# **A MFA USING TAP SEQUENCE AND BARCODE AUTHENTICATION**

**BY OCTAGON TEAM**

# **Introduction**

Password capture is one of the simplest and most convenient authentication mechanisms to deal with the confidentiality of data. Adequate and reliable authentication is the key point for protecting computing resources. There are various authentication techniques that have been used in different scenarios: Personal Identification Number (PIN), alphanumeric passwords, token-based, and graphical passwords. These techniques are easy to implement but they have been subjected to several weaknesses in terms of security threats. The research implements a Multi-Factor Authentication framework aimed to improve some variables in the authentication scheme: usability (by reducing authentication time), wide search space (infinite possible rhythms), reducing shoulder surfing attacks, smudge attack proof, and keylogging proof.

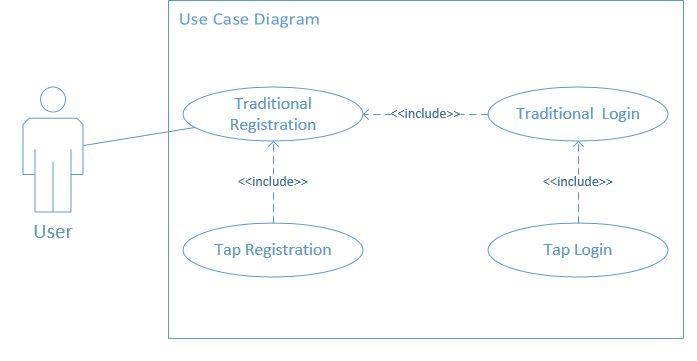


Fig.1 Use Case diagram

**Table 1 Use Case Description for Register Tap Sequence**

| Use Case ID | CASE\_001 |
| --- | --- |
| Use Case Name | Traditional Registration |
| Description | This use case allows the user to register with email and password |
| Primary Actor | User |
| Secondary Actor | None |
| Pre-Conditions | CASE\_001 |
| Post-Condition | The system stores the email and password in the data store |

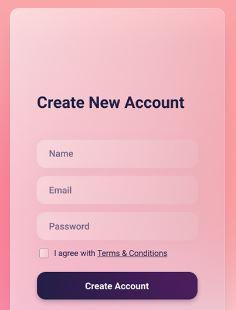
| Use Case ID | CASE\_002 |
| --- | --- |
| Use Case Name | Tap Registration |
| Description | This use case allows the user to tap on specific areas on the screen three times in order to create a rhythm for the password. Once the user is tapping the screen the system is determining the number of taps made, the number of taps remaining, and the seconds between taps. Once the tap has gotten to three, the system prompts the user to confirm the rhythm. If the user confirms, then the rhythm is saved in the data store. If for any reason the user does not confirm, then the authentication parameters such as the tapCount and other necessary data types are then cleared |
| Primary Actor | User |
| Secondary Actor | None |
| Pre-Conditions | CASE\_001 |
| Post-Condition | The system saves rhythm or clears authentication parameters |

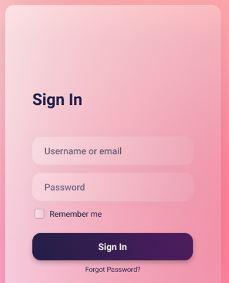
| Use Case ID | CASE\_003 |
| --- | --- |
| Use Case Name | Traditional Login |
| Description | This use case allows the user to input their email and password. The system verifies the inputted details against instances in a data store. |
| Primary Actor | User |
| Secondary Actor | None |
| Pre-Conditions | CASE\_001 |
| Post-Condition | The system grants access to tap login phase if a match is found, otherwise; access denied |

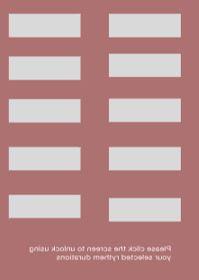
| Use Case ID | CASE\_004 |
| --- | --- |
| Use Case Name | Tap Login |
| Description | This use case allows the user to tap on the screen three times in order to be authenticated. Once the user is tapping the screen the system is determining the number of taps made, the number of taps remaining, and the seconds between the taps in seconds. In addition, the system also determines the specific area of the screen the user is tapping on. Once the tap has gotten to three, the system then verifies the tap time difference and screen tap location |
| Primary Actor | User |
| Secondary Actor | None |
| Pre-Conditions | CASE\_002, CASE\_003, |
| Post-Condition | The system shall authorize the user if match found, otherwise; access denied |

**Table 2 Requirement Specification**

| **ID** | **Requirement Description** |
| --- | --- |
| R\_01 | The system shall allow users to tap on screen to create a rhythm password |
| R\_02 | They system shall allow users to pick specific points to tap on the screen. |
| R\_03 | The system shall determine the tap time difference of rhythm to represent a password |
| R\_04 | The system shall save the corresponding tap time difference and user specific points in the storage |
| R\_05 | On the login screen, the system shall allow a user to tap on screen |
| R\_06 | On the login screen, the system shall allow a user pick specific points to tap on screen |
| R\_07 | After the user taps on screen to login, the system shall determine the time difference between rhythm taps on the screen |
| R\_08 | The system shall grant access if the tap time difference of inputted rhythmical password exists in the database and also the screen tap location. |
| R\_09 | The system shall deny access R\_08 fails |
| R\_10 | The system will have a user-friendly interface |
| R\_11 | The UI of the system will be responsive: must have a consistent look on all screen size |
| R\_12 | The UI will have interactive controls and animations |
| R\_13 | The system will produce helpful error message where applicable |
| R\_14 | The system must be easy to navigate to and from its various pages |
| R\_15 | The system will be able to provide feedback to users based on actions carried out by the users |
| R\_16 | The system would hash the password of a user with a hashing algorithm and salting |
| R\_17 | The system shall allow users to register with email and password |
| R\_18 | The system shall allow users to login with email and password |



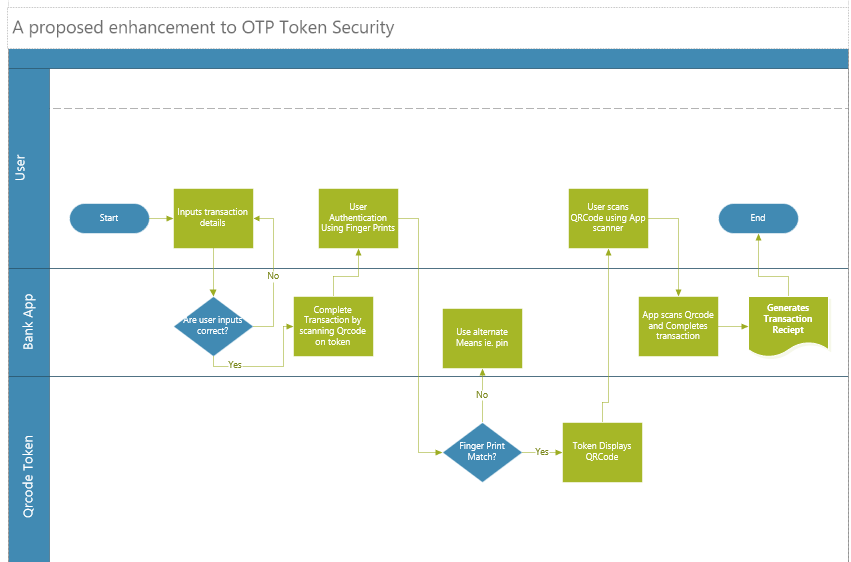




**Introduction**

Token-based authentication is a mechanism that allows users to prove their identity and obtain a unique access one time password(OTP) in exchange. During the OTP’s lifetime, users may access the website or app for which the token was issued, rather than having to re-enter credentials every time they return to the same webpage, app, or other resource secured by that same token. Authentication tokens function similarly to a stamped ticket. As long as the token is valid, the user has access. The OTP is invalidated when the user logs out or exits an app. Token-based authentication differs from standard password- or server-based authentication methods. Tokens provide an additional degree of protection, and administrators have complete control over every activity and transaction.

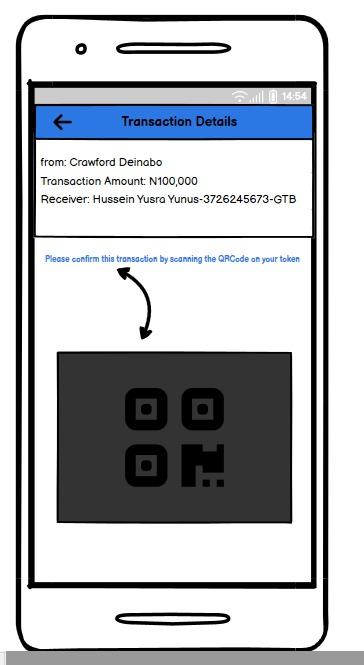
This research, on the other hand, tries to improve and validate the existing hardware token-based authentication. The suggested model is a hardware authentication that acts as an extra layer of security by scanning the user's finger to authenticate user identity before creating the QR-Code necessary for the transaction. Unlike conventional token-based authentication, which generates a set of numbers that are vulnerable to attacks such as shoulder surfing/theft and brute-force, particularly if the numbers are short, this suggested architecture incorporates these types of assaults into the design model.



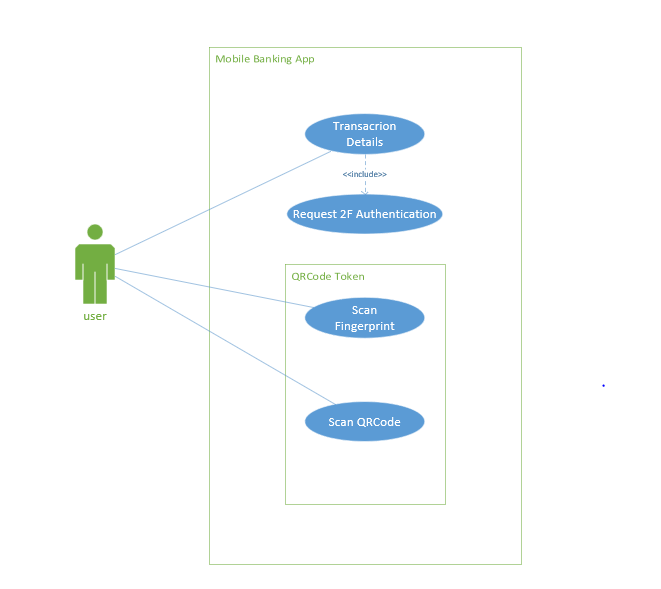
Process Flow Diagram



Device sample Diagram



Mobile Phone with ongoing transaction



Uses case diagram

Use case diagram FOR the QR- Code Application

| USE CASE ID | CASE-001 |
| --- | --- |
| USE CASE NAME | Transaction Details |
| DESCRIPTION | Use case allows the user to input details needed for a Debit transaction i.e. receiver's account number, receivers Bank name and Amount to transfer to receiver. |
| TRIGGER | Login |
| PRIMARY ACTOR | User |
| SECONDARY ACTOR | None |
| PRECONDITION | CASE-001 |
| POST-CONDITION | System checks details inserted by the user and verifies that it matches a legitimate user of a bank selected and requests transaction approval. |

| USE CASE ID | CASE-002 |
| --- | --- |
| USE CASE NAME | Request 2F Authentication |
| DESCRIPTION | Use case redirects user to a hardware token to re-verify authentication as a second layer of security |
| TRIGGER | Verified transaction details |
| PRIMARY ACTOR | User |
| SECONDARY ACTOR | None |
| PRECONDITION | CASE-001 |
| POST-CONDITION | QR code token verifies User |

| USE CASE ID | CASE-003 |
| --- | --- |
| USE CASE NAME | Scan Fingerprint |
| DESCRIPTION | QR-code scanner needs to authenticate the user to make sure the user that initiated the transaction is the same user trying to approve it. |
| TRIGGER | 2f authentication Request |
| PRIMARY ACTOR | User |
| SECONDARY ACTOR | None |
| PRECONDITION | CASE-002 |
| POST-CONDITION | System displays QR code |

| USE CASE ID | CASE-004 |
| --- | --- |
| USE CASE NAME | Scan QR code |
| DESCRIPTION | Token displays a QRcode and user uses mobile App to scan the displayed QR code. The purpose of this, is to identify user and approve transactions on mobile App. |
| TRIGGER | Verified Finger print |
| PRIMARY ACTOR | User |
| SECONDARY ACTOR | None |
| PRECONDITION | CASE-003 |
| POST-CONDITION | System Approves transaction and makes Appropriate Debit |

Requirement specification

| Req-001 | System shall be verify user thumbprint |
| --- | --- |
| Req-002 | System shall produce unique QRcodes each time a transaction is triggered |
| Req-003 | System shall allow user scan QRcode |
| Req-004 | System shall approve transaction after QRcode has been scanned |

**Combined Architecture**

*Step 1: perform a traditional registration*

*Step 2: perform a tap sequence registration*

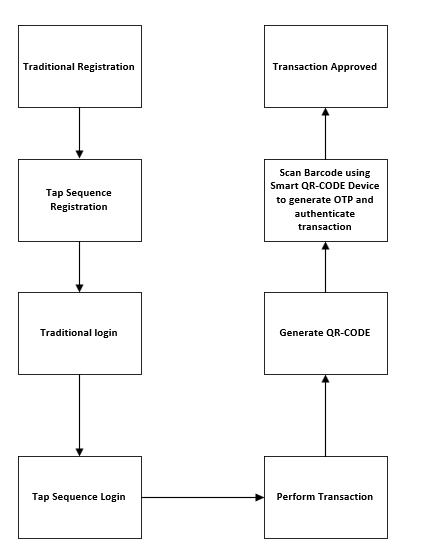
*Step 3: Login using traditional login*

*Step 4: login using the tap sequence*

*Step 5: perform transaction*

*Step 6: generate barcode*

*Step 7: Use a smart device given by the bank to authenticate by scanning the barcode(i.e like whatsapp web barcode scanner).*



**link to the figma prototype:**

<https://www.figma.com/proto/Wf6igOhCiEs1qLDfui64q2/Untitled?node-id=1%3A16&scaling=scale-down&page-id=0%3A1>