.toString() );
215 
216                 // the car is book and waiting for acceptance
217                 JOptionPane.showMessageDialog(null, "Car Successfully Booked !");
218                 DigitalSign ds = new DigitalSign();
219                 KeyPairGeneration kpg = new KeyPairGeneration();
220                 kpg.generate(bookingID);
221 
222                 //encryption of car booking
223                 ASymmCrypto asc = new ASymmCrypto();
224                 try {
225                     encryptedTxt = asc.encrypt(String.join("|", uid,carId, sd.toString(), ed.toString(), approvalStatus), bookingID);
226                 } catch (Exception e1) {
227                     // TODO Auto-generated catch block
228                     e1.printStackTrace();
229                 }
230                 String sJoin = String.join("|", bookingID, encryptedTxt) ;
231                 IO.write(FILE_BOOKDATA, sJoin);
232 
233                 //user signing the block of data
234                 String signature= ds.sign(sJoin, uid);
235                 IO.write(FILE_SIGNATURE, String.join("|", bookingID, signature));
236             }
237         }
238 
```

*Figure 45: CarBooking.java.*

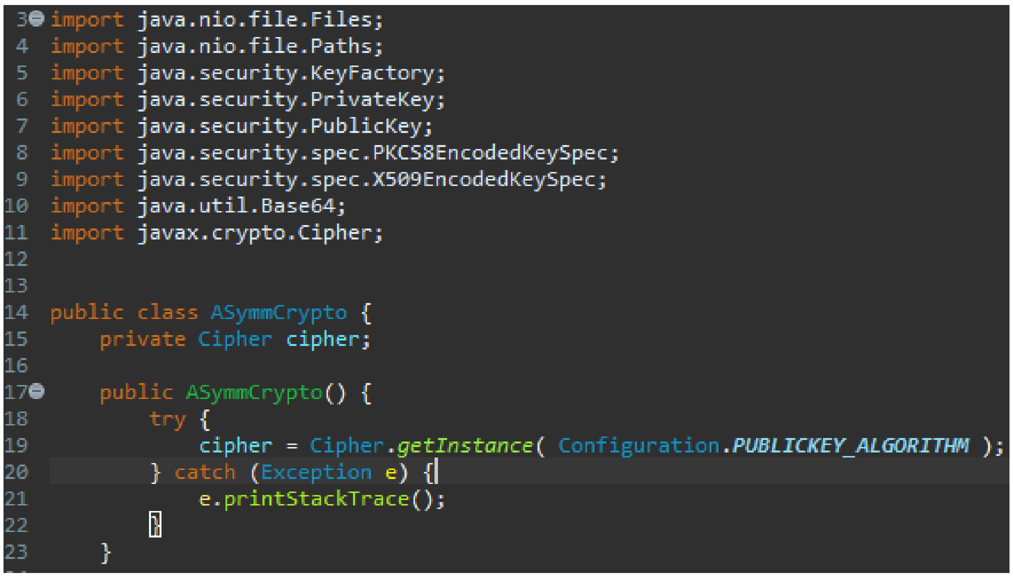
The system will display the details of the car selected by the user from CarSelection class. The user is redirect to the CarBooking class to select the start date and end date. The system will alert the users to enter the correct date when start date and end date is not selected. Besides, the system will also check if the start date and end date are today date onwards. After the date is verified, the pending verification of encrypted booking data is stored in the database shown in line 235. The data is encrypted in line 223 to 225 and it is stored in the database in line 230.



```
boolean flag = true;
if(textSignature.getText().isEmpty()) {
    JOptionPane.showMessageDialog(frame, "No valid booking selected!");
    flag=false;
}
if(flag) {
ASymmCrypto asc = new ASymmCrypto();
//ArrayList<String> idList = new ArrayList<String>();
List<String> bookData = IO.read(FILE_BOOKDATA);
for (int i = 0; i < bookData.size(); i++) {
    String[] split = bookData.get(i).split("\\|");
    String checkBookingID = split[0];
        if (!(checkBookingID.equals(bookingID))) {
            IO.write(FILE_TEMP, bookData.get(i));

        }
        else {
            try {
                decryptedTxt = asc.decrypt(split[1], bookingID);
                String[] split2 = decryptedTxt.split("\\|");
                userID = split2[0];
                carID = split2[1];
                startDate = split2[2];
                endDate = split2[3];
                split2[4] = "Accepted";
                String encryptedTxt = asc.encrypt(String.join("|",split2[0],split2[1],split2[2],split2[3],split2[4]),bookingID)
                String sJoin = String.join("|", bookingID, encryptedTxt);
                IO.write(FILE_TEMP, sJoin);
            } catch (Exception e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
        }
    }
IO.delete(FILE_BOOKDATA);
IO.rename(FILE_TEMP,"bookingData.txt");
JOptionPane.showMessageDialog(frame, "Booking approved");
```

*Figure 46: VerifyBooking.java.*

The encryption and decryption of data are implemented at booking validation during the selection of pending booking, accept status and decline status of the booking. The selection function in VerifyBooking class will display the booking details and allow the validators to verify the booking using digital signature that is discuss at the following section. Line 355 shows that the decrypt method is called to retrieve the data and line 362 shows the encryption of the data to include the updated status of booking in VerifyBooking class.

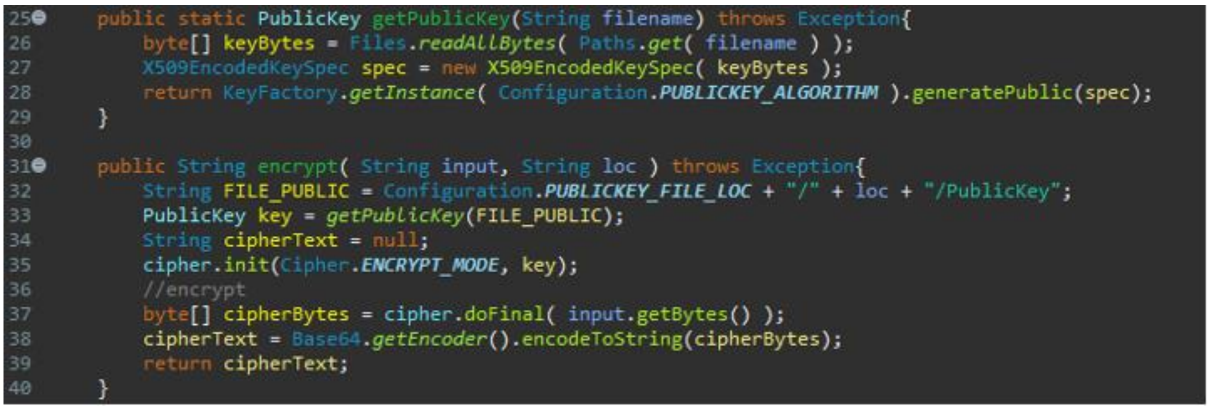


```java
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import javax.crypto.Cipher;


public class ASymmCrypto {
    private Cipher cipher;

    public ASymmCrypto() {
        try {
            cipher = Cipher.getInstance( Configuration.PUBLICKEY_ALGORITHM );
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

*Figure 47: ASymmCrypto.java.*

The Cipher class will provide encryption and decryption which is also the core of the Java Cryptographic Extension (JCE) framework. The ASymmCrypto constructor calls the *javax.crypto.Cipher* class to create a cipher object from the transformation parameters for the getInstance method using RSA algorithm.

## Encryption using public key

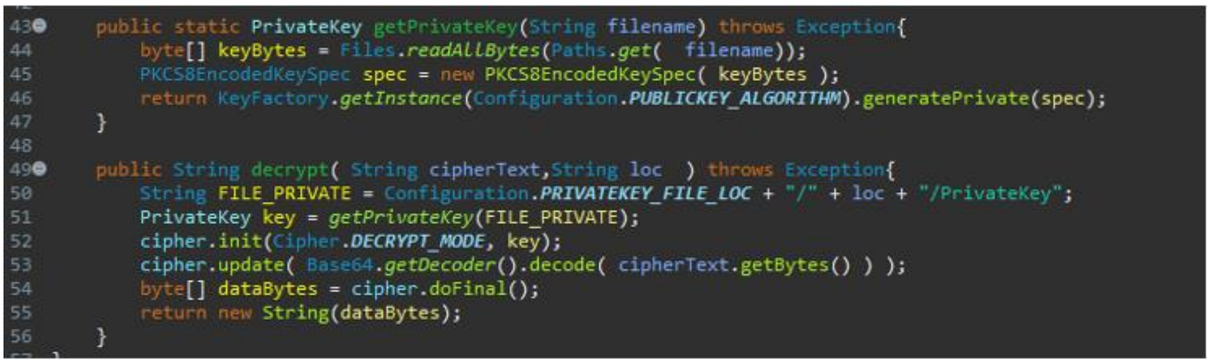


```java
    public static PublicKey getPublicKey(String filename) throws Exception{
        byte[] keyBytes = Files.readAllBytes( Paths.get( filename ) );
        X509EncodedKeySpec spec = new X509EncodedKeySpec( keyBytes );
        return KeyFactory.getInstance( Configuration.PUBLICKEY_ALGORITHM ).generatePublic(spec);
    }

    public String encrypt( String input, String loc ) throws Exception{
        String FILE_PUBLIC = Configuration.PUBLICKEY_FILE_LOC + "/" + loc + "/PublicKey";
        PublicKey key = getPublicKey(FILE_PUBLIC);
        String cipherText = null;
        cipher.init(Cipher.ENCRYPT_MODE, key);
        //encrypt
        byte[] cipherBytes = cipher.doFinal( input.getBytes() );
        cipherText = Base64.getEncoder().encodeToString(cipherBytes);
        return cipherText;
    }
```

*Figure 48: ASymmCrypto.java.*

The encrypt method from ASymmCrypto class is called in CarBooking class and VerifyBooking class. The public key is used for encryption by calling the path directory of the public key in line 32. Line 27 and 28 for the getPublicKey method will return byte array from the file and the key is encoded using X.509 standard to prevent modification. The *java.security.KeyFactory* library is used to generate public key from the key factory object in line 28. The public key will encrypt the data inputted from line 32 to 38 and the cipher text is returned as String format.

**Decryption using private key**

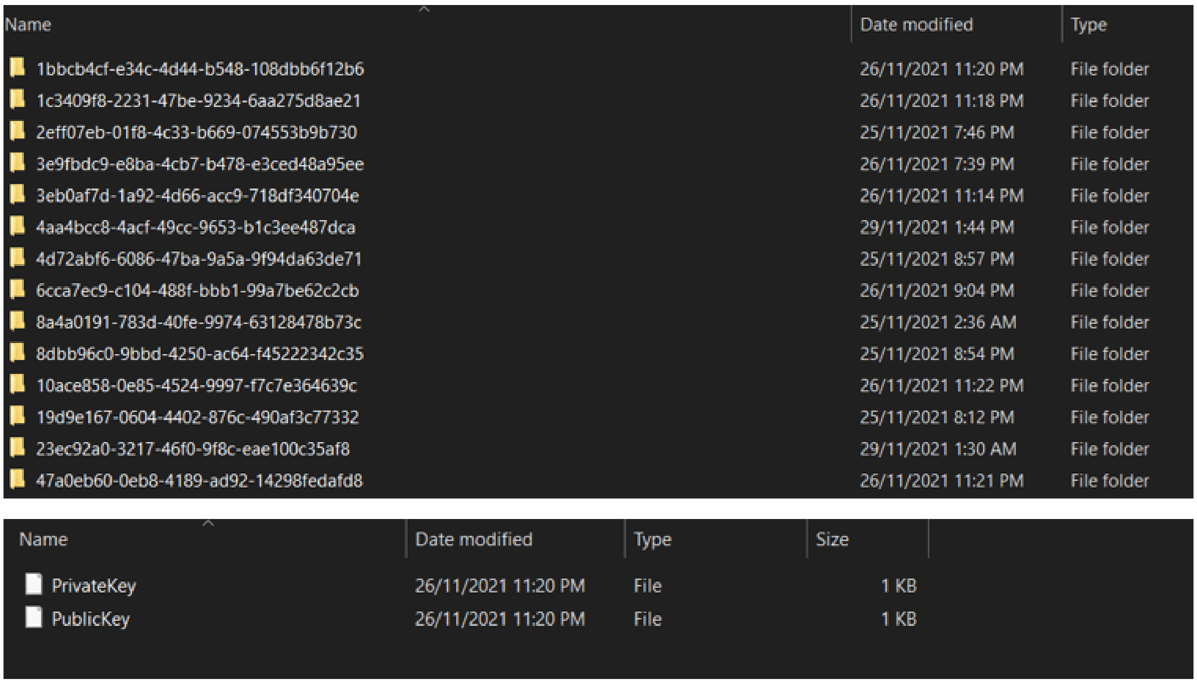


```
43     public static PrivateKey getPrivateKey(String filename) throws Exception{
44         byte[] keyBytes = Files.readAllBytes(Paths.get(  filename));
45         PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec( keyBytes );
46         return KeyFactory.getInstance(Configuration.PUBLICKEY_ALGORITHM).generatePrivate(spec);
47     }
48
49     public String decrypt( String cipherText,String loc  ) throws Exception{
50         String FILE_PRIVATE = Configuration.PRIVATEKEY_FILE_LOC + "/" + loc + "/PrivateKey";
51         PrivateKey key = getPrivateKey(FILE_PRIVATE);
52         cipher.init(Cipher.DECRYPT_MODE, key);
53         cipher.update( Base64.getDecoder().decode( cipherText.getBytes() ) );
54         byte[] dataBytes = cipher.doFinal();
55         return new String(dataBytes);
56     }
```

*Figure 49: ASymmCrypto.java.*

The decrypt method of the data is called from the VerifyBooking class. The private key is used for decryption by also calling the path directory of the private key generated. The array of bytes from the private key is obtained and the bytes is encoded using the PKCS #8 standard in line 44 to 45. The *java.security.KeyFactory* library is used to return private key from the key factory object in line 46. The decryption of the data is performed and the ciphertext is decided to data bytes which is converted into String format from line 50 to 55.

## Storing of the public and private keys



| Name | Date modified | Type |
|---|---|---|
| 1bbcb4cf-e34c-4d44-b548-108dbb6f12b6 | 26/11/2021 11:20 PM | File folder |
| 1c3409f8-2231-47be-9234-6aa275d8ae21 | 26/11/2021 11:18 PM | File folder |
| 2eff07eb-01f8-4c33-b669-074553b9b730 | 25/11/2021 7:46 PM | File folder |
| 3e9fbdc9-e8ba-4cb7-b478-e3ced48a95ee | 26/11/2021 7:39 PM | File folder |
| 3eb0af7d-1a92-4d66-acc9-718df340704e | 26/11/2021 11:14 PM | File folder |
| 4aa4bcc8-4acf-49cc-9653-b1c3ee487dca | 29/11/2021 1:44 PM | File folder |
| 4d72abf6-6086-47ba-9a5a-9f94da63de71 | 25/11/2021 8:57 PM | File folder |
| 6cca7ec9-c104-488f-bbb1-99a7be62c2cb | 26/11/2021 9:04 PM | File folder |
| 8a4a0191-783d-40fe-9974-63128478b73c | 25/11/2021 2:36 AM | File folder |
| 8dbb96c0-9bbd-4250-ac64-f45222342c35 | 25/11/2021 8:54 PM | File folder |
| 10ace858-0e85-4524-9997-f7c7e364639c | 26/11/2021 11:22 PM | File folder |
| 19d9e167-0604-4402-876c-490af3c77332 | 25/11/2021 8:12 PM | File folder |
| 23ec92a0-3217-46f0-9f8c-eae100c35af8 | 29/11/2021 1:30 AM | File folder |
| 47a0eb60-0eb8-4189-ad92-14298fedafd8 | 26/11/2021 11:21 PM | File folder |

| Name | Date modified | Type | Size |
|---|---|---|---|
| PrivateKey | 26/11/2021 11:20 PM | File | 1 KB |
| PublicKey | 26/11/2021 11:20 PM | File | 1 KB |

*Figure 50: Private and public key files generated in the folder directory.*

Each folder directory generated have its unique public and private keys used for encryption and decryption of the booking data. The files are retrieved by specifying the file directory explained in the code implementation above.

## 4.3 Digital Signature

### 4.3.1 Overview

Digital signature is refereed as the electronic fingerprint in digital coded format by associating the signer with the document or data recorded. The digital signature uses the Public Key Infrastructure (PKI) requirements that includes private key and public key. The private key is used to digitally sign the documentation from the hash generated in the documents, whilst the public key is only available for the decryption on the signed document to be validated. The matching hash will ensure the data is not tampered or modified.

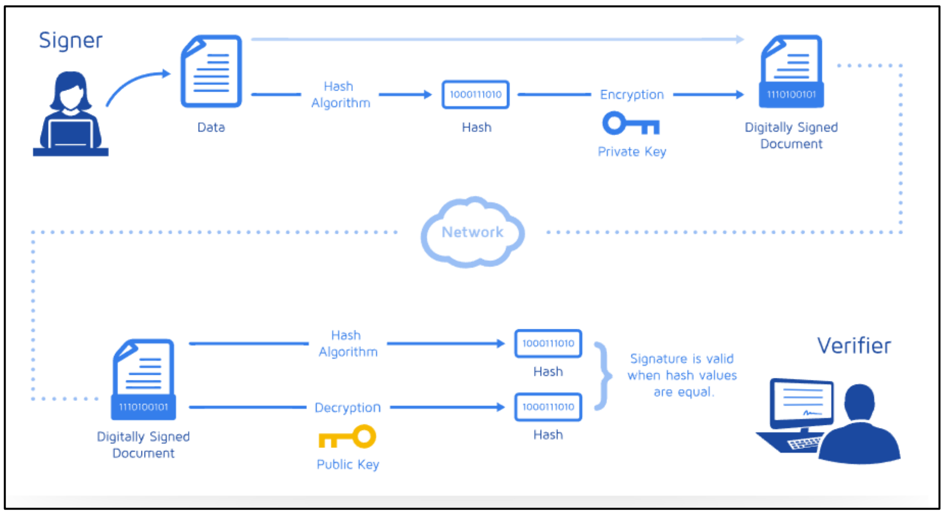


*Figure 51:Illustration of digital signature implementation (DocuSign, 2021).*

The asymmetric cryptography is used for the implementation of the digital signature. The digital signature provides security features of security and validation for message sent over insecure network channels. The digital signature is broken down into 3 algorithms, also known as 3-tuple algorithms, which are the generation of keys, digital signature and verification for the authenticity of the document (Seetha, 2017). The generation and signing is probabilistic, and the verification is deterministic (Ferradi, 2016). The Digital Signature Algorithm (DSA) is developed by the National Institute of technology.

The document is hashed before signing process as it increases the efficiency as the hashing algorithm performs faster than the signing algorithm. Besides, a shorter signature will be generated after hashing. The hashing process will allow the data to be converted in desired format for compatibility on RSA algorithm used. The document or data will require to be split into smaller sections for the digital signature if hashing algorithm is not implemented.

### 4.3.2 Benefits of digital signature

The advantages of the implementation digital signature can be compared with the electronically-sign documents process. The digital signature is trusted with consent, highly secure and provides better convenience.

The digital signature that is generated uses the user credentials to uniquely identify the specific individual when signing the electronic document. The signing of the document is also giving consent for the signature for confirmation of identity. The signer of the document is legally bind with compliant with laws, supporting with written evidence of the signer’s identity. The ownership of the digital secret key will be bound to the specific user. The authenticity is important in financial context that require high confidence.

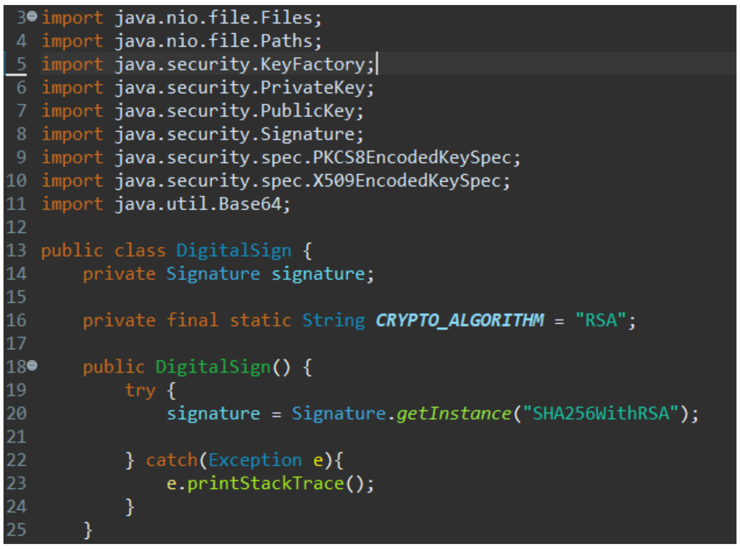
Besides, digital signature allows better protection and security on validating any changes in the electronic document. There is possibility on changing the encrypted message using encryption tools without understanding the original message, but the digital signature will invalidate the message that have been altered. There is no efficient way and proven to be infeasible to reproduce the message and the signature for replacing as valid signature.

Cost effectiveness and time saving is another benefit of using digital signature for documents. The digital signature can replace the need for physical copies by saving time for manually signing the documents like printing, arranging appointments, scanning documents and sending to the respective parties. The cost of manual paper work can be reduced significantly as digital signature allows the verifier to validate the unique digital signature generated.

## 4.3.3 Source code and implementation of digital signature in business

In this section, the implementation of signing and verifying the digital signature is implemented. The source code for the digital signature and screenshots of the system is provided below.

### Initialization

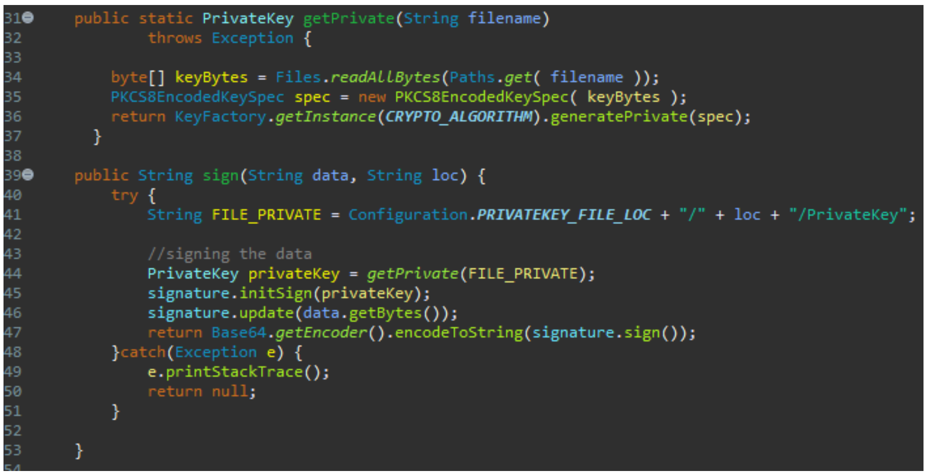


```java
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;

public class DigitalSign {
    private Signature signature;

    private final static String CRYPTO_ALGORITHM = "RSA";

    public DigitalSign() {
        try {
            signature = Signature.getInstance("SHA256WithRSA");

        } catch(Exception e){
            e.printStackTrace();
        }
    }
```

*Figure 52: DigitalSign.java*

The implementation of digital signature is shown in DigitalSign class by importing the libraries required, the initialization of cryptographic algorithm of RSA and signature using SHA-256 with RSA. The *java.security.PrivateKey* and *java.security.PublicKey* libraries are used for generation of both the private key and private key respectively. The *java.security.KeyFactory* library is used to create bi-directional key for translation for compatible key translation. The *java.security.spec.PKCS8EncodedKeySpec* library is used for encoding bytes by private key, while *java.security.spec.X509EncodedKeySpec* library is used for encoding bytes by public key. Both the libraries are using abstract syntax notation one (ASN.1) for cryptography.

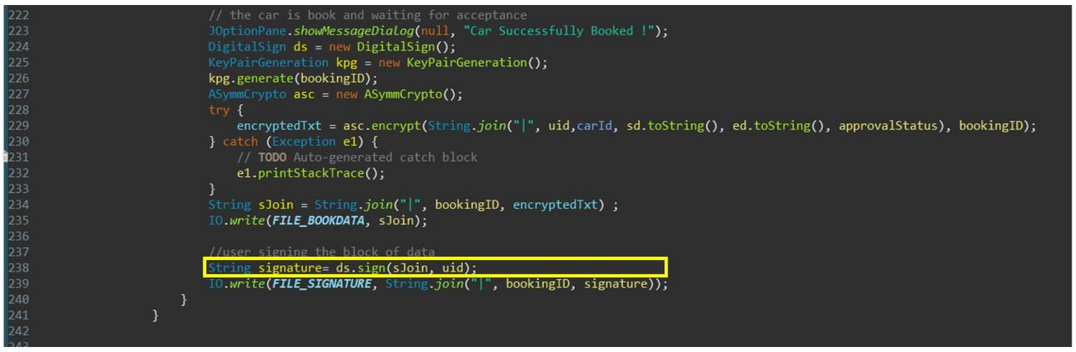
## Signing of digital signature



```
31   public static PrivateKey getPrivate(String filename)
32           throws Exception {
33
34       byte[] keyBytes = Files.readAllBytes(Paths.get( filename ));
35       PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec( keyBytes );
36       return KeyFactory.getInstance(CRYPTO_ALGORITHM).generatePrivate(spec);
37     }
38
39   public String sign(String data, String loc) {
40       try {
41           String FILE_PRIVATE = Configuration.PRIVATEKEY_FILE_LOC + "/" + loc + "/PrivateKey";
42
43           //signing the data
44           PrivateKey privateKey = getPrivate(FILE_PRIVATE);
45           signature.initSign(privateKey);
46           signature.update(data.getBytes());
47           return Base64.getEncoder().encodeToString(signature.sign());
48       }catch(Exception e) {
49           e.printStackTrace();
50           return null;
51       }
52
53   }
```

Figure 53: DigitalSign.java.

The pending booking data is digitally signed in the sign method from the DigitalSign class. The location of the stored the private key is retrieved and the private key is obtained based on the file path in line 41. The getPrivate method will generate the key specification of the private key in line 34 to 36 using RSA algorithms. The signature object will generate the private key with SHA-256 algorithm with RSA in line 45 and it is updated with the data input in bytes format. Line 47 shows the signature bytes of the private key is encoded to String format in Base 64 and the value is returned.

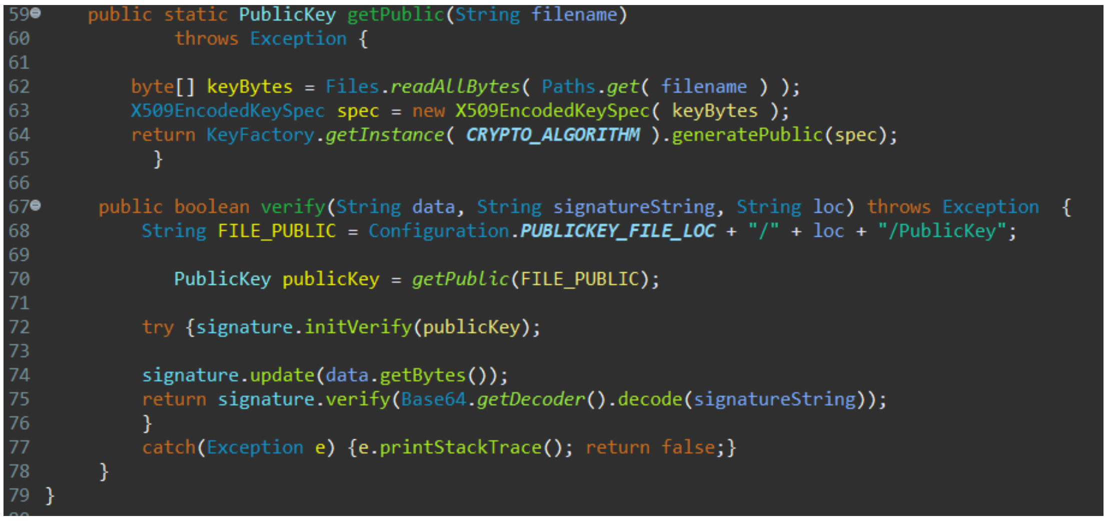


```
222                 // the car is book and waiting for acceptance
223                 JOptionPane.showMessageDialog(null, "Car Successfully Booked !");
224                 DigitalSign ds = new DigitalSign();
225                 KeyPairGeneration kpg = new KeyPairGeneration();
226                 kpg.generate(bookingID);
227                 ASymmCrypto asc = new ASymmCrypto();
228                 try {
229                     encryptedTxt = asc.encrypt(String.join("|", uid,carId, sd.toString(), ed.toString(), approvalStatus), bookingID);
230                 } catch (Exception e1) {
231                     // TODO Auto-generated catch block
232                     e1.printStackTrace();
233                 }
234                 String sJoin = String.join("|", bookingID, encryptedTxt) ;
235                 IO.write(FILE_BOOKDATA, sJoin);
236
237                 //user signing the block of data
238                 String signature= ds.sign(sJoin, uid);
239                 IO.write(FILE_SIGNATURE, String.join("|", bookingID, signature));
240             }
241         }
242
```

Figure 54: CarBooking.java.

The digital signature is generated after the user successfully booking for the car rental. The booking details containing the user id, car id, start date, end date and the pending approval are encrypted in line 229. A digital signature is generated by calling the sign method from the encrypted data of the in line 238.

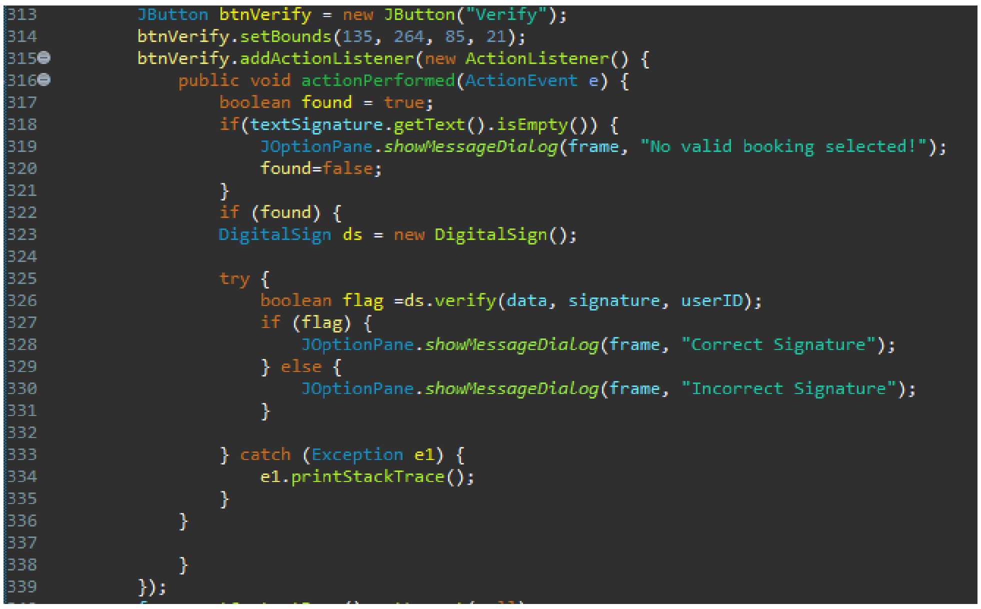
## Verification of digital signature



```
59      public static PublicKey getPublic(String filename)
60              throws Exception {
61
62          byte[] keyBytes = Files.readAllBytes( Paths.get( filename ) );
63          X509EncodedKeySpec spec = new X509EncodedKeySpec( keyBytes );
64          return KeyFactory.getInstance( CRYPTO_ALGORITHM ).generatePublic(spec);
65            }
66
67       public boolean verify(String data, String signatureString, String loc) throws Exception  {
68           String FILE_PUBLIC = Configuration.PUBLICKEY_FILE_LOC + "/" + loc + "/PublicKey";
69
70              PublicKey publicKey = getPublic(FILE_PUBLIC);
71
72           try {signature.initVerify(publicKey);
73
74           signature.update(data.getBytes());
75           return signature.verify(Base64.getDecoder().decode(signatureString));
76           }
77           catch(Exception e) {e.printStackTrace(); return false;}
78       }
79 }
```

*Figure 55: DigitalSign.java.*

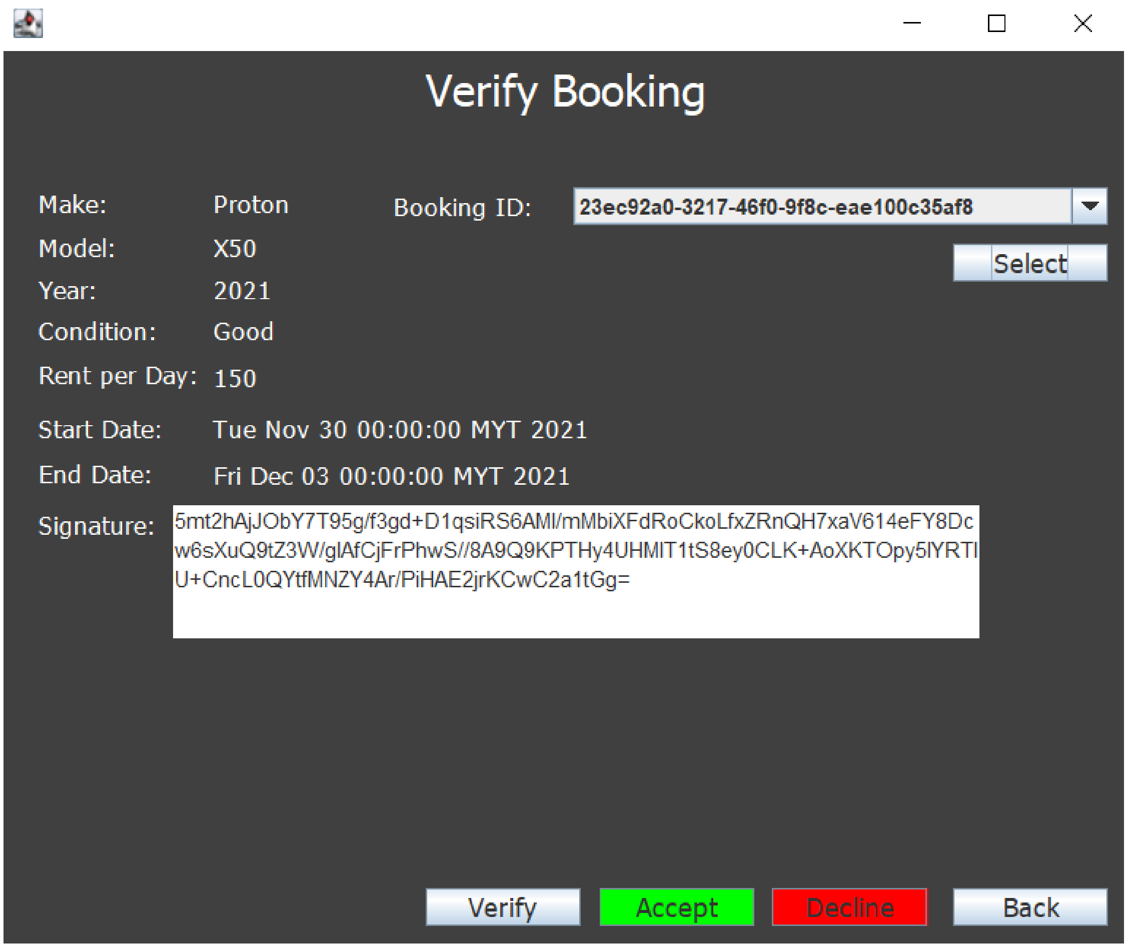
The verify method will receive the parameters of the data and the location of the stored public key in VerifyBooking class. The location of the public key is retrieved and obtained from the line 69. The getPublic method will generate the key specification of public key in line 62-64 using RSA algorithm. The signature object will initialize the object for verification in line 72 and the data is updated in bytes format in line 74. The signature object then calls the verify method to return true if the signature is verified and false if the signature is not verified.



```
        JButton btnVerify = new JButton("Verify");
        btnVerify.setBounds(135, 264, 85, 21);
        btnVerify.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                boolean found = true;
                if(textSignature.getText().isEmpty()) {
                    JOptionPane.showMessageDialog(frame, "No valid booking selected!");
                    found=false;
                }
                if (found) {
                DigitalSign ds = new DigitalSign();

                try {
                    boolean flag =ds.verify(data, signature, userID);
                    if (flag) {
                        JOptionPane.showMessageDialog(frame, "Correct Signature");
                    } else {
                        JOptionPane.showMessageDialog(frame, "Incorrect Signature");
                    }

                } catch (Exception e1) {
                    e1.printStackTrace();
                }
            }

            }
        });
```

*Figure 56: VerifyBooking.java.*

Line 318 to 321 would check if any booking is selected for verification The data is verified by calling the method verify from DigitalSign Class and return a true or false value in line 326. The system will alert the user if the return value is true for correct signature, and the return value is false for incorrect signature from line 327 to 331.



# Verify Booking

| | |
|---|---|
| Make: | Proton |
| Model: | X50 |
| Year: | 2021 |
| Condition: | Good |
| Rent per Day: | 150 |
| Start Date: | Tue Nov 30 00:00:00 MYT 2021 |
| End Date: | Fri Dec 03 00:00:00 MYT 2021 |
| Signature: | 5mt2hAjJObY7T95g/f3gd+D1qsiRS6AMl/mMbiXFdRoCkoLfxZRnQH7xaV614eFY8Dc w6sXuQ9tZ3W/glAfCjFrPhwS//8A9Q9KPTHy4UHMIT1tS8ey0CLK+AoXKTOpy5lYRTI U+CncL0QYtfMNZY4Ar/PiHAE2jrKCwC2a1tGg= |
| Booking ID: | 23ec92a0-3217-46f0-9f8c-eae100c35af8 |

*Figure 57: Verification of digital signature in car rental system.*

The verification page in the system shows the implementation of the verification and the approval mechanism. The unique digital signature is displayed corresponding to the booking ID selected.

## 5.0 Evaluation of developed solution

The proposed blockchain car rental system is evaluated to determine the viability and advantage to implement blockchain technology into car rental system.

### 5.1 Security

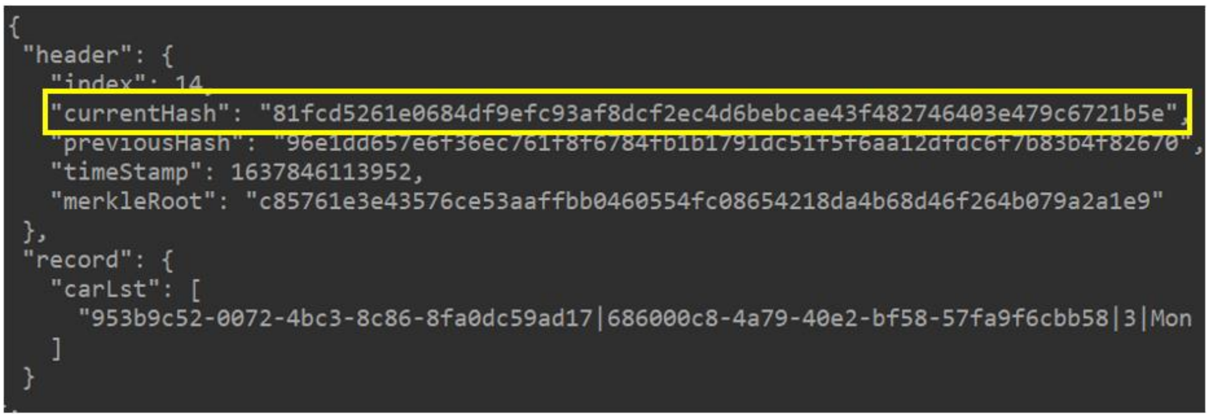
Data security within the software system is a crucial focal point for many software developments in any industry, including the automotive industry. Without proper data security, data breaches could occur, which increase the risk of sensitive data exposure. In the long term, lack of data security will result in businesses losing trust with their customers and clients, resulting in poor business reputation. As such, the implementation of blockchain technology can improve and ensure data security due to the nature of blockchain that changes the way data is viewed and accessed.

For the proposed system, hashing algorithm using SHA-256 is implemented for login credentials, which secures the login password as the hash code for the passwords cannot be modified or altered, thus stealing the hash code will not grant unauthorized access to the user's account as the hash code cannot be applied in any other encryption scheme. On top of that, salt technique is also used for hashing password in the car rental system, which provides an additional input for secure the hashed password by using a randomly generated data. This technique can create a unique password even if two users are using the same password as well as protecting the hashed password against brute force attacks as it creates another level of computational requirements to obtain the password. Hash value is also generated for blocks when a car booking request is approved, which ensure the security of the blockchain.

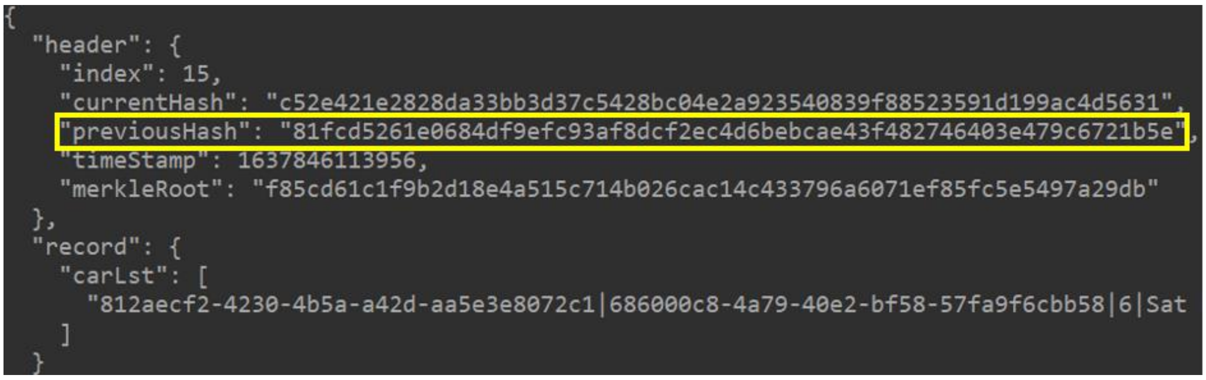
Besides that, cryptography is also used in the car rental system to conceal the actual data value from trespassers. When the information regarding the vehicle booking request is saved, it is saved in encrypted format, also known as ciphertext, which prevents unauthorized users from accessing the actual information even if they somehow managed to obtain the raw data. This is done by using public key and private key, in which the public key is used by to encrypt the booking request made by the clients into ciphertext, while the private key is used by the management to decrypt the cipher text to obtain the actual information. By doing so, the data transportation can be performed in a secured environment.

## 5.2 Transparency

Greater transparency can be obtained by implementing blockchain technology in the car rental system. By using blockchain, distributed ledger is sued to record identical transactions and data across multiple locations. This allows anyone within the connected network to view the information. For the car rental system, digital ledger is applied on the approved booking request. Any vehicle that was confirmed to be rented will be included in the digital ledger through blockchain.



```
{
 "header": {
   "index": 14,
   "currentHash": "81fcd5261e0684df9efc93af8dcf2ec4d6bebcae43f482746403e479c6721b5e",
   "previousHash": "96e1dd657e6f36ec761f8f6784fb1b1791dc51f5f6aa12dfdc6f7b83b4f82670",
   "timeStamp": 1637846113952,
   "merkleRoot": "c85761e3e43576ce53aaffbb0460554fc08654218da4b68d46f264b079a2a1e9"
 },
 "record": {
   "carLst": [
     "953b9c52-0072-4bc3-8c86-8fa0dc59ad17|686000c8-4a79-40e2-bf58-57fa9f6cbb58|3|Mon
   ]
 }
```



```
{
  "header": {
    "index": 15,
    "currentHash": "c52e421e2828da33bb3d37c5428bc04e2a923540839f88523591d199ac4d5631",
    "previousHash": "81fcd5261e0684df9efc93af8dcf2ec4d6bebcae43f482746403e479c6721b5e",
    "timeStamp": 1637846113956,
    "merkleRoot": "f85cd61c1f9b2d18e4a515c714b026cac14c433796a6071ef85fc5e5497a29db"
  },
  "record": {
    "carLst": [
      "812aecf2-4230-4b5a-a42d-aa5e3e8072c1|686000c8-4a79-40e2-bf58-57fa9f6cbb58|6|Sat
    ]
  }
```

*Figure 58: Comparison of the hash value of the corresponding blocks.*

Each request will be one block linked with other blocks to form a blockchain containing all the details from every other request made. Each block will also contain the time stamp of the record as well as the hashed values and previous hash value. This would ensure that blocks are immutable as changes on any of the block would result in different hash values, which results in the connecting node to be altered, thus breaking the away from the blockchain. By doing so, the entire car rental transaction history can be viewed by members within the network, thus eliminating any risks of fraudulent activities.

## 5.3 Accessibility

The proposed blockchain car rental system uses digital signature whenever clients perform booking request. Each booking request made will be recorded and sign by the client’s unique signature, ensuring that the information signed on are not immutable. By using digital blockchain, the message to be sent will be confirmed, and authentication can be done by approving the digital signature with a generated private key from the sender. In the case of the car rental system, the digital signature can be accessed and verified by the management through the client’s private key. This ensure that the clients are authentic and prevent fraudulent activities through stealing of identities, thus resulting in a more secure access to information within the system.

# 6.0 Conclusion

The research project started with discussion about the characteristics of different blockchain technology and the hybrid blockchain car rental management system is chosen. The system architecture and data management are briefly explained in the findings section. The implementation of the blockchain technology with compromise of the hashing, digital signature and cryptography provides immutability, security, transparency, and accessibility for the system. The car rental management system with blockchain is developed based on blockchain technology features which provides tremendous benefits for the automotive industry. Some improvements on the system are incorporating the insurance provides, tracking of user's overall usage and repair damages, integrating fuel cards and allowing payment using cryptocurrency. The actors that can be included in future development are the repair shops, dealerships, insurances providers and fuel card providers. The developers have gained a comprehensive understanding of the blockchain technologies from research, development and implementation of the concepts that will be beneficial for future endorsement in blockchain field.

## Reference

Cortez, D. M. A., Sison, A. M., & Medina, R. P. (2020). Cryptanalysis of the Modified SHA256. *PervasiveHealth: Pervasive Computing Technologies for Healthcare*, 179–183. https://doi.org/10.1145/3409501.3409513

Davies, S. (2020). *Trust, transparency, efficiency: The trillion-dollar reasons to rethink blockchain - Technology Insights - PwC UK blogs*. https://pwc.blogs.com/technology-insights/2020/10/trust-transparency-efficiency-the-trillion-dollar-reasons-to-rethink-blockchain.html

Dhar, S., Pandey, K., Premalatha, M., & Suganya, G. (2018). A tree based approach to improve traditional collision avoidance mechanisms of hashing. *Proceedings of the International Conference on Inventive Computing and Informatics, ICICI 2017*, *Icici*, 339–342. https://doi.org/10.1109/ICICI.2017.8365368

DocuSign. (2021). *How Digital Signatures Work | DocuSign*. https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq

Du, L., He, Z., Wang, Y., Wang, X., & Ho, A. T. S. (2020). An image hashing algorithm for authentication with multi-attack reference generation and adaptive thresholding. *Algorithms*, *13*(9). https://doi.org/10.3390/A13090227

Ferradi, H. (2016). Integrity, authentication and confidentiality in public-key cryptography. *HAL*. https://tel.archives-ouvertes.fr/tel-01745919

InfoSec. (2020). *Types of Encryption: What to Know About Symmetric vs Asymmetric Encryption*. https://sectigostore.com/blog/types-of-encryption-what-to-know-about-symmetric-vs-asymmetric-encryption/

KeyTool. (2021). *Keytool - Generate/convert cryptographic keys*. https://keytool.online/

Liu, D., & Xu, S. (2015). Comparison of hash table performance with open addressing and closed addressing: An empirical study. *International Journal of Networked and Distributed Computing*, *3*(1), 60–68. https://doi.org/10.2991/ijndc.2015.3.1.7

Mostafa, M. (2019). *What is salting? | Huawei Enterprise Support Community*. https://forum.huawei.com/enterprise/en/what-is-salting-cyber-security-awareness/thread/492569-867

PwC. (2020). *Businesses are benefiting from the use of blockchain*. https://www.pwc.co.uk/blockchain.html

Rachmawati, D., Tarigan, J. T., & Ginting, A. B. C. (2018). A comparative study of Message Digest 5(MD5) and SHA256 algorithm. *Journal of Physics: Conference Series*, *978*(1). https://doi.org/10.1088/1742-6596/978/1/012116

Rathod, U., Sonkar, M., & Chandavarkar, B. R. (2020). An Experimental Evaluation on the Dependency between One-Way Hash Functions and Salt. *2020 11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020, October*. https://doi.org/10.1109/ICCCNT49239.2020.9225503

Rojasree, V., Jayanthi, G., & Sujatha, C. (2020). *Research Intuitions of Hashing Crypto System – IJERT*. International Journal of Engineering Research & Technology (IJERT) Volume 09, Issue 12 (December 2020). https://www.ijert.org/research-intuitions-of-hashing-crypto-system

Seetha, R. (2017). An enhanced digital signature scheme. *International Journal of Applied Engineering Research*, *12*(22), 11878–11884.

Simmons, G. J. (2019). Symmetric and asymmetric encryption. In *Secure Communications and Asymmetric Cryptosystems* (pp. 241–298). https://doi.org/10.36074/05.06.2020.v3.36

Yusuf, A. D., Abdullahi, S., Boukar, M. M., & Yusuf, S. I. (2021). Collision Resolution Techniques in Hash Table: A Review. *International Journal of Advanced Computer Science and Applications*, *12*(9), 757–762. https://doi.org/10.14569/ijacsa.2021.0120984