electronics

MDPI

Article

Device Identity-Based User Authentication on Electronic Payment System for Secure E-Wallet Apps

Md Arif Hassan * and Zarina Shukur

Center for Cyber Security, Faculty of Information Science and Technology, National University Malaysia (UKM), Bangi 43600, Malaysia; zarinashukur@ukm.edu.my

* Correspondence: arifhassane72@gmail.com; Tel.: +60-102-483-220

Abstract: E-wallets are a modern electronic payment system technology that easily recognize consumer interest, making our transactions very convenient and efficient. E-wallets are intended to substitute the existing physical wallet, which may tell others something about us as a person. That is why using a physical wallet is a unique, personal experience that cannot be duplicated. A solution would be to replace the physical wallet with an e-wallet on an existing mobile device. The personal nature of the e-wallet is that it should be installed on a unique device. One of the fundamental protections against any illegal access to e-wallet application is through authentication. In particular, the fundamental authentication category used in an existing e-wallet is based on knowledge (i.e., what you know), ownership (i.e., what you have), and biometric (i.e., what you are) authentication, which are sometimes prone to security threats such as account takeover, sim swapping, app cloning, or know your customer verification attacks. The design of an e-wallet authentication on mobile device solution must take into consideration the intensity of the security. To address this problem, this study proposes a design of e-wallet apps with an extension security element that focuses on the device identity in the existing user authentication mechanism. This study covers four fundamental categories of authentication: password, one time password, fingerprints, and international mobile equipment identifier. Using IMEI limits an e-wallet to be in one specific device in one time; this brings it into line with the nature of a physical wallet. In addition, it will be ready to handle the mentioned threats above, which will ultimately result in the far more reliable to use of e-wallet apps. The proposed authentication design has two phases, a registration phase and an authentication phase. The proposed method has been developed and implemented based on an Android Studio Firebase real-time database management and PayPal. In addition, the complete design has been evaluated using functional requirement testing to see how closely it meets functionality requirements. The results obtained from functional testing show that the functionalities of the proposed method meet the requirements, and one cannot use a same account on two devices; hence, it is secure from attacks. The result also shows that the proposed method has no errors. Moreover, it has been shown that our proposed method has better security parameters in terms of the existing method.

Keywords: electronic payments; e-wallet; knowledge; ownership; multifactor authentication

check for updates

Citation: Hassan, M.A.; Shukur, Z. Device Identity-Based User Authentication on Electronic Payment System for Secure E-Wallet Apps. *Electronics* **2022**, *11*, 4. https://doi.org/10.3390/electronics11010004

Academic Editors: Laith Farhan, Khaled Rabie, Xingwang Li and Waleed Ejaz

Received: 30 October 2021
Accepted: 8 December 2021
Published: 21 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The concept of electronic wallets is not new. In certain countries such as Japan, the e-wallet has been in popular usage as early as 2004 [1,2]. An electronic wallet, sometimes called a "digital wallet" or "e-wallet", is an electronic adaptation of a payment card that is approved for electronic exchanges [3–5]. The e-wallet payment system uses sophisticated authentication methods to enhance protection. The identity of the user must be checked, and only authenticated users are allowed to use the system. To secure user data and privacy, the development of trustworthy e-wallet authentication is an extremely important challenge. Authentication is the principal element associated with e-wallet payment. Currently, knowledge-based authentication is commonly used throughout the community, as it is distinctive and operator-friendly [6,7].

Authentication based on knowledge (KBA) offers access to the device using a single authentication process. There are security issues related to knowledge-based authentication such as password schemes [8], where online consumers are required to pay in a single click. Password-based authentication is highly vulnerable to manipulation, where intruders could use different methods to steal passwords, and it is not suitable for all users [9]. Ownership-based authentication has been seen as a solution for e-wallet transactions to mitigate the issues of knowledge-based authentication. In previous studies, the use of authentication that is dependent on ownership-based factors was limited and did not stop fraud [10]. The use of ownership-based authentication suffers from token forgery [8,11,12] and insider attacks [13,14]. Due to this security flaw, fraudsters who provide details of account holders to make unauthorized transactions can exploit users' accounts. A significant challenge for the financial industry is the avoidance of fraudulent transactions involving e-wallets. Hence, while using the current authentication mechanisms, distinguishing between fraudulent and authorized transactions is quite challenging.

A reliable multifactor authentication framework is required to verify the validity of the user. To mitigate this limitation, several researchers suggested and implemented a multifactor authentication technique. A few researchers focused on biometric techniques with various authentication elements, including fingerprinting and facial recognition. Authentication capabilities are still limited, and due to the difficulty in implementing them, some are not used, while others are only framework-based and not implemented in real life. Furthermore, improvements are needed in the method of authentication based on device identity using (give the full form of IMEI here) IMEI. Therefore, the existing authentication mechanisms need to be enhanced with device identity-based multifactor authentication features to increase the security of e-wallet apps.

This study resolves the above-stated problems through the actualization of the research question. The following are the specific research questions that will be addressed: (RQ1) "What are the design considerations for secure e-wallet apps based on device identity?"; and (RQ2) "What are the tools and methods to assess device identity-based e-wallet apps?".

Consequently, this study proposes e-wallet (Zamwallet) apps that cover four authentication categories for the verification of a user. The key contribution of this study is that the nature of the e-wallet is such that it should be installed on a unique device. As a result, the proposed Zamwallet cannot run on two devices simultaneously and will have a security protocol that is designed and implemented with the inclusion of four fundamental authentication categories.

This paper is divided into seven sections. Introducing electronic payments and its related study, which is already discussed above, is the Section 1. Section 2 will include an overview of the existing system and related work. The Section 3 is the description of the proposed conceptual design. Section 4 presents how the model would be implemented. Section 5 presents the results of the proposed method and analysis, while Section 6 presents the discussion. The Section 7 is the conclusion.

2. Literature Review

Authentication based on knowledge offers access to the device using a single authentication process [15]. A majority of the user authentication and even bank access controls for internet banking systems are based on KBA due to its features and familiarity to the provider and user [6]. However, the KBA is prone to different kinds of attacks such as brute-force attacks [16–18], rainbow table attacks [19,20], dictionary attacks [21–23], social engineering [10,24–26], phishing [27,28], framework for man-in-the-middle attacks (MITMF) [29,30], password-based attacks [16,31], session hijacking [32], and malware [28,33,34].

Furthermore, authentication with only a password was recognized as not being secure enough to provide protection due to a variety of security threats [24]. User login tests have found that 86% of user-selected passwords are insecure [35]. To overcome the KBA issue, ownership-based authentication was considered a solution for securing, authenticating, and verifying online transactions for a device or application. Authentication by ownership

is a method that implements more than one factor and is known to be stronger and safer than the KBA. Authentication by two factors such as tokens or cards has proven effective and difficult to exploit [15,36–38].

Although ownership usage is strong, malicious assaults such as lost/stolen cards, token expenses, token forgery, and token losses continue to affect them [8,39]. As e-wallet transactions increase, ownership is not sufficient for secure transactions [6,11]. Therefore, a better and secure authentication scheme is required to validate users' authenticity.

Multifactor authentication was regarded as a formula for protecting e-wallet transactions to alleviate the problem of ownership. Authentication through multifactor solutions continues to lead the market by securing the future and reputation of e-wallets through emerging technology. They also deter hackers and crimes from attacking users. Multifactor authentication is an access control method that an entity can effectively perform through multiple stages of authentication [40,41], and it helps make it difficult for any intruder to hijack the identity of the real user [42].

Until now, many methods were available for secure user authentication, including biometric authentication [43]. Biometric means automated detection of people based on their specific behavioral and physical attributes, such as experience, speech, fingerprint, iris, etc. [44]. There are two types of biometrics, namely unimodal biometrics and multimodal biometrics [25]. Biometric systems have many uses. Several biometric-based authentication systems are shown in [9,43,45–47]. Additionally, facial biometric authentication [48] plays a crucial part in electronic payment authentication. Built-in video cameras, particularly front-facing digital cameras, are now more common in mobile phones. Facial biometric authentication has become very popular compared to the optical fingerprint recognition component in cell phones, such as in the approach of [49,50]. The works related to e-wallets and their findings are shown in Table 1.

Table 1. Work related to e-wallets.

Authors	Methods and Models	Technical Feature	Finding/Weakness	Evaluation Approach
[9]	Biometric authentication	• Fingerprint authentication • Camera	The prototype tries to ensure the captured fingerprint is consistent and the process is repeatable (usability). Facial recognition does not work well under poor conditions such as poor lighting	Likert-5, VeriFinger's software development kit (SDK)
[11]	WinAuth/Google Authenticator	• Fingerprints • One time password	Shared secrets—regular renewal to keep security due to deterministic function of storing password	Framework
[43]	Biometric authentication	• Pin • Fingerprint	This study requires additional hardware	RSA
[45]	Biometric authentication	• Facial	Does not work well under poor conditions such as poor lighting	Framework
[46]	cryptographic algorithm	• OTP • Pattern lock	Pattern lock is vulnerable to side-channel, smudge, and shoulder surfing attacks	Framework
[51]	Biometric Authentication	• Username • Password • Fingerprint	This study approach user ID, password, and fingerprint. In order to continue to grow, the security and the privacy aspects need to be improved	Android studio
[52]	Biometric authentication	• Fingerprints	This study requires an extra device to authentication fingerprint. Costly and single elements	Public key cryptosystem
[53]	Biometric authentication	• Iris	Poor practicality, unable to distinguish iris colors by ethnic features, the issue of financial difficulty is thus high market price, and therefore, the decrease in general utility is difficult to miniaturize. Requires extra hardware	Mat lab

The literature review focuses on the need for device identity-based authentication in electronic payment systems such as e-wallet applications that use over two attributes (what you know, what you have, and what you are). Very few of the online banking or mobile banking studies used IMEI to generate tokens for one time password (OTP) authentication. This technique, however, was not related to e-wallets. In addition, the literature review indicates that there is a possibility of implementing device identity-based user authentication for e-wallets. From the literature study, it is clear that passwords and OTP are not sufficient for secure transactions. Multifactor authentication research has been carried out using many attributes. However, the study does not provide a mixture of attributes that combine specific authentication mechanisms and unique identification of the mobile device to improve the security of e-wallet apps. Many researchers used passwords, OTP, and biometric elements for e-wallet authentication. Most of the work is based on a framework that does not have enough justification for user authentication. Therefore, this research proposes a device identity-based user authentication for e-wallet apps.

3. The Authentication Conceptual Design

To satisfy the user, every application must apply design. A design is a technique for organizing elements in the most effective way to achieve a particular goal. Registration and authentication phases are the two steps in the authentication design. Before using the Zamwallet, the user must enter their information through a process called registration. A technique known as the authentication step is used to verify the information. In this registration phase, the user has to input their credentials, such as password, fingerprint, and OTP, into the mobile device, from which the information will be moved into the database. The user information will be stored in the Firebase server through real-time databases. The user is required to enter their registered credential information, such as password, fingerprint, and OTP, into the mobile device throughout this authentication phase, where the server will compare registration and authentication value. During transactions, where the Paypal gateway is connected to the Firebase interface, the same rule will be applied. We will utilize Android Studio in our proposed method. The server and application are linked to Android Studio using a variety of (insert full form of API here) APIs, making it simple to construct an e-wallet using sophisticated features and libraries available on the Android platform. For the back-end, we will utilize a real-time Firebase database and PayPal gateway. Once the proposed apps have been completed, the next process is the evaluation process. The proposed method has been used to test the performance of the Zamwallet in the requirement test. The key aim of the requirement testing is to validate how well the functionality works. The proposed conceptual design of authentication is shown in Figure 1.

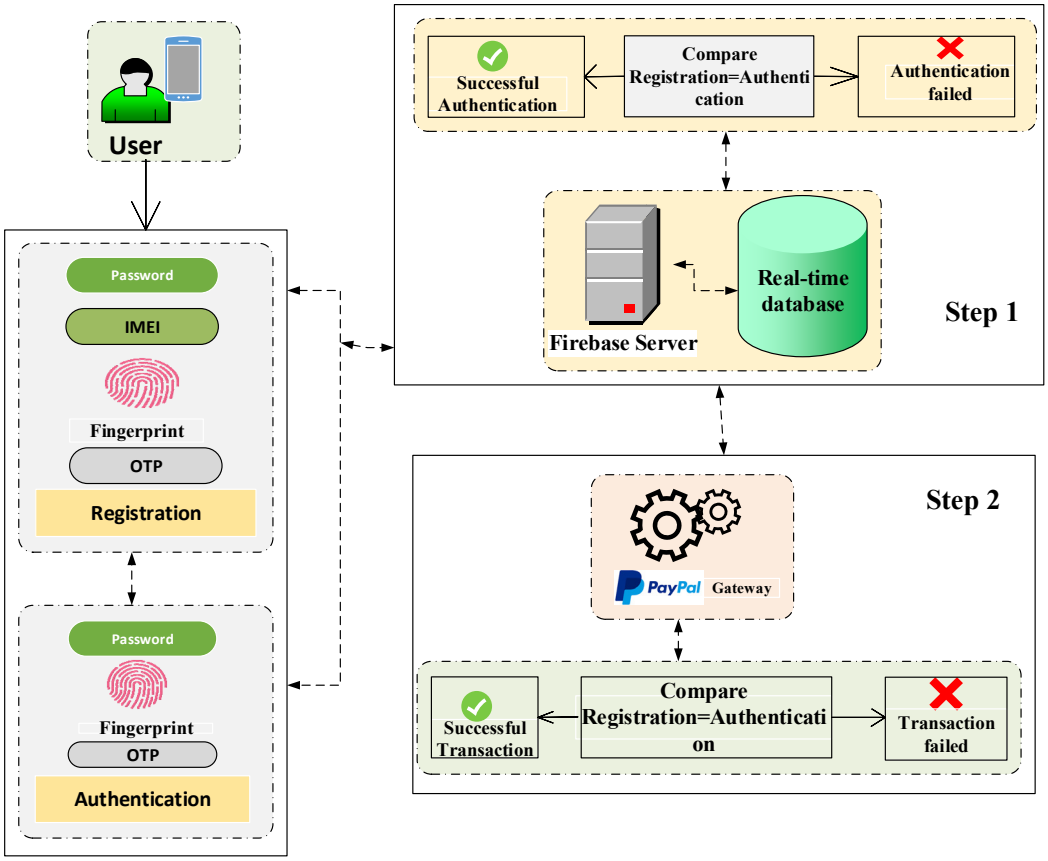


Figure 1. The framework for the proposed technique.

3.1. Authentication Process Flow

The authentication process flow is one of the important elements of the proposed Zamwallet. The primary concern here is to focus on user authentication. The authentication process flow has two phases—registration and authentication phases. The authentication process flow of the study is given in this section. Table 2 presents the notations used in the proposed method.

Table 2. List of the symbols used.

Notations	Description
Ui	User
IDi	Unique Identifier of User
PWDi	Password of the User
BFi	Biometric feature of User
IMEI	International Mobile Station Equipment Identity of user
d	Private key in *RSA*
e	Public key in *RSA*
n	Computed as product of chosen prime numbers (*p* and *q*)
RC	Random Challenge *(in this context—One Time Password)*
CS	Concatenated String *(Ui)*
ESQ	Encrypted String
PK*P*	Plain Private Key
PK*E*	Encrypted Private Key
E (Data, Encryption Key)	Encryption Function
R*F*	Result of Comparison Function (*PWDi, IMEI, BFi)*
D (Data, Encryption Key)	Decryption Function

3.1.1. Registration Phase

To use the service, the user will need to perform a one-time registration. In the registration phase, the users' information is gathered. Using the information given and the method used during the login, the server checks if the user is legitimate. The proposed registration procedure is presented in Figure 2. Here, the user has to register their account after giving the relevant details. The registration procedures are as follows:

Step 1: Start Initialing Registration process

Step 2: User inputs their information, (full form) ***PWDi, IMEI***, (full form)***BFi*** on the mobile device

Step 3: Generates two large prime numbers p and q, where it computes $N = p \times q$

Step 4: Chooses integers e and d, **which satisfy E** $\times$ **d,** mod $((p-1) \times (q-1)) = 1$

Step 5: Generates OTP and send to input numbers

Step 6: Input OTP,

Step 7: If

"OTP Match"?

Goto Step 8

Else

Goto Step 6

Step 8: Random Challenge ***RC*** for the ***Ui***, where it generates ***a*** **IDi** = ***PWDi, IMEI, BFi***

Step 9: Store all the ***Ui*** information

Step 10: Stop

In the registration phase, the information that the user would like to divulge to log into the system will be gathered. In this phase, the user enters their personal identity and can access the system only then.

3.1.2. Authentication Phase

When the customer tries to log in, the authentication server has to authenticate the user. If both values are the same, the authentication is successful. The authentication process flow is composed of two processes—the login process and authentication process. The user must log in using the approved password, fingerprint, and OTP for authentication. After logging in utilizing this method, the user can see only the account details. To complete a transaction, the user must authenticate themself via fingerprint. The transaction will be completed only after the user has authenticated with the fingerprint details. The authentication process flow is shown in Figure 3. The authentication measures can be explained as follows:

Step 1: Start initialing authentication process

Step 2: User inputs their information, ***PWDi, IMEI, BFi*** on the mobile device

Step 3: Check for verification, where user encryption function to concatenated string ***ESQ = E (CS)***

Step 4: Use encryption function to plain private key, concatenated string ***PKE = E (PKP, CS)***

Step 5: ***RF***, matched with decryption function ***IF RF = D (PKE, CS)***

Step 6: Generates OTP and sends to input numbers

Step 7: Input OTP,

Step 8: If

"OTP Match"?

Goto Step 9

Else

Goto Step 7

Step 9: Access granted

Stop

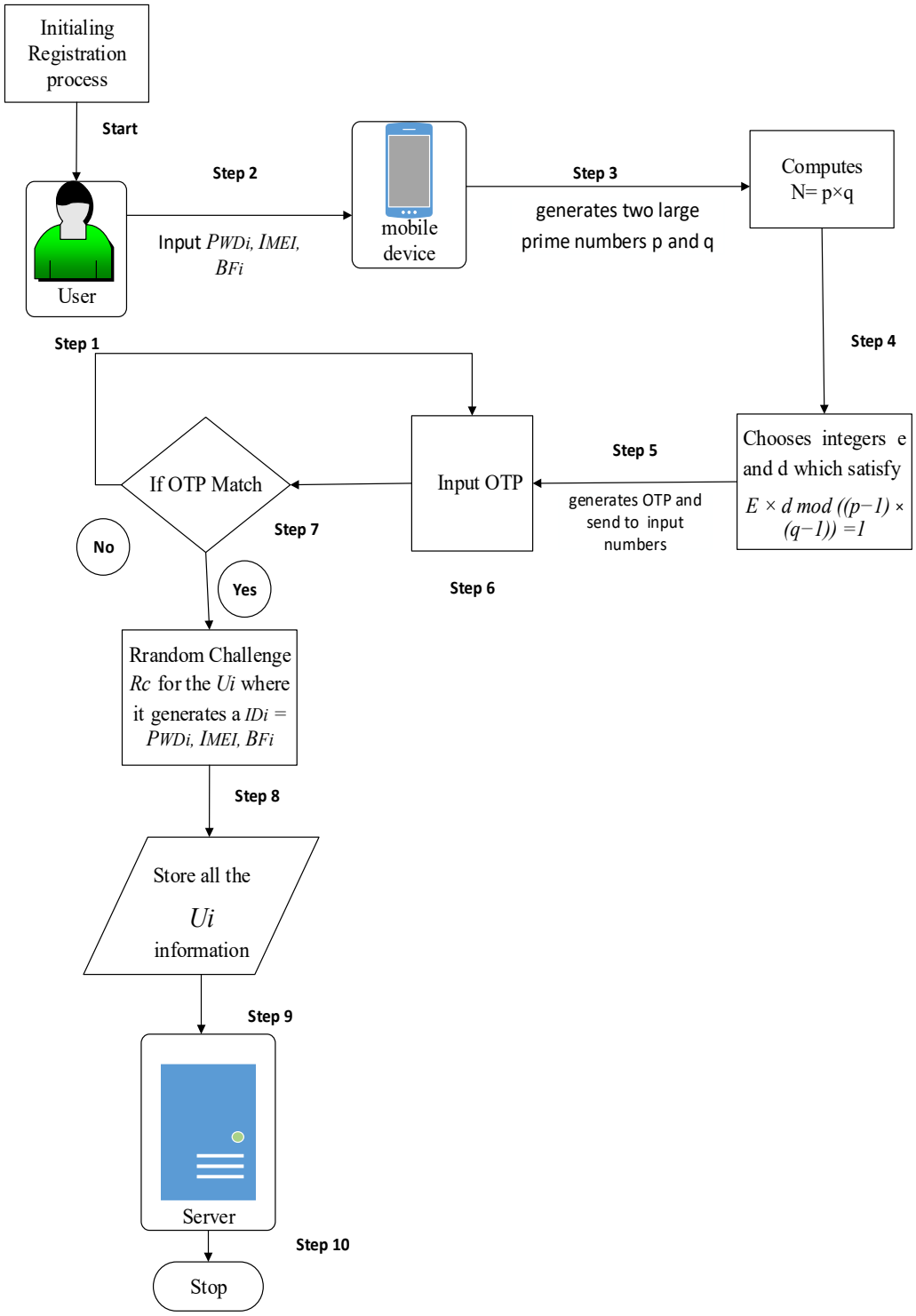


Figure 2. Registration process flow.

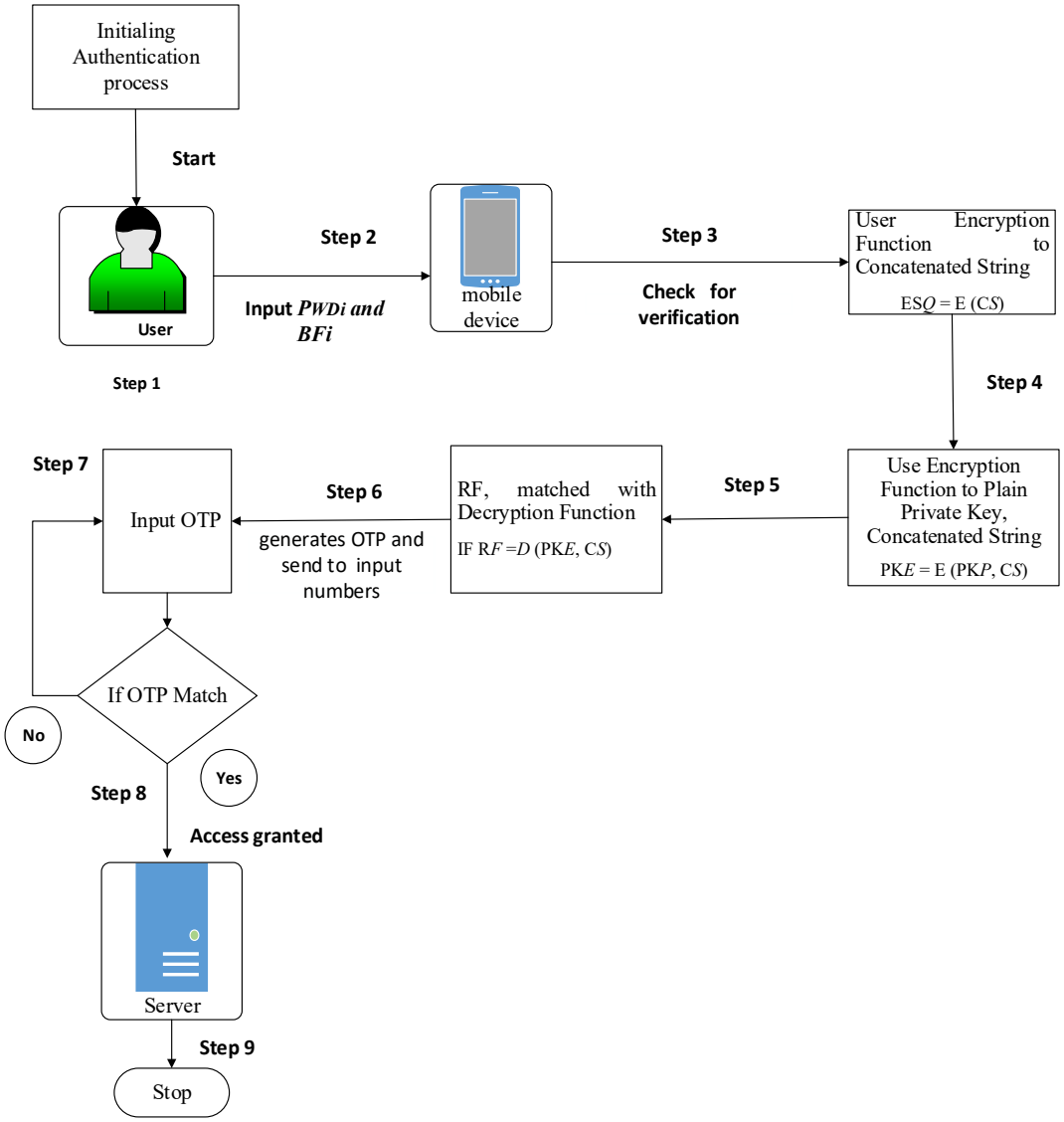


Figure 3. Authentication process flow.

3.2. Detailed Design

The suggested authentication design does not consist of registration and authentication phases alone. It also includes diagrams of components, class, and sequence.

3.2.1. Modules on Zamwallet

Two fundamental modules are available in the proposed Zamwallet application—one is the admin module, and the other is the user module. The admin module is used to enter login details, add/update/delete items, add-on/update/delete details for registered users, and update order status. It is a system management module that provides an interface for dealing with admin and user management. This systems management monitoring aids the admin in connecting things to the database, updating and removing user information, reviewing user orders, collecting transaction summaries, and collecting all instructions received by the registered user. Management admins are frequently in charge of the system. It is also known as the back-end module. The real-time Firebase database, which uses automated user updates, is employed in our proposed method. Firebase also offers machine-learning techniques for application data security via the ML Kit mobile SDK that offers a selection of highly competent and efficient pre-trained models based on deep

learning algorithms [54]. Various kinds of attacks can be reduced using machine-learning techniques, such as malware detection and cloud security [54,55].

The front-end Graphical User Interface (GUI) is the user module. The user interface enables users, by registering in the application, to log in, edit profiles, view transactions, top-up money, transfer money, and log out.

3.2.2. Component Diagram

The components diagram explains how this system is divided into many subs-systems. There is a multilayer component in Zamwallet such as the Firebase database, Android device, Firebase SDK, PayPal SDK, and framework controller. The Firebase real-time databases are connected to the Firebase management. The application controller includes all the applications that are the middle layer within the Firebase SDK, PayPal SDK, and framework components. The controller (mobile device) is connected to both sides (i.e., the Firebase real-time database and framework). The Firebase is responsible for connecting to the real-time database server. Figure 4 illustrates the component diagram of this application.

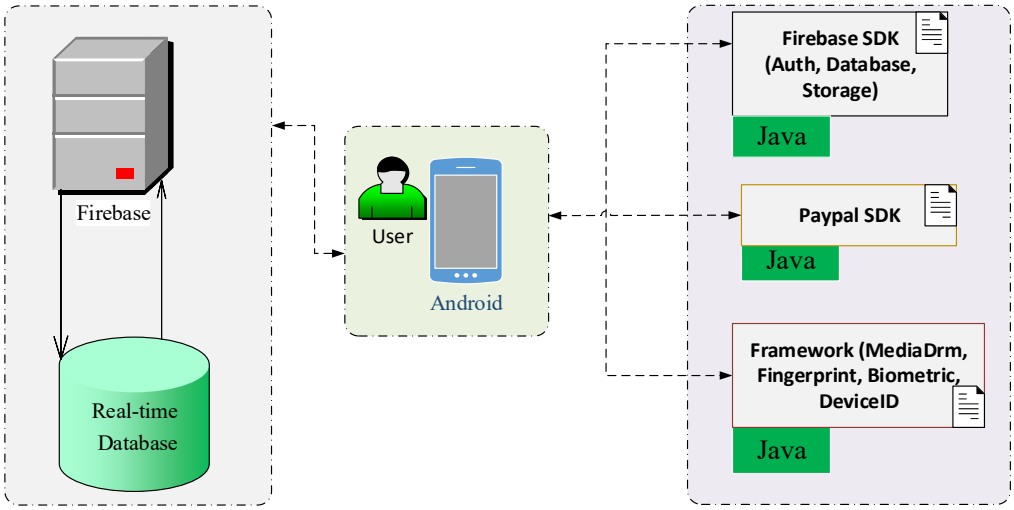


Figure 4. Component diagram of the proposed method.

3.2.3. Class Diagram

The implementation requirements depend on the analysis of the system. The unified modeling language (UML) is used to draw and visualize the architectural structure of the projects. The UML is a unique, standardized representation. In a system review, a developer may use the case diagram to display all the participants taking part in the project. Figure 5 presents the class diagram of the Zamwallet.

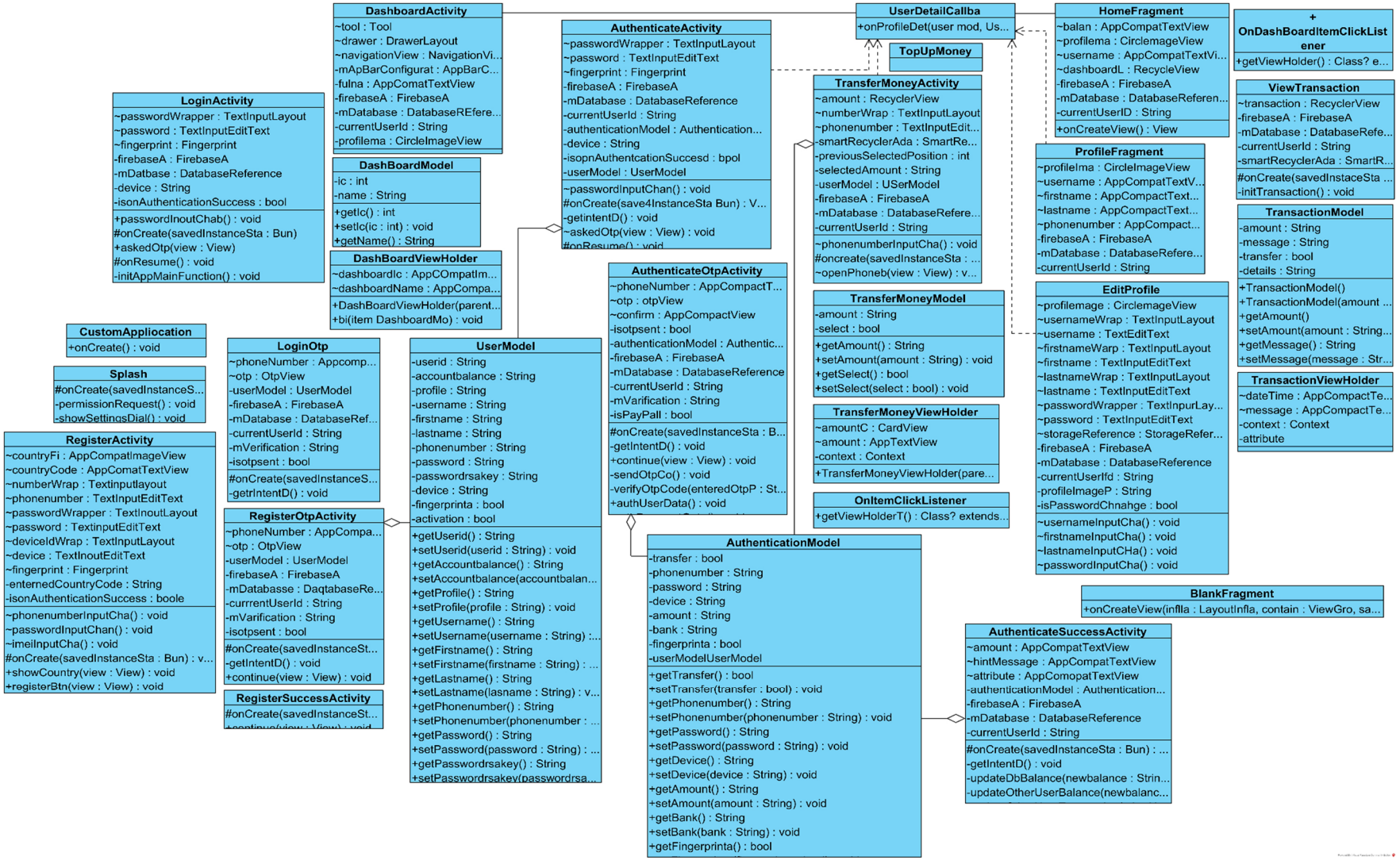


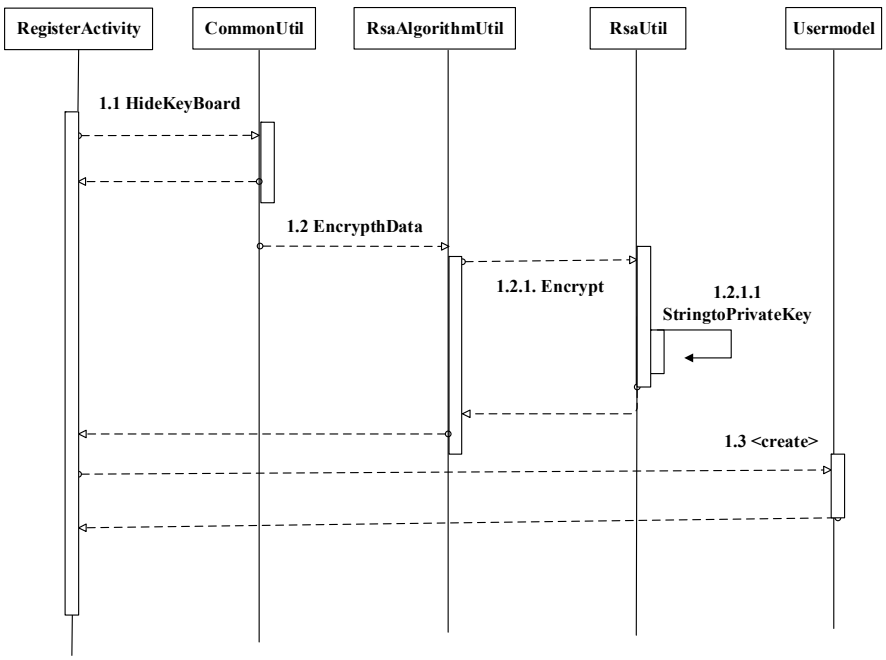
Figure 5. Class diagram of the entire proposed method.

3.2.4. Sequence Diagram

A sequence diagram is an interaction diagram that describes how the exchange of messages is carried out over time. A sequence diagram is a good way to display different runtime scenarios and to validate them. This can help predict how a system is going to behave and recognize roles that a class may have in designing a new system. This sub-section describes the sequence diagram of proposed Zamwallet apps for the front-end interface where a customer will register, authenticate, top-up money, and transfer money through a payment gateway.

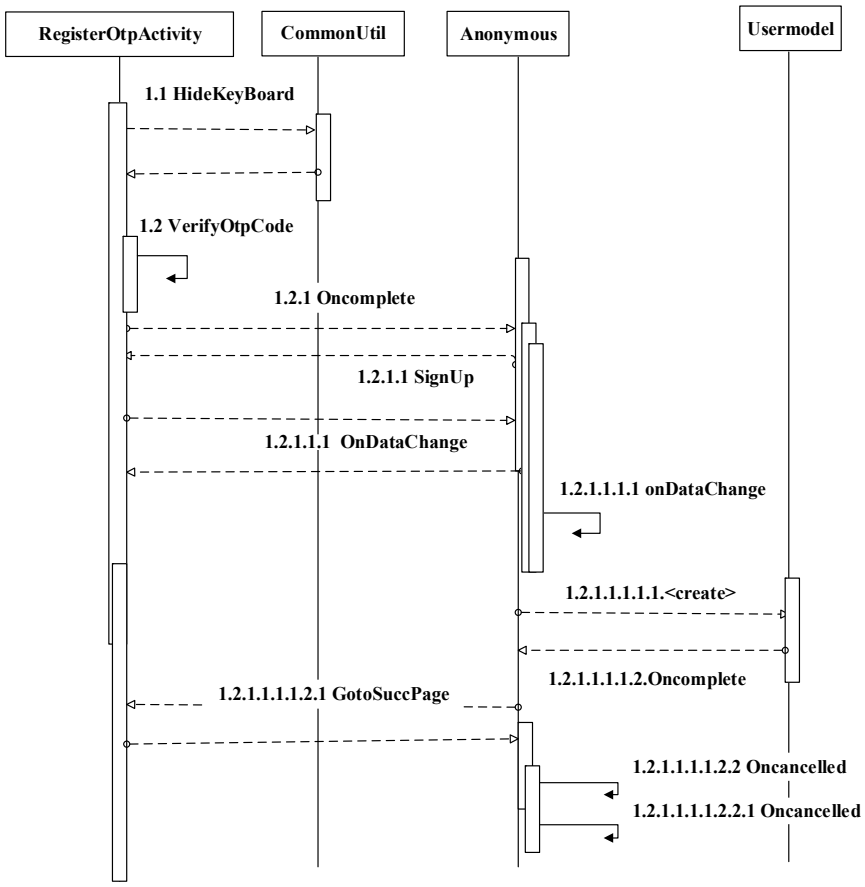
1. User Registration Function

The proposed application has two process register activities and a register OTP activity. Creating an account is the first step in using this application. The method of registration is quite simple. Initially, the user must complete all the fields needed in the registration form. The registration form consists of four fundamental categories of authentication: password, device ID, fingerprints, and OTP. Then, users will submit the form by clicking the register button. The backend controller of the registration page will ensure that all fields needed are correct. If not, the program shows the user an error message indicating the precise details of the error, and the user is re-directed to the registration page. In the proposed Zamwallet, the user passwords are encrypted using the full form of the RSA (RSA) algorithm for security. In the following figure, the hide keyboard function is used to keep the virtual keyboard to prevent autocorrect for spelling mistakes; here, the hide keyboard function is used for security. Encrypted data is the process of encrypting user input data using an RSA string to the public key and finally creating a user account in register activity. After input, an OTP verification code will be sent to the registered phone number of the user. If all the information is correct, the user is automatically directed to the default page to log in as a member. The registration is shown in Figure 6a, while Figure 6b shows the OTP activity.



(**a**)

Figure 6. *Cont.*



(**b**)

Figure 6. (**a**): Register activity of proposed method and (**b**): Register OTP activity of proposed method. In (**a**), Step 1.1: The *RegisterActivity* will hide the keyboard in the *CommonUtil* (where util functions are stored) and return a message to the *RegisterActivity.* Step 1.2: The *EncrptData* function is the process in *RsaAlgorithmUtil*, which waits for the user input data to process the message. Step 1.2.1: The user input data encrypts into *RsaUtil.* Step 1.2.1.1: Then, user input data encrypt using RSA string to public key in *RSAUtil* and send return message from the *RsaAlgorithmUtil* to *RegisterActivity.* Step 1.3: Finally, the register activity sends a message to the UserModel to create an account and wait for the RegisterActivity to respond. In (**b**), Step 1.1: The *RegisterOtpActivity* will hide the keyboard in the *CommonUtil* and return a message to the *RegisterOtpActivity*. Step 1.2: The *verifyOtp* code starts, ends with *RegisterOtpActivity,* and waits for the response. Step 1.2.1: Once OTP is verified, the *onComplete* function then waits for the anonymous (where Utils functions are stored) response to process the request. Step 1.2.1.1: Anonymous responds to the *RegisterOtpActivity* request and confirms that sing up has been successful. Step 1.2.1.1.1: Then, *RegisterOtpActivity* sends request *ondataChange* (real-time database) to anonymous. Step 1.2.1.1.1.1: The *onDataChange* function starts and ends with the same lifeline (i.e., anonymous). Step 1.2.1.1.1.1.1: Then, anonymous sends a request to the *UserModel* to create a user profile in the real-time database. Step 1.2.1.1.1.1.2: *UserModel* sends response to the anonymous *onComplete* response. Step 1.2.1.1.1.1.2.1: Then, anonymous sends a response to the *RegisterOtpActivity* for successful registration.

2. User Login Function

To log into the system, the client needs a password, fingerprints, and OTP verification. The user password is encrypted with an RSA algorithm with a user private key in the RSA unit. When a user submits their account credentials and presses the ask OTP button, the login method will send a verification code to the registered phone number if the provided data match the user's login details stored in the database. If it does not match, an error message will be displayed to the user for the corresponding field. If the data submitted

match the credentials in the database, the customer is forwarded with a login profile to the main page. The login function can be divided into two categories—login activity and login OTP activity. Figure 7a describes the sequence diagram of user login activity, and Figure 7b describes the login OTP activity.

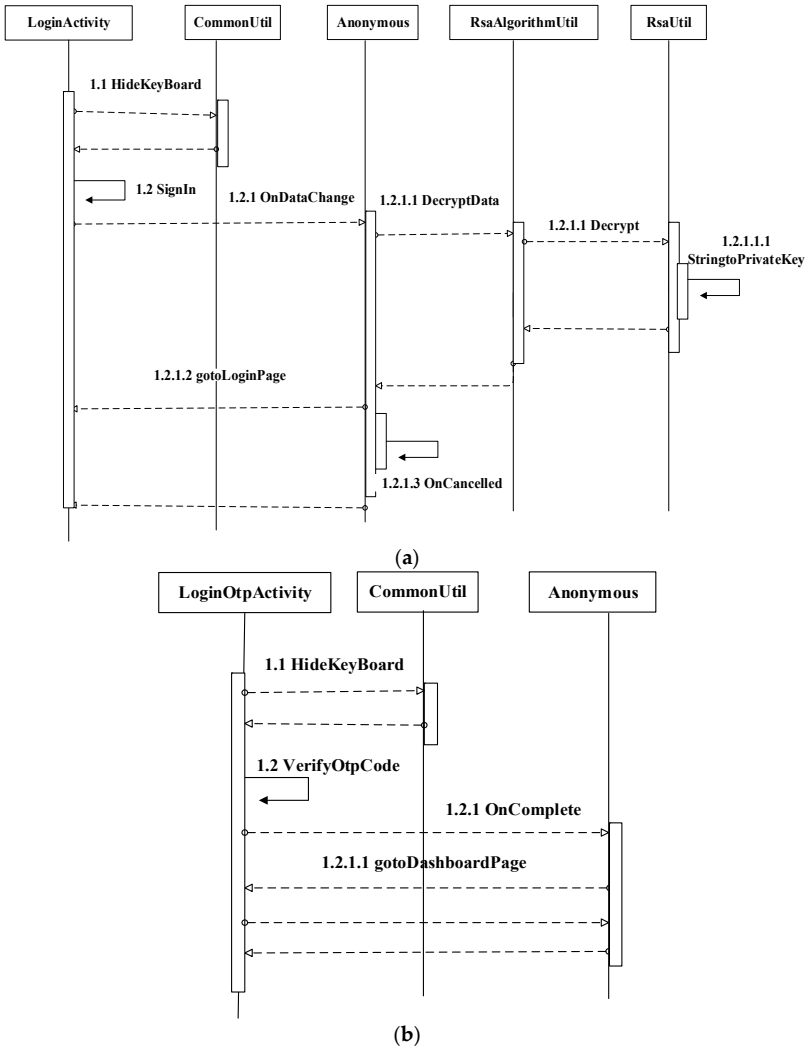


Figure 7. (**a**): Login activity of proposed method and (**b**): Login OTP activity of proposed method. In (**a**), Step 1.1: The *LoginActivity* will hide the keyboard in the *CommonUtil* and return to the *LoginActivity*. Step 1.2: The signing starts and ends with *LoginActivity* and waits for the response. Step 1.2.1: Then, *loginactivity* sends a request *onDataChange* to anonymous. Step 1.2.1.1: Then, anonymous sent a request to *decrytData* into the *RsaAlgorithmUtil*. Step 1.2.1.1.1: User input data decrypt and sent to the *RsaUtil*. Step 1.2.1.1.1.1: Then, user input data decrypt using RSA string to private key in *RSAUtil* and send a return message to the *RsaAlgorithmUtil* to anonymous. Step 1.2.1.2: Then, anonymous sends a response to the *LoginActivity* and goes to the login page. Step 1.2.1.3: Finally, *LoginActivity onCancelled*. In (**b**), Step 1.1: The *LoginOtpActivity* will hide the keyboard in the *CommonUtil* and return to the *LoginOtpActivity*. Step 1.2: The *verifyOtp* code starts and ends with *LoginOtpActivity* and waits for the response. Step 1.2.1: Once OTP is verified, then the *onComplete* function waits for the anonymous response to process the request. Step 1.2.1.1: anonymous responds to the *LoginOtpActivity* and goes to the *DashboardPage*.

3. User Authenticating Function

Authentication functionality is one of the most important functions in Zamwallet. For top-up and transfer of money, functionality authentication is a single method. Users

require a password, fingerprints, and OTP authentication for access to the system. With a private user key, the user password is encrypted into the RSA unit using the RSA algorithm. Authentication of the OTP identification code with user credentials is required when logging into the system. When a user confirms their account credentials and clicks the OTP button, the authentication system will deliver the registered phone number to the success list, to the verification code, where the submitted data should match the user login information. If this does not match, an error message is shown to the client, and it must be validated by the customer with OTP authentication. The user shall be forwarded to the profile if the requested data matches the credentials in the database. The user will see the updating transaction in this process. The user transaction updates the user balance and servers instantaneously. The authenticating function can be divided into three categories—*AuthenticateActivity*, *AuthenticateOtpActivity,* and *AuthenticateSuccessActivity*. Figure 8a describes the sequence diagram of *AuthenticateActivity*, Figure 8b describes the *AuthenticateOtpActivity*, and finally, Figure 8c describes the *AuthenticateSuccessActivity*.

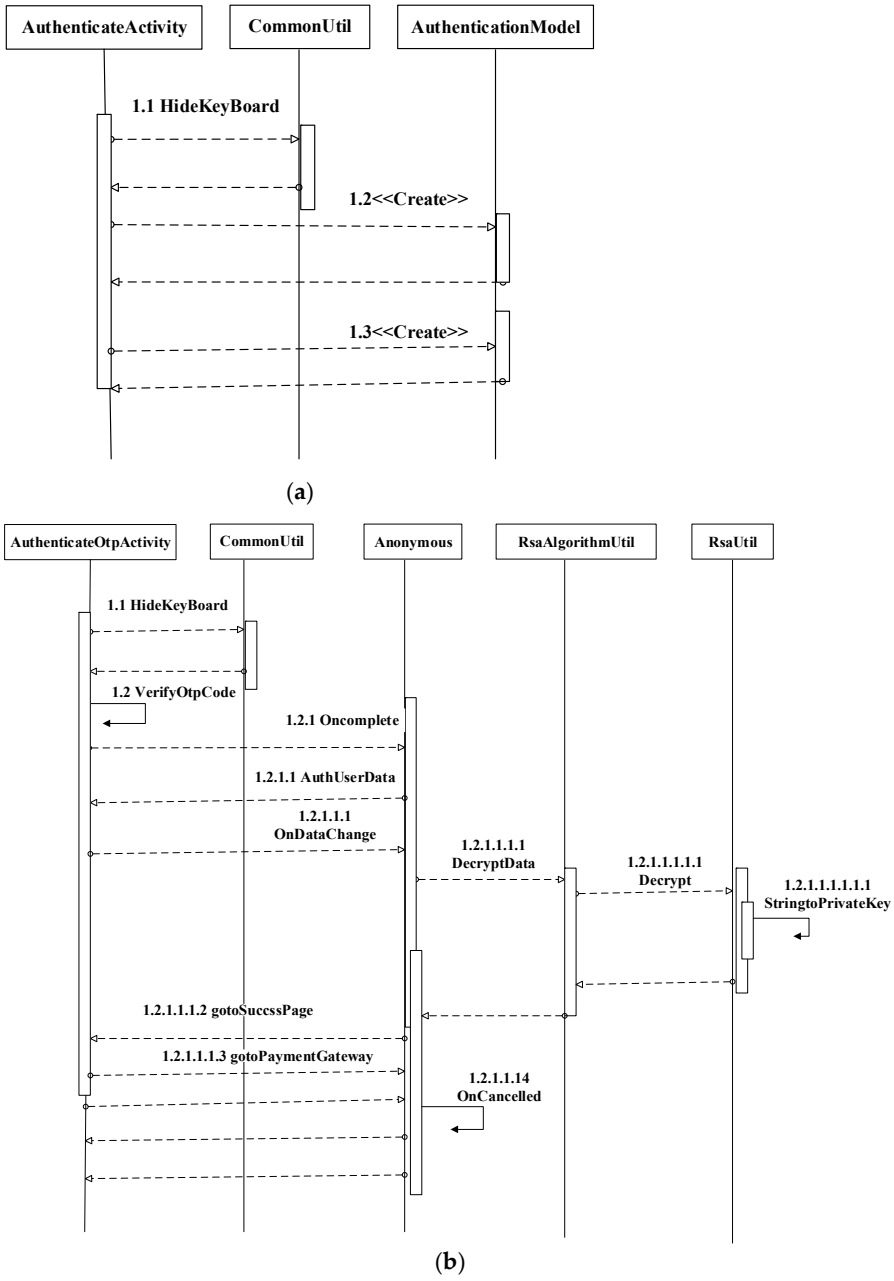
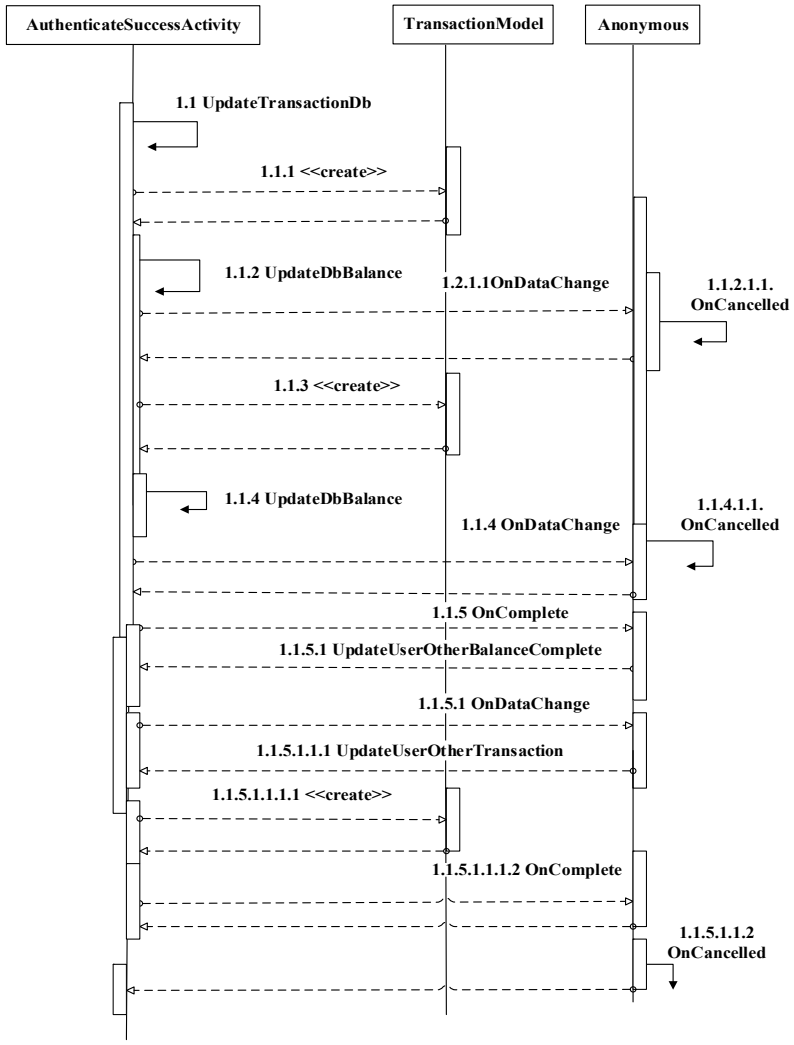


Figure 8. *Cont.*



(**c**)

Figure 8. (**a**): Authentication activity of proposed method, (**b**): Authentication OTP activity of proposed method, and (**c**): Authentication successful activity of proposed method. In (**a**), Step 1.1: The *AuthenticateActivity* will hide the keyboard in the *CommonUtil* and return to the *AuthenticateActivity*. Step 1.2: *AuthenticateActivity* sends a message to the *AuthenticationModel* to create an account and waits for *AuthenticateActivity* to respond. Step 1.3: Finally, *AuthenticationModel* creates and sends a response to *AuthenticateActivity.* In (**b**), Step 1.1: The *AuthenticateOtpActivity* will hide the keyboard in the *CommonUtil* and return a message to the *AuthenticateOtpActivity*. Step 1.2: The *verifyOtp* code starts, ends with *AuthenticateOtpActivity*, and waits for the response. Step 1.2.1: Once OTP is verified, the *onComplete* function waits for anonymous response to process the request. Step 1.2.1.1: Anonymous sends a response to *AuthUserData* and sends it back to the *AuthenticateOtpActivity*. Step 1.2.1.1.1: Then, *AuthenticateOtpActivity* sends a request *ondataChange* to anonymous. Step 1.2.1.1.1.1: Then, anonymous sends a request to *decrytData* into the *RsaAlgorithmUtil*. Step 1.2.1.1.1.1.1: User input data decrypted and sent to the *RsaUtil.* Step 1.2.1.1.1.1.1.1: Then, user input data decrypted using RSA string to private key in *RSAUtil* and sends a return message to the *RsaAlgorithmUtil* to anonymous. Step 1.2.1.1.1.2: Then, anonymous sends a response to the *AuthenticateOtpActivity* and goes to the *SuccessPage*. Step 1.2.1.1.1.3: Once *AuthenticateOtpActivity* succeeds, the request sent to the *PaymentGateway.* Step 1.2.1.1.1.4: Finally, *AuthenticateOtpActivity onCancelled.* In (**c**), Step 1.1: The *UpdateTransactionDb* will start and return a message to the *AuthenticateSuccessActivity*. Step 1.1.1: The *AuthenticateSuccessActivity* creates an amount in the *TransactionModel* and sends it back to the *AuthenticateSuccessActivity*. Step 1.1.2: The *updateDbBalance* starts and ends with *AuthenticateSuccessActivity* and waits for the response. Step 1.1.2.1: Then, *AuthenticateSuccessActivity*

sends a request *ondataChange* to anonymous. Step 1.1.2.1.1: Then, *onCancelled* starts and ends with anonymous and sends a request to the *AuthenticateSuccessActivity*. Step 1.1.3: The *AuthenticateSuccessActivity* creates an amount in the *TransactionModel* and sends it back to the *AuthenticateSuccessActivity*. Step 1.1.4: The *updateDbBalance* starts and ends with, *AuthenticateSuccessActivity* and waits for the response. Step 1.1.4.1: Then, *AuthenticateSuccessActivity* sends a request *ondataChange* to anonymous. Step 1.1.4.1.1: Then, *onCancelled* starts and ends with anonymous and sends a request to the *AuthenticateSuccessActivity*. Step 1.1.5: Then, *onComplete* request sent to anonymous. Step 1.1.5.1: Then, anonymous sends *UpdateOtherUserBalance* to the *AuthenticateSuccessActivity*. The next steps process the same flow.

4. Top-Up Money Function

Users can access multiple banks for topping up money after authentication into the system. They need to choose the amount they want to update in this application. Once the user selects their preferred bank account, the payment gateway is sent directly to the user. PayPal is the default payment gateway for transactions. Each registered user can add funds to their PayPal account. The PayPal sandbox account helps users to work conveniently. The user's money is immediately updated in the user balance and database. Figure 9 describes the sequence diagram of user top-up money for the proposed Zamwallet apps.

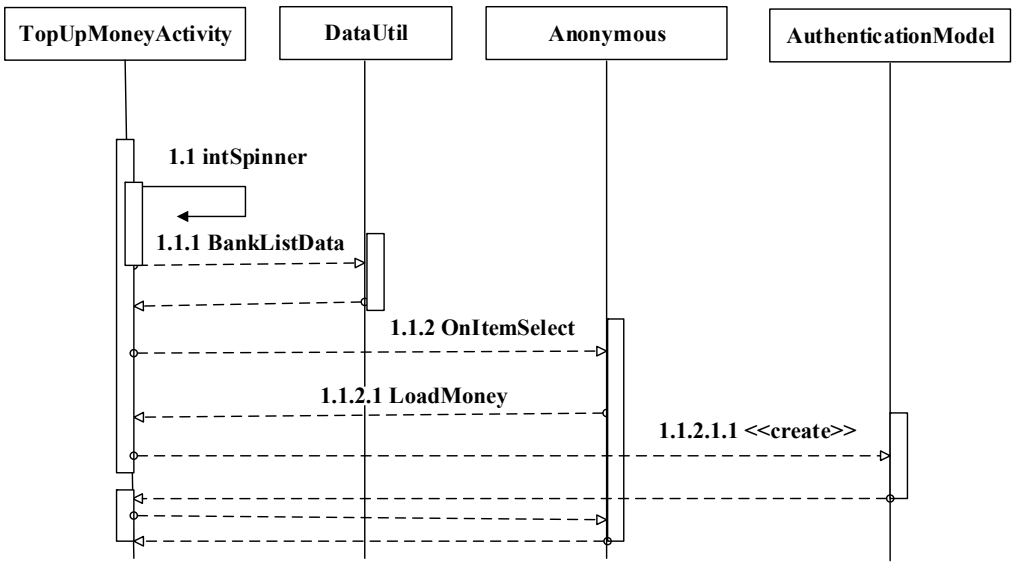


Figure 9. Top-up money activity of proposed method. Step 1.1: The *InitSpinner* will start and return a message to the *TopUpMoneyActivity.* Step 1.1.1: The *TopUpMoneyActivity* sends request *BankListData* to the *DataUtil* and waits for the response. Step 1.1.2: The *TopUpMoneyActivity* sends request *OnItemSelect* to anonymous and waits for the response. Step 1.1.2.1: Anonymous sends the response *loadMoney* to the *TopUpMoneyActivity* and waits for the request. Step 1.1.2.1.1: The *TopUpMoneyActivity* creates an amount in the *AuthenticationModel* and sends it back to the *TopUpMoneyActivity*.

5. Transfer Money Function

Users can transfer money from their account to another registered Zamwallet account after the money has been added to the system. In this process, the user has to choose the amount they want to transfer before transferring the money to another Zamwallet account. Users could also use the contact list to get the recipient's profile information. Once the user has chosen the recipient, they have to authenticate themself using their password, fingerprint, and OTP. The money is sent to the recipient's account after successful authentication. Figure 10 describes the sequence diagram for user money transfer for proposed Zamwallet apps.

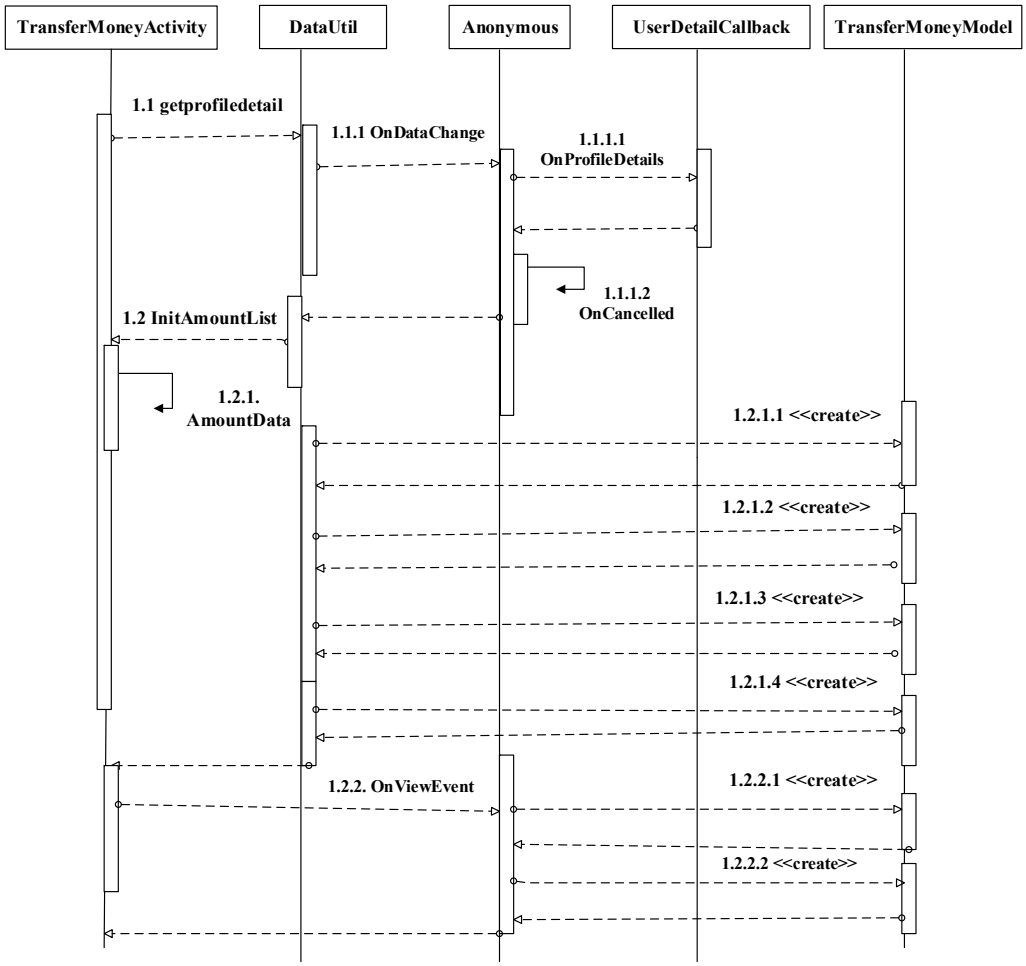


Figure 10. Transfer money activity of proposed method. Step 1.1: The *TransferMoneyActivity* sends request *getProfileDetails* to the *DataUtil* and waits for the response. Step 1.1.1: The *DataUtil* sends a request *onDataChange* to anonymous and waits for the response. Step 1.1.1.1: Anonymous sends a request *onDataProfileDetails* to the *UserDetailscallback* and sends the response to anonymous. Step 1.2: The *InitAmmountList* will start and return a message to the *TransferMoneyActivity.* Step 1.2.1: The *TransferMoneyActivity* sends request *AmountData* to the *DataUnit* and waits for the response. Step 1.2.1.1: The *DataUtil* creates an amount in the *TransferMoneyModel* and sends it back to the DataUtil. Step 1.2.2: The *TransferMoneyActivity* sends a request *onViewEvent* to anonymous and waits for the response. Step 1.2.2.1: The anonymous request created in the *TransferMoneyModel* is sent back to anonymous and waits for the response.

4. Proposed Method Implementation

The proposed Zamwallet has some functional capabilities that need to be implemented. For each functionality, a full procedure is followed. All the functionality has been developed and implemented based on an Android Studio framework and a Firebase in real-time database management. The prototype is evaluated based on the registration stage and authentication stages. The simulation is run on the webserver side on a DELL laptop computer with Intel Core i7 CPU, 3.40 GHz CPU, as well as 6 GB RAM. The operating system is Windows 10 Professional. Android is an open-source operating system built on Linux, and the Android platform makes everyday activities simple and fast, and it has helpful apps for mobile devices. The prototype implementation is discussed as follows:

1. User Registration

To perform a transaction from Zamwallet to another Zamwallet, the user must be a registered member. When a new user signs up on the Zamwallet by clicking the register

button, the switch on the welcome page. If a user clicks on "Register," the user will receive a registration form. The user must complete all the necessary fields in the following four specific categories: knowledge (i.e., password), device ID (i.e., IMEI), biometrics (i.e., fingerprints), and ownership (i.e., OTP). A registered member can access this method only when the registration is complete. A *"RegisterSuccessActivity"* feature can process and verify all the data entered in the registry form before saving it in the database. After the user details have been completely input into the system, OTP verification is asked for the registration. If the user registers successfully, then a notification message, "successfully registered" appears. The proposed Zamwallet has three activities in the registration—"*RegisterActivity*", "*RegisterOtpActivity*", and "*RegisterSuccessActivity*". A display message will appear to the customer after a successful registration. The details of each step are shown in Figure 11.

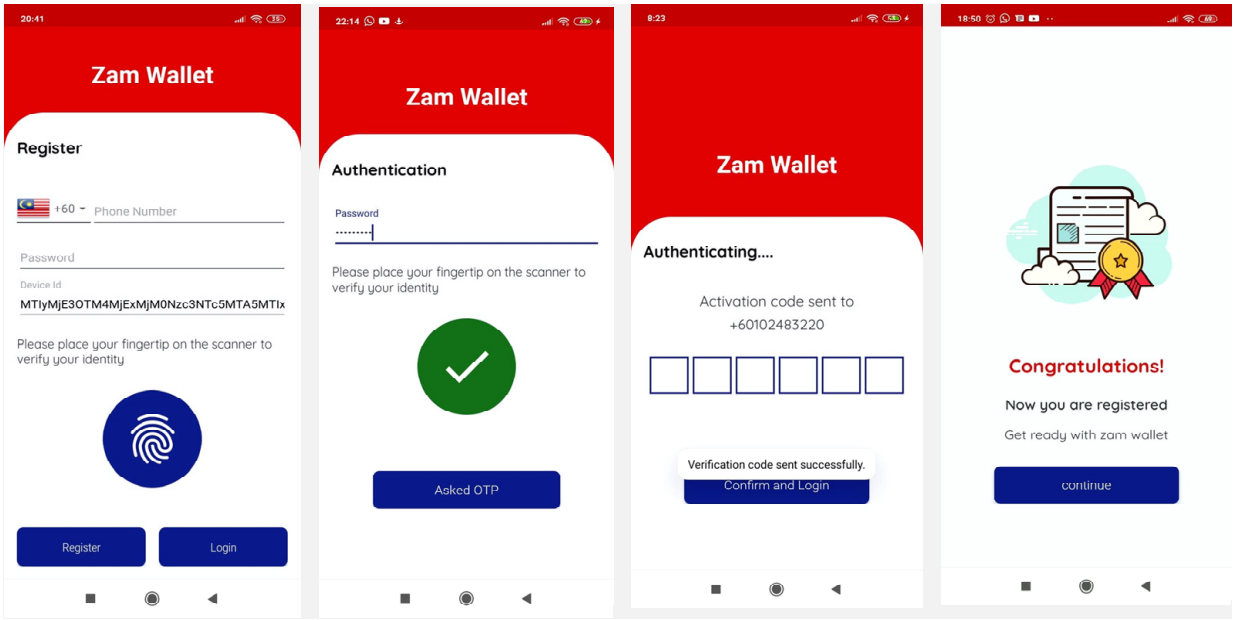


Figure 11. Registration activity proposed method.

2. User Login

The user has to use their credentials for logging into the application each period of usage. The system asks the user to first authenticate themself with their information. To log in, the user has to provide the registered details, such as password, fingerprint, and OTP. If the user enters details not registered, then the user will get a notification message. The user cannot log in to the system through system authentication properties. A request is executed, and a login form is delivered to the user when the user clicks on the login table. A user *"LoginActivity"* feature performs all the logic and procedures required to connect to the database and stay connected to it. The proposed Zamwallet has two activities in the login phase—*"LoginActivity"* and *"LoginOtpActivity"*. There is a string query to extract the correct login credentials from the database to match the requested data of the user. The user is routed to the application's homepage after active login. The user will view their name and a log out option. A login screen displaying the authentication pages is listed in Figure 12.

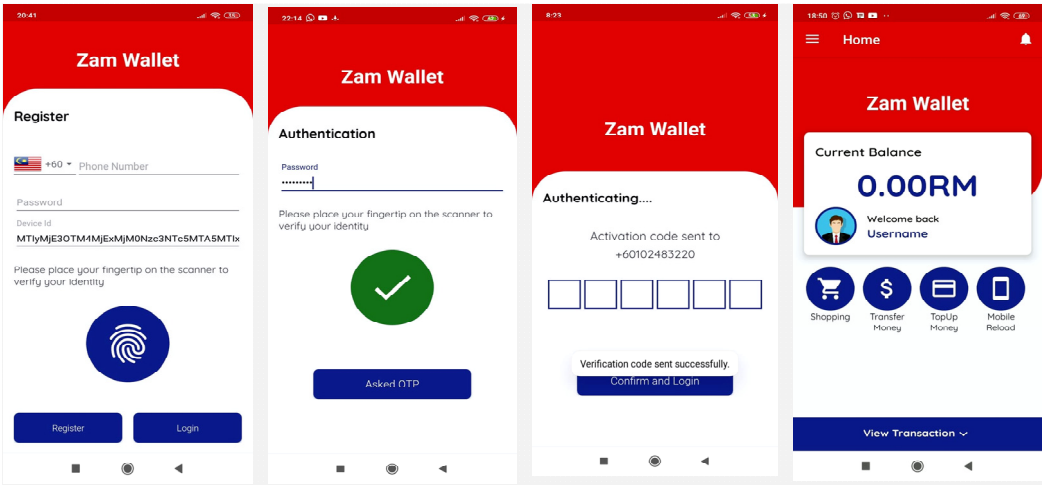


Figure 12. Login process of Zamwallet.

3. Profile Activity and Dashboard

A user profile includes the following functions: update and access profile as a system member, name of the user, user phone number, and user profile image. A feature called *"ProfileActivity"* protects both logic and procedures for connecting to the server. Meanwhile, the dashboard shows the user's profile image, wallet balance, mobile number, and transaction view. All usable links can be found on the dashboard. A function named *"DashboardActivity"* will handle all the logic and processes to connect to the application. The activity of this proposed system is shown in Figure 13.

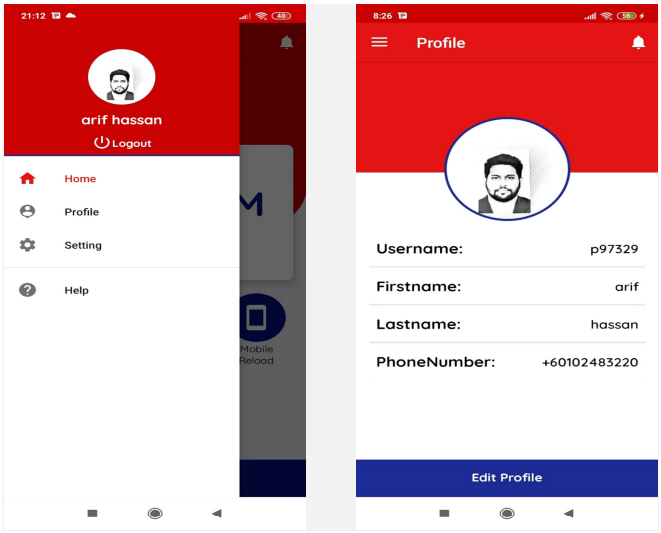


Figure 13. Profile page and dashboard activity of Zamwallet.

4. Top-Up Money

After being authenticated by the system, users can access multiple banks to top-up money. Within this option, users need to choose the level of updating they require. The users can choose their favorite bank account. Whenever the users select their preferred bank account, the password, fingerprint, and OTP verification have to be authenticated by the users. The payment gateway is submitted directly to the recipient of the application. *"TopupMoneyActivity"* is the transaction process developed for this. The execution of the money transfer happens after successful authentication. The details of each step are shown in Figure 14.

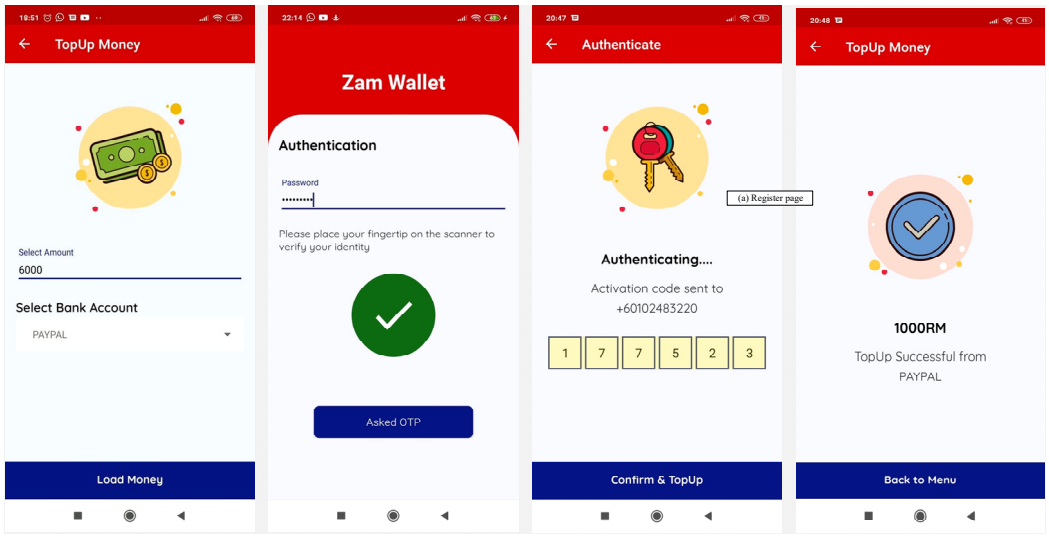


Figure 14. Top-up money activity in Zamwallet.

5. Transfer Money

Users can transfer money from their account to another registered Zamwallet account after the money has been added to the system. In this process, the users have to choose the amount to be sent before transferring the money to another Zamwallet account. The users have to authenticate themselves using their password, fingerprint, and OTP. A function named *"TransferMoneyActivity"* handles all the logic and processes involved. After successful authentication, the money transfer will be successfully executed. The entire money transfer activity testing is shown in Figure 15.

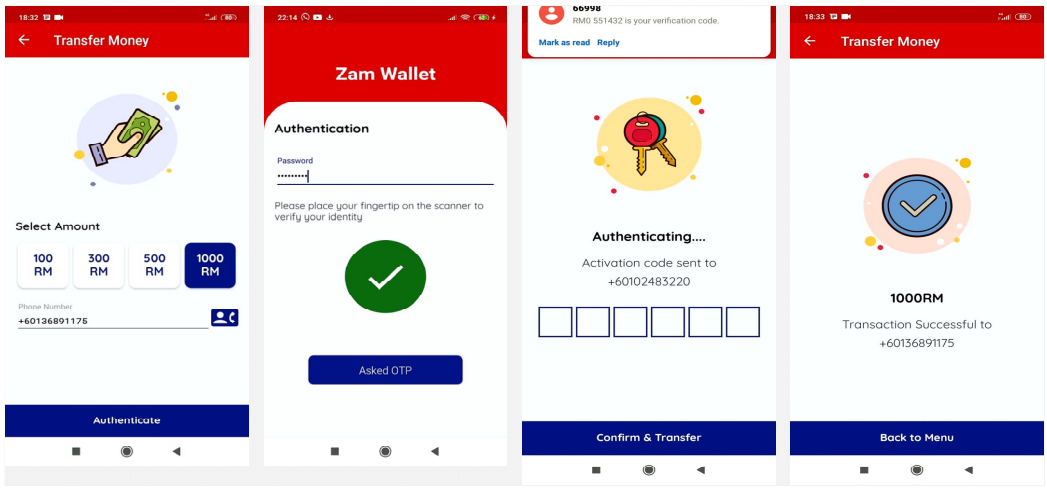


Figure 15. Money transfer activity in Zamwallet.

6. View Transaction

Any registered member can view all the purchase details in this module. After successful login, the user can check transaction history. A function named *"ViewTransactionActivity"* has been developed to monitor transactions by clicking the view transaction table. All the transactions of the users, such as transfer money and top-up money, can be seen using the view transaction button. The view transaction page is shown in Figure 16.

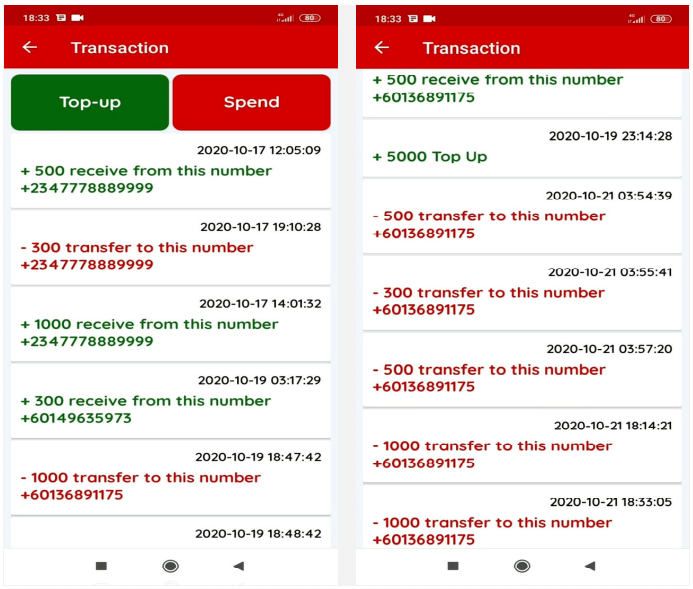


Figure 16. View transaction activity of Zamwallet.

Prototype Evaluation

To ensure their functionality, it is important to evaluate mobile apps. Evaluation is the main way to comprehend how users communicate with a specific interface when they use the system. For evaluating the entire design, we performed a functionality test. Functionality testing is a type of apps testing that validates the software system against functional requirements. The discussion here focuses on the essential implementing considerations of the functional aspects of the proposed device identity-based Zamwallet. The requirement test is performed to ensure that the system is executing as per expectations with no bugs or errors. This section describes the outcomes of the experiments on the prototype. The objectives of the evaluation are:

- To ensure that the functionality requirement is fulfilled in different Android versions/devices;
- To ensure that Zamwallet registered to a single entity/user cannot run on two devices at a time;
- To ensure that Zamwallet satisfies the conditions of registration, authentication, transfer of money, and top-up of money with the functionality test.

5. Result and Analysis

The proposed method uses four-authentication techniques in this study to secure Zamwallet e-wallet apps. This paper has demonstrated the system and pointed out its output using different functions. The proposed Zamwallet was installed on different Android devices to measure their performance for functionality. The main functionality and performance results are presented in the following subsections.

5.1. Experimental Result of OTP

We have evaluated the Zamwallet using three different mobile phone devices—Redmi, Vivo, and Neffos. The experimental OTP results of the Redmi user are shown in Table 3. Table 4 shows the experimental results on the Vivo device. Additionally, Table 5 shows the results on the Neffos device.

Table 3. The OTP results of the Redmi user.

	No	OTP	Status
User 1-Redmi	1	984913	Successful
	2	224264	Successful
	3	845862	Successful
	4	584132	Successful
	5	112617	Successful
	6	248980	Successful

Table 4. The OTP results of the Vivo user.

	No	OTP	Status
User 2- Vivo	1	884713	Successful
	2	639264	Successful
	3	389202	Successful
	4	184033	Successful
	5	714668	Successful
	6	297947	Successful

Table 5. The OTP results of the Neffos user.

	No	OTP	Status
User 3-Neffos	1	232975	Successful
	2	638645	Successful
	3	800352	Successful
	4	492469	Successful
	5	985624	Successful
	6	382478	Successful

5.2. Experiment Result of Top-Up Money

Seven transactions were completed in the top-up functionality involving different times and different amounts of money. The top-up money functionality was successfully completed using the PayPal payment gateway, and all the records were given by a PayPal sandbox account. We also evaluated transactions between the PayPal gateway and Zamwallet. The list of top-up money transactions is presented in Table 6.

Table 6. Top-up transaction results.

Currency MYR	Transaction ID	Item Title	Status
5000	3YC06329A41446107	Top-up	Completed
300	4JM153463M403735T	Top-up	Completed
500	50B06205AB1105831	Top-up	Completed
1000	9UD61707CH4090247	Top-up	Completed
700	4SB80386FE0038111	Top-up	Completed
1000	16D61116V0923694H	Top-up	Completed
100	50C8271251302894N	Top-up	Completed

5.3. Experiment of Transaction Money

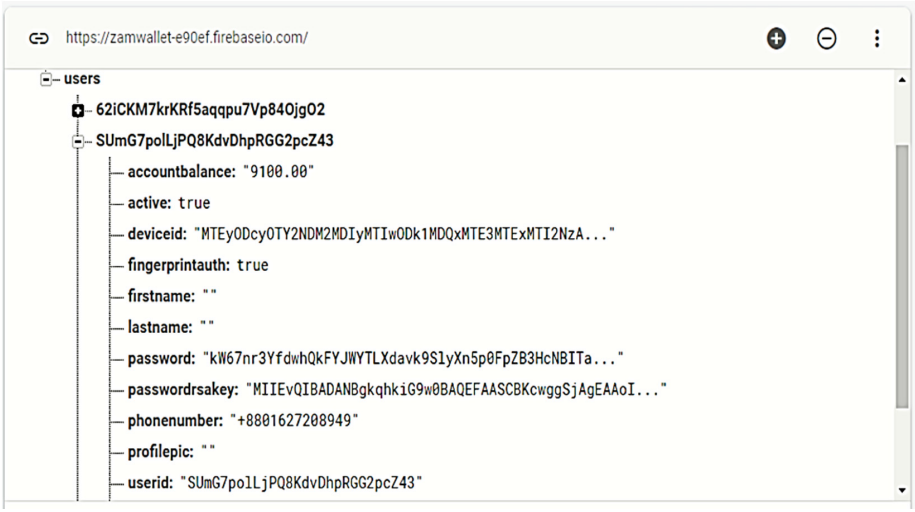
Eleven transactions were completed by the user registered phone numbers and currencies in the transfer functionality test. The transfer money functionality was successfully completed by transferring money from one Zamwallet user to another Zamwallet user. We hid the users' numbers to conceal their identities. Table 7 shows the overall characteristics of the transfer money functionality. For security purpose we hided user last 3 digits as ***.

Table 7. Money transaction results.

Number	Currency MYR	Status
+60777888999	500	Completed
+60777888999	300	Completed
+60777888999	1000	Completed
+60149635 ***	300	Completed
+60136891 ***	1000	Completed
+60136891 ***	500	Completed
+60136891 ***	500	Completed
+60136891 ***	300	Completed
+60136891 ***	500	Completed
+60136891 ***	1000	Completed
+60112126 ***	500	Completed

5.4. Experiment on Firebase Database

The admin must log in to enter the real-time database. To log into the Firebase console, the user must have a Google account. After successful login to the Firebase console, the user can select the database's features. Only an admin can monitor the system. Moreover, the admin could edit and delete user details from the system. Figure 17 shows each step of the real-time database of the proposed Zamwallet.



(**a**)

(**b**)

Figure 17. Overview of real-time database activities. (**a**): User information on Firebase database and (**b**): user transaction details on Firebase database.

As Figure 17a shows, the user device identity (ID) is encrypted with base64. However, Firebase real-time is very secure with support from Google. The user password is encrypted with RSA. Full form (NIST) has been proposing a minimum of 2048-bit keys for RSA since 2015, an upgrade from the accepted 1024-bit minimum recommendation from at least 2002 onwards [56,57]. RSA 2048 key is the most secure [58,59]. To secure interactions with the webserver, a 2048-bit RSA key and a self-signed SSL certificate are generated to encrypt all the full form (HTTPS) communications [60–62]. Figure 18 shows the performance for Zamwallet, and Figure 19 shows the real-time database overview.

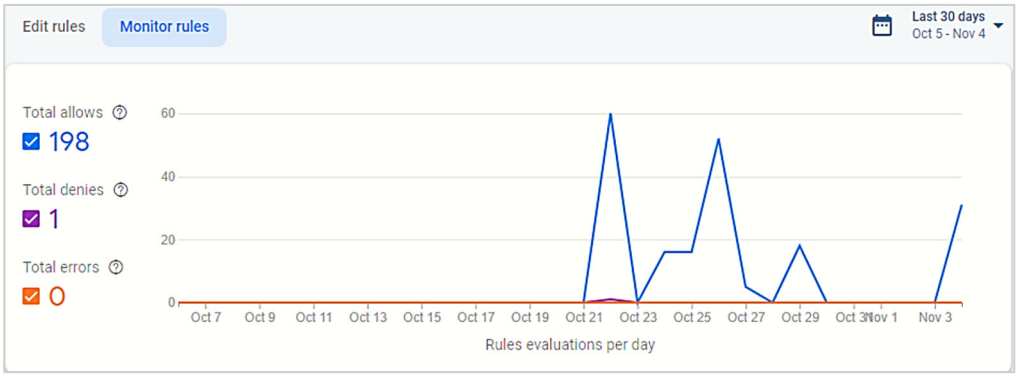


Figure 18. Screenshot of performance of Zamwallet.

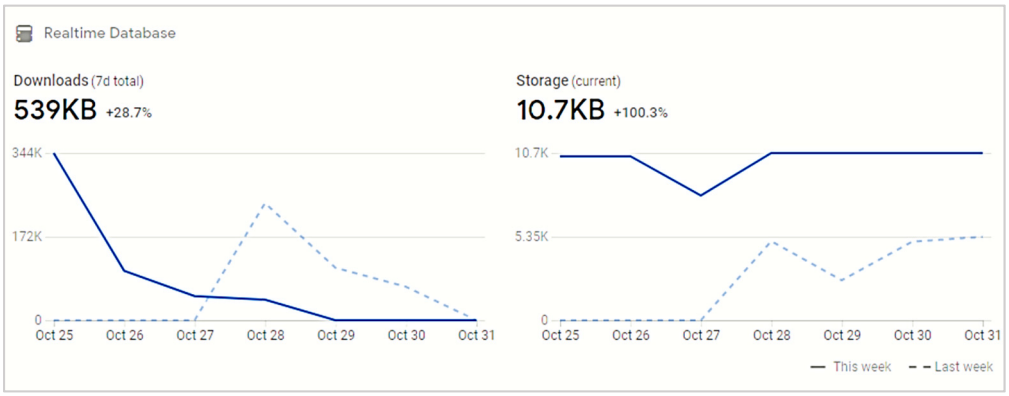


Figure 19. Screenshot of the overview of Zamwallet in the real-time database.

Total activity, total denies, and total errors of the system were operated and checked. After analyzing Zamwallet, it shows here that there is no error in the database so far.

The functionality test focuses on four areas. First is security management, which confirms that the system can create new users and properly manage their credentials in the registration phase. The second is identification and authentication, to ascertain that the registered details of the user can be authenticated. The third is the session, to confirm that the system can perform different kinds of transactions such as topping up money and transferring money. The fourth test and the main component of the system is a device identity based on user authentication, i.e., apps cannot run on two devices simultaneously. Only one user can use Zamwallet on their device. The user cannot create accounts on the same device using two different numbers. Table 8 shows the functionality test of the proposed method.

Table 8. Functionality test of Zamwallet.

Test	Test Case	Result
Functionality	Registration	Passed
	Authentication	Passed
	Top-up	Passed
	Transaction	Passed
	Profile/View Transaction	Passed
	Device Cloning prevention	Passed

Hence, results also show that the entire functionality test has passed. This shows that the system is operating as planned with no system bugs or errors.

6. Discussion

Authentication is becoming more and more necessary. Even in the modern age, most users rely on systems authentication and permission to add traditional passwords in multifactor operations. While there are still questions about privacy, safety, accessibility, and accuracy, full form (MFA) is becoming a system that guarantees modern uses of available security and efficiency for those who need these while accessing sensitive data. Biometrics is without question one of the primary foundations for MFA development. This capability is often seen to be an extension of standard security methods, such as passwords, security tokens, and PINs, and is not a standalone method. Once the user of an e-wallet app is authenticated, two or more authentication methods should be paired for better security.

The discussion here focuses on the key implementation and security aspects of the proposed device identity-based Zamwallet. The application covers four main authentication categories: knowledge (i.e., password), device identification (i.e., IMEI), biometrics (i.e., fingerprint), and ownership (i.e., OTP). Each of these techniques provides good security on their own, but a combination of two systems offers much more security against different attacks and types of social engineering while maintaining functionality and accessibility. This section outlines the research findings on the experiment on the prototype. In addition, the results are also linked to the literature review and research gaps. In the previous methods, various levels of authentication had been identified; as in, the authentication stage they had used was ownership or multifactor. Many of them have a framework-based theory, such as multifactor authentication, which is not applied in reality. The actual performance of these methods is not effective. There is also an opportunity to develop multifactor user authentication techniques. Here, we have used a user device ID and its interaction with individual user credentials covering the user's knowledge, ownership, and biometrics, which can be a good strategy to ensure a proper authentication process. We evaluated our results in Table 9, where we compared the existing authentication scheme and the proposed authentication scheme. The base method has three security features, and the proposed method has four security features.

(1) Knowledge: (U = username), (Pas = password), (Pi = pin), (Pa = pattern lock);
(2) Ownership: (O = SMS OTP), (S = smart cards), (H = hardware token);
(3) Biometrics: (Fi = fingerprint), (Fa = facial), (I = iris);
(4) Device ID: IMEI.

Table 9. Comparison of authentication categories within the proposed scheme and existing scheme.

Authors	Authentication Catagories											Total Point
	Knowledge				Ownership			Biometrics			Device ID	
	U	Pas	Pi	Pat	O	S	H	Fi	Fa	I	IMEI	
Harish et al., 2019						1			1			2
Vengatesan et al., 2020			1			1						2
Benli et al., 2017			1						1			2
Okpara and Bekaroo, 2017								1	1			2
L. Sharma and Mathuria, 2018	1		1						1			3
K. Tiwari, 2016									1			1
Gupta et al., 2020										1		1
Houngbo et al., 2019									1			1
Proposed Zamwallet			1			1		1			1	4

Table 9 shows that Zamwallet, at four points, has the most active points among authentication categories. The second authentication category, L. Sharma and Mathuria (2018), had three points. Harish et al. (2019), Vengatesan et al. (2020), and Benli et al. (2017) had similar authentication categories at two points. The rest of the authors had the same authentication categories, which was one point. Among all the proposed work, some provide a framework-based principle not applicable in actual life.

7. Conclusions and Future Work

Security is the main element associated with an e-wallet. The proposed system provides users with safe access to authorization through multifactor authentication. The proposed scheme verifies the user utilizing their password, fingerprint, and OTP. This approach improves upon the existing authentication methods. The proposed method improves the security of user authentication systems in e-wallet apps by covering four authentication categories. The implementation and evaluation process checked for stable and consistent functioning of the proposed Zamwallet. Specifically, users without extensive experience of using mobile phones will be comfortable using the apps after the initial glimpse. It ensures that unauthorized users cannot sniff or steal data, as they are being exchanged between the user device and database servers. In addition, the proposed solution has shown that the security performance of authentication and authorization could be dramatically improved relative to the existing systems. In comparison to other authentication schemes, the proposed Zamwallet has low effective cost. This is because, in the proposed Zamwallet, the user does not need an additional device to authenticate their fingerprint and hardware device. In this study, we presented the conceptual design of an authentication method in keeping with the viewpoint of the research of a device identity-based e-wallet. A prototype was designed and implemented to evaluate the proposed security method. Further investigations and enhancements could be considered in the future, as the proposed method is only for Android users. However, there is a possibility of developing the same for the iOS operating system. Mobile reloads, bill payments, blood donation features, etc. can be added as additional features for the proposed apps. Moreover, static and dynamic tools will be used to test the security of the proposed apps. The limitation of this study is that the proposed method cannot be used to access two separate devices and is not available for conventional mobile phones that do not support fingerprint recognition.

Author Contributions: Conceptualization, M.A.H.; Data curation, M.A.H. and Z.S.; Formal analysis, M.A.H. and Z.S.; Funding acquisition, Z.S.; Investigation, M.A.H.; Methodology, M.A.H.; Resources, M.A.H.; Software, M.A.H. and Z.S.; Supervision, Z.S.; Validation, M.A.H. and Z.S.; Writing—review and editing, M.A.H. and Z.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Malaysia Ministry of Education, Universiti Kebangsaan Malaysia, under grants, KKP 2020/UKM-UKM/4/3 and FRGS/1/2019/ICT01/UKM/01/2.

Conflicts of Interest: The authors declare no conflict of interest regarding this paper.

References

1. Cole, A.; Mcfaddin, S.; Narayanaswami, C.; Tiwari, A.; Heights, Y.; Cole, A.; Mcfaddin, S.; Narayanaswami, C.; Tiwari, A. *IBM Research Report toward a Mobile Digital Wallet*; Research Report; IBM: New York, NY, USA, 2009; p. 24965.
2. Batra, R.; Kalra, N. Are Digital Wallets the New Currency? *Apeejay J. Manag. Technol.* **2016**, *11*, 1–12.
3. Hassan, A.; Shukur, Z. Review of Digital Wallet Requirements. In Proceedings of the 2019 International Conference on Cybersecurity (ICoCSec), Negeri Sembilan, Malaysia, 25–26 September 2019; pp. 43–48. [CrossRef]
4. Omariba, Z.B.; Masese, N.B. Security and Privacy of Electronic Banking. *Kidney Int. Suppl.* **2013**, *3*, 262. [CrossRef]
5. Hassan, Z.S.A. A Review on Electronic Payments Security. *Symmetry* **2020**, *12*, 1344. [CrossRef]
6. Ometov, A.; Bezzateev, S.; Mäkitalo, N.; Andreev, S.; Mikkonen, T.; Koucheryavy, Y. Multi-Factor Authentication: A Survey. *Cryptography* **2018**, *2*, 1. [CrossRef]
7. Fan, K.; Li, H.; Jiang, W.; Xiao, C.; Yang, Y. U2F based secure mutual authentication protocol for mobile payment. In Proceedings of the ACM Turing 50th Celebration Conference—China, Shanghai, China, 12–14 May 2017.
8. Shaju, S.; Panchami, V. BISC authentication algorithm: An efficient new authentication algorithm using three factor authentication for mobile banking. In Proceedings of the 2016 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, India, 19 November 2016; pp. 1–5. [CrossRef]
9. Okpara, O.S.; Bekaroo, G. Cam-Wallet: Fingerprint-based authentication in M-wallets using embedded cameras. In Proceedings of the 2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe), Milan, Italy, 6–9 June 2017. [CrossRef]
10. Khattri, V.; Singh, D.K. Implementation of an Additional Factor for Secure Authentication in Online Transactions. *J. Organ. Comput. Electron. Commer.* **2019**, *29*, 258–273. [CrossRef]
11. Harish, M.; Karthick, R.; Rajan, R.M.; Vetriselvi, V. *A New Approach to Securing Online Transactions—The Smart Wallet*; Springer: Singapore, 2019; Volume 500, ISBN 978-981-13-0211-4.
12. Newcomb, A. Phishing Scams Can Now Hack Two-Factor Authentication | Fortune. Available online: https://fortune.com/2019/06/04/phishing-scam-hack-two-factor-authentication-2fa/ (accessed on 21 March 2020).
13. Wang, F.; Shan, G.B.; Chen, Y.; Zheng, X.; Wang, H.; Mingwei, S.; Haihua, L. Identity Authentication Security Management in Mobile Payment Systems. *J. Glob. Inf. Manag.* **2020**, *28*, 189–203. [CrossRef]
14. Huseynov, E.; Seigneur, J.-M. Physical presence verification using TOTP and QR codes. In Proceedings of the 34th International Conference on ICT Systems Security and Privacy Protection, Lisbon, Portugal, 25–27 June 2019.
15. Kaur, N.; Devgan, M. A Comparative Analysis of Various Multistep Login Authentication Mechanisms. *Int. J. Comput. Appl.* **2015**, *127*, 20–26. [CrossRef]
16. Emeka, B.O.; Liu, S. Security Requirement Engineering Using Structured Object-Oriented Formal Language for M-Banking Applications. In Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), Prague, Czech Republic, 25–29 July 2017; pp. 176–183.
17. Ali, M.A.; Arief, B.; Emms, M.; Van Moorsel, A. Does the Online Card Payment Landscape Unwittingly Facilitate Fraud? *IEEE Secur. Priv. Mag.* **2017**, *15*, 78–86. [CrossRef]
18. *Enisa Security of Mobile Payments and Digital Wallets*; European Union Agency for Cybersecurity (ENISA): Athens, Greece, 2016; ISBN 978-92-9204-199-1.
19. Sudar, C.; Arjun, S.K.; Deepthi, L.R. Time-based one-time password for Wi-Fi authentication and security. In Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13–16 September 2017; pp. 1212–1216. [CrossRef]
20. Kogan, D.; Manohar, N.; Boneh, D. T/Key: Second-Factor Authentication from Secure Hash Chains Dmitry. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 983–999.
21. Isaac, J.T.; Sherali, Z. Secure Mobile Payment Systems. *IT Prof.* **2014**, *16*, 36–43. [CrossRef]
22. Dwivedi, A.; Dwivedi, A.; Kumar, S.; Pandey, S.K.; Dabra, P. A Cryptographic Algorithm Analysis for Security Threats of Semantic E-Commerce Web (SECW) for Electronic Payment Transaction System. *Adv. Intell. Syst. Comput.* **2013**, *2013*, 367–379. [CrossRef]
23. Yang, W.; Li, J.; Zhang, Y.; Gu, D. Security analysis of third-party in-app payment in mobile applications. *J. Inf. Secur. Appl.* **2019**, *48*, 102358. [CrossRef]
24. Gualdoni, J.; Kurtz, A.; Myzyri, I.; Wheeler, M.; Rizvi, S. Secure Online Transaction Algorithm: Securing Online Transaction Using Two-Factor Authentication. *Procedia Comput. Sci.* **2017**, *114*, 93–99. [CrossRef]
25. Venugopal, H.; Viswanath, N. A robust and secure authentication mechanism in online banking. In Proceedings of the 2016 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, India, 19 November 2016; pp. 1–3. [CrossRef]

26. Roy, S.; Venkateswaran, P. Online payment system using steganography and visual cryptography. In Proceedings of the 2014 IEEE Students' Conference on Electrical, Electronics and Computer Science, Bhopal, India, 1–2 March 2014; pp. 1–5. [CrossRef]
27. Hassan, A.; Shukur, Z.; Hasan, M.K.; Hassan, A. An Efficient Secure Electronic Payment System for E-Commerce. *Computer* **2020**, *9*, 66. [CrossRef]
28. Ataya, M.A.M.; Ali, M.A.M. Acceptance of Website Security on E-banking. A-Review. In Proceedings of the 2019 IEEE 10th Control and System Graduate Research Colloquium (ICSGRC), Shah Alam, Malaysia, 2–3 August 2019; pp. 201–206. [CrossRef]
29. Hassan, A.; Shukur, Z.; Kamrul, M. An Improved Time-Based One Time Password Authentication Framework for Electronic Payments. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 359–366. [CrossRef]
30. Chaudhry, S.A.; Farash, M.S.; Naqvi, H.; Sher, M. A secure and efficient authenticated encryption for electronic payment systems using elliptic curve cryptography. *Electron. Commer. Res.* **2016**, *16*, 113–139. [CrossRef]
31. Skračić, K.; Pale, P.; Kostanjčar, Z. Authentication approach using one-time challenge generation based on user behavior patterns captured in transactional data sets. *Comput. Secur.* **2017**, *67*, 107–121. [CrossRef]
32. Ibrahim, R.M. A Review on Online-Banking Security Models, Successes, and Failures. In Proceedings of the 2nd Interantional Conference on Inventive Systems and Control (ICISC—2018), Coimbatore, India, 19–20 January 2018.
33. Elliot, M.; Talent, K. A robust and scalable four factor authentication architecture to enhance security for mobile online transaction. *Int. J. Sci. Technol. Res.* **2018**, *7*, 139–143.
34. Kanimozhi, G.K.K. Security Aspects of Mobile Based E Wallet. *Int. J. Recent Innov. Trends Comput. Commun.* **2017**, *5*, 1223–1228.
35. Tan, S.F.; Samsudin, A. Enhanced Security of Internet Banking Authentication with EXtended Honey Encryption (XHE) Scheme. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*; Springer International Publishing: New York, NY, USA, 2017; pp. 201–216. [CrossRef]
36. Tellini, N.; Vargas, F. *Two-Factor Authentication: Selecting and Implementing a Two-Factor Authentication Method for a Digital Assessment Platform*; KTH Royal Institute of Technology: Stockholm, Sweden, 2017.
37. Huseynov, E.; Seigneur, J.-M. *Context-Aware Multifactor Authentication Survey*; Elsevier BV: Amsterdam, The Netherlands, 2017; pp. 715–726.
38. Wang, C.; Wang, Y.; Chen, Y.; Liu, H.; Liu, J. User authentication on mobile devices: Approaches, threats and trends. *Comput. Networks* **2020**, *170*, 107118. [CrossRef]
39. Mohammed, A.J.; Yassin, A.A. Efficient and Flexible Multi-Factor Authentication Protocol Based on Fuzzy Extractor of Administrator's Fingerprint and Smart Mobile Device. *Cryptography* **2019**, *3*, 24. [CrossRef]
40. Dasgupta, D.; Roy, A.; Nag, A. Toward the design of adaptive selection strategies for multi-factor authentication. *Comput. Secur.* **2016**, *63*, 85–116. [CrossRef]
41. Nwabueze, E.E.; Obioha, I.; Onuoha, O. Enhancing Multi-Factor Authentication in Modern Computing. *Commun. Netw.* **2017**, *9*, 172–178. [CrossRef]
42. Borrego, C.; Amadeo, M.; Molinaro, A.; Jhaveri, R.H. Privacy-Preserving Forwarding Using Homomorphic Encryption for Information-Centric Wireless Ad Hoc Networks. *IEEE Commun. Lett.* **2019**, *23*, 1708–1711. [CrossRef]
43. Benli, E.; Engin, I.; Giousouf, C.; Ulak, M.A.; Bahtiyar, S. BioWallet: A Biometric Digital Wallet. In Proceedings of the Twelfth International Conference on Systems (Icons 2017), Venice, Italy, 23–27 April 2017; pp. 38–41.
44. Alibabaee, A.; Broumandnia, A. Biometric Authentication of Fingerprint for Banking Users, Using Stream Cipher Algorithm. *J. Adv. Comput. Res.* **2018**, *9*, 1–17.
45. Houngbo, P.J.; Hounsou, J.T.; Damiani, E.; Asal, R.; Cimato, S.; Frati, F.; Yeun, C.Y. *Embedding a Digital Wallet to Pay-with-aSelfie, from Functional Requirements to Prototype*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; Volume 206, ISBN 978-3-319-67836-8.
46. Vengatesan, K.; Kumar, A.; Parthibhan, M. *Advanced Access Control Mechanism for Cloud Based E-Wallet*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; Volume 31, ISBN 978-3-030-24642-6.
47. Patel, C.; Joshi, D.; Doshi, N.; Veeramuthu, A.; Jhaveri, R. An enhanced approach for three factor remote user authentication in multi—Server environment. *J. Intell. Fuzzy Syst.* **2020**, *39*, 8609–8620. [CrossRef]
48. Alzu'Bi, A.; Albalas, F.; Al-Hadhrami, T.; Younis, L.B.; Bashayreh, A. Masked Face Recognition Using Deep Learning: A Review. *Electronics* **2021**, *10*, 2666. [CrossRef]
49. Wang, Z.; Zhang, X.; Yu, P.; Duan, W.; Zhu, D.; Cao, N. A New Face Recognition Method for Intelligent Security. *Appl. Sci.* **2020**, *10*, 852. [CrossRef]
50. Massaro, A.; Galiano, A. Image Processing and Post-Data Mining Processing for Security in Industrial Applications. *Adv. Malware Data-Driven Netw. Secur.* **2020**, 117–146. [CrossRef]
51. Sharma, L.; Mathuria, M. Mobile banking transaction using fingerprint authentication. In Proceedings of the 2018 2nd International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 19–20 January 2018; pp. 1300–1305.
52. Tiwari, K. *Secure Digital Wallet Authentication Protocol*; DalSpace: Bedford, Nova Scotia, Canada, 2016.
53. Gupta, A.; Kaushik, D.; Gupta, S. Integration of Biometric Security System to Improve the Protection of Digital Wallet. *SSRN Electron. J.* **2020**, 1–6. [CrossRef]
54. Churcher, A.; Ullah, R.; Ahmad, J.; Rehman, S.U.; Masood, F.; Gogate, M.; Alqahtani, F.; Nour, B.; Buchanan, W. An Experimental Analysis of Attack Classification Using Machine Learning in IoT Networks. *Sensors* **2021**, *21*, 446. [CrossRef]

55. Massaro, A.; Gargaro, M.; Dipierrro, G.; Galiano, A.M.; Buonopane, S. Prototype Cross Platform oriented on Cybersecurity, Virtual Connectivity, Big Data and Artificial Intelligence Control. *IEEE Access* **2020**, *8*, 197939–197954. [CrossRef]
56. Qiu, S.; Liu, Q.; Zhou, S.; Wu, C. Review of Artificial Intelligence Adversarial Attack and Defense Technologies. *Appl. Sci.* **2019**, *9*, 909. [CrossRef]
57. Tirtea, R. *Algorithms, Key Sizes and Parameters Report*; European Union Agency for Cybersecurity (ENISA): Athens, Greece, 2013; pp. 1–5.
58. Sönmez, F.; Abbas, M.K. Development Of A Client/Server Cryptography-Based Secure Messaging System using RSA Al-gorithm. *J. Manag. Eng. Inf. Technol.* **2017**, *4*, 6.
59. Nwoye, C.J. Design and Development of an E-Commerce Security using RSA Cryptosystem. *Int. J. Innov. Res. Inf. Secur.* **2015**, *2*, 2349–7017.
60. Aina, F.; Yousef, S.; Osanaiye, O. *Design and Implementation of Challenge Response Protocol for Enhanced e-Commerce Security*; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; Volume 3, ISBN 9783030026837.
61. Massaro, A. *Electronics in Advanced Research Industries: Industry 4.0 to Industry 5.0 Advances*; John Wiley & Sons: Hoboken, NJ, USA, 2021. [CrossRef]
62. Hassan, A.; Shukur, Z. A Secure Multi Factor User Authentication Framework for Electronic Payment System. In Proceedings of the 2021 3rd International Cyber Resilience Conference (CRC), Langkawi Island, Malaysia, 29–31 January 2021; pp. 1–6. [CrossRef]