Requirements for the User App and the Backend Windows Desktop/Server App

Process Flow

|  |  |  |
| --- | --- | --- |
| Step # | Server App | User App (Client App) |
|  | Installation happens on Windows machine (Desktop/Server)  Q: (a) For server app, is setup program required or manual deployment is sufficient? (b) What is the Windows Desktop/Server version the server app is running on?  Ans: Server app requires installable that is used to manually install. Right now, I don’t need auto deploy. Win 8 and Win Server 2012 are the oldest versions it should run on. |  |
|  | Configuration of PIN and SECRET Q & A happens on first launch. |  |
|  | Can be closed and restarted OR continue to run as chosen by the Server App owner. If restarted, PIN will be used to logon.  Q: Server app is run, closed or restarted by user manually?  Ans: Server app is run, closed and restarted manually. |  |
|  | Server is listening for any communication from clients |  |
|  |  | Client Installation happens |
| FIRST USER APP (CLIENT APP) LAUNCH | | |
|  |  | Configuration of PIN and SECRET Q & A happens on first launch. |
|  |  | Server details configured at first launch. |
|  |  | Device Id generated and username accepted from User App owner at the first launch and stored. |
|  | Server knows the new User App on new device. Receives the username, device id | Server contacted using TLSv1.2 on first launch using configured details and sends username and Device Id. |
|  | Establish secure connection – an additional layer of security aside from TLSv1.2. | Establish secure communication connection – an additional layer of security aside from TLSv1.2. Currently there is only function stub. |
|  |  | Creates identity (RANDOM 10 character string). This 10 character string is generated in a function stub currently that will be elaborated by me later.  Hash of the full identity (10 character string) is computed and stored.  Broken into two parts 5 character each. Part 1 & Part 2 are identified |
|  | Encrypted Part 1 received and stored. | Symmetric Encryption Key for Part 1 is accepted from owner.  Part 1 is encrypted and sent the encrypted Part 1 to Server App.  Encrypted Part 1 is deleted (secure delete). |
|  | Symmetric Encryption Key for Part 2 is automatically generated, stored and sent to User App. | Receives the Symmetric Encryption Key for Part 2.  Encrypts Part 2, stores it and deletes (secure delete) Symmetric Encryption Key for Part 2.  Deletes (secure delete) unencrypted Part 1 and Part 2. |
| The User Authentication Performed by User App Including Its Trigger | | |
|  | Either Windows Azure AD or the dummy Web App will send an authentication request to the Server App with the username. |  |
|  | Server App send sends notification to all Device Ids registered by the username for an authentication. | User App (Client App) receives the authentication request.  Q: What happens if the user app is not active (Forced closed, after device reboot state) when the notification is sent?  Ans: The notification to the User Apps when clicked should be start the User App again. |
|  |  | User App (Client App) requests the User App owner to provide the PIN to login. |
|  | Server matches up with username and Device Id and fetches the requested information from the database (stored encrypted in the database record) and sends it back to the User App. | User App (Client App) sends username and Device Id and requests for Encrypted Part 1 and Symmetric Encryption Key for Part 2. |
|  |  | Decrypts stored Part 2 and Part 1 and then puts them together and hashes it, securely deletes the plain text identity brought together and plain text parts to compare against the hashed identity to check if that matches. |
|  | Server receives authentication result. | Notifies Server App about success or failure of the authentication process. |
|  | Server notifies the authentication request provider of the result. |  |
|  |  |  |
|  |  |  |
|  |  |  |

We will develop a User App (Android, iOS and Windows Desktop/Server) that will do the following:

1. At the time of application prelaunch for the first time,
   1. it will generate a device identity corresponding to the device it is installed on (Android, iOS or Windows Desktop/Server) that will be used during all future authentication requests. - I will populate the algorithm’s code to create identity. For the purpose of this exercise, you can create a function stub and call it. I will populate the function stub code. The function stub can return a 10-character constant string.
   2. This generated identity will then be broken into two parts. The first part will be sent to the Backend for storage in encrypted format. The second part will be stored encrypted locally on the device. The encryption algorithm should be symmetric encryption that takes a key as input for the encryption. The encryption key should be different for both ends – the Backend and the device. It will also store the hashed version of the identity (of the two parts together). It will also generate a Device Id and send it to Backend. Backend will use this Device Id to uniquely identify the device.

Q: Who will provide the encryption keys?

Ans: The identity owner – The User App owner. The person who owns the client devices (iOS, Android or Windows App)

* 1. It will inform the Device owner when the secure communication establishment is successful.
  2. It will establish a secure connectivity with the Backend in addition the TLSv1.2 protocol it will use to connect with the Backend. This will be used in all future communications with Backend. – I will populate the algorithm’s code to establish secure connectivity. For the purpose of this exercise, you can create a function stub and call it. I will populate the function stub code.
  3. It will ask the Device owner to configure the PIN that will be stored in a separate location as compared to the rest of the data in a hashed form. Whenever owner enters PIN in the future, the hashed PIN will be compared to check for valid entry.
  4. It will ask the owner for the Device PIN recovery questions and answers. They will be stored hashed along with the hashed PIN.

1. The Device will store the username assigned by the Device owner, Device Id, hashed combined identity, encrypted second part of the identity and first part identity’s key in encrypted form in a SECRET (not publicised) location.
2. The encrytion key for second part identity encryption is generated at Backend. Backend will store username, device Id, second part identity’s encryption key, encrypted first part of the identity (using symmetric encryption algorithm with key) as a record for this device in its database. Columns containing second part identity’s encryption key, encrypted first part of identity should be stored encrypted in the database. The first part identity’s encryption key will never reach the Backend.
3. In the future, when the Device is requested to perform an authentication, the device will gather the two pieces of identity (Backend and Device) and verify against the hashed identity. The result of the match will be communicated with the Backend.
4. When the authentication request is received by the Device, the following steps must be carried out:
   1. Ask the user for the username and PIN to access the encrypted first identity component.
   2. Then the App will communicate with the Backend to get all data stored for this device’s record.
   3. Device will compute the hash of the combined identity to match and establish successful authentication.
5. It will have a UI that will allow the Device owner to create a username and other owner configured items. The username and Device Id should also be stored at both ends – Device and Backend.
6. If the user needs to change the PIN, they should know the answer to the secret questions. Similarly, if they have to change the secret question, they must know the PIN. If they do not know either of PIN or secret questions, then they cannot change both at this stage. We will add a feature for this scenario in a later version. We can have a list of questions as part of the App building.
7. I should be notified if there are certain version of Devices where this App will NOT work during development work.
8. The UI is not defined yet, but should be professional.
9. The source code should be amply commented.
10. I should have the ability to add these Apps to respective App stores after development. All compliance requirements must be met for that. Changing code to ensure compliance will not be considered an iteration.
11. All identified bugs must be fixed independent of the number of iterations required. We will set a mutually agreed time limit for identifying bugs based on the delivery timeline.
12. There should be logo placeholders in all pages including the logon page.

We will develop a Backend Windows Desktop/Server App that will do the following:

Q: Is there a preference on platform or programming languages used for development?

Ans: Not particularly. But use of popular languages, development environment (free ones) and tools are required so it is easy to maintain and enhance going forward. We prefer use of Flutter for all development.

1. It will establish a secure connectivity with the User App when at a User App pre-launch in addition the TLSv1.2 protocol it will use to connect with the User App. This will be used in all future communications with User App. – I will populate the algorithm’s code to establish secure connectivity. For the purpose of this exercise, you can create a function stub and call it. I will populate the function stub code.

The code in the function stub I will populate later. But during this code development (for the purpose of this exercise) a function code stub should be provided and called, so that when the function is populated, then the code structure will not require any change.

1. Backend will store username, device Id, second part identity’s encryption key (after randomly generating), encrypted first part of the identity (using symmetric encryption algorithm with key) as a record for this device in its database. Columns containing second part identity’s encryption key, encrypted first part of identity should be stored encrypted in the database. The first part identity’s encryption key will never reach the Backend. The database can be a conventional database with a query language supported. No additional SQL Query is expected to be written other than for the functionality of these components.

Q: The second part of client’s identity will be encrypted at server side and then stored in database column?

Ans: Encrypted Key Part 1 is stored at Server (this App). The encryption key is provided by client (User App) and stored at client (User App). When these data are stored on the server side, it will be stored in encrypted columns – an additional encryption.

Encrypted Key Part 2 is stored at client (User App). The encryption key is provided by server (this App) and stored at the server (this App)

Q: Encryption key used to encrypt second part of client’s identity is shared for all clients or different encryption key for each client?

Ans: Different for each client.

Q: All columns: encryption key for second part of client, encrypted second part of client identity, encrypted first part of client identity will be stored in encrypted columns in database?

Ans: Specified columns are encrypted columns. Other columns get stored unencrypted.

1. It will also display the username and Device Id of a User App when a new device is registered, either successfully or otherwise.
2. It will display a list of registered devices with a simple search facility.
3. It will show that the authentication request was successful or not.

Q: Are these on the same UI page? (4,5,6)

Ans: It does not matter whether #4, #5 and #6 are developed on separate pages or the same. The UI should be intuitive and easy to comprehend and understand.

1. It will ask the Backend installer to configure the PIN that will be stored in a separate location as compared to the rest of the data in a hashed form. Whenever owner enters PIN in the future, the hashed PIN will be compared to check for valid entry.
2. It will ask the Backend installer PIN ~~recovery~~ RESET questions and answers. They will be stored hashed along with the hashed PIN.

Q: When logging in to the server using UI for the first time, user will be asked to configure PIN and recovery questions and answers? PIN and answers will hashed and stored in separate location.

Ans: This feature is to secure access to the server (this App). It should be stored separately on the server App’s device so that if a hacker gets access to one, they don’t get access to the other too. They will have to separately hack to get access to the other component (PIN, say).

Q: Upon successful PIN recovery, user creates new PIN, PIN recovery questions and answers? (6, 7)

Ans: The SECRET Questions and Answers are to reset the forgotten PIN. The PIN is never stored encrypted or in plain text, ONLY in hashed form. So, the user will never actually recover the PIN, but can reset after providing the answers to the SECRET questions. Answers to the SECRET Questions will also be stored hashed only. So, they too can never be recovered. They can be reset by providing correct PIN. However, the user loses PIN and SECRET Answers to Questions simultaneously, they cannot reset both. At this stage we will not provide a way to RESET from this situation. The solution for this will be implemented in the next version of the solution later.

1. I should be notified if there are certain version of Devices where this App will NOT work during development work.
2. The UI is not defined yet, but should be professional.
3. The source code should be amply commented.
4. All identified bugs must be fixed independent of the number of iterations required. We will set a mutually agreed time limit for identifying bugs based on the delivery timeline.
5. There should be logo placeholders in all pages including the logon page.
6. The Server App MUST be able to accept multiple simultaneous requests from different User Apps. The Server App MUST be able to service them all. Assume that maximum 5 User App clients will connect simultaneously to the Server App.

Development for testing the functionality

* + - 1. UI does not have to be great, but should be clear and easy to decipher the functionality.
      2. Only minimal functionality is required such that the functionality can be easily demonstrated.

Web Page

A dummy web page to trigger the User App authentication when some dummy items on the web page is selected and purchased. There is no need to add any payment method or connect to payment gateway.

Q: A web application with a dummy page to be created? If yes where will the web app be running on?

Ans: I will create a domain to host it. I have Not finalised the domain name yet. I will provide you access to the domain once created.

Q: When items on web page are selected and purchased it will trigger Android, iOS or Windows client app to authenticate?

Ans: One HUMAN user can have multiple devices associated with him/her/them. All of these devices will have same username but different Device Id. The Device Id is unique to Device and username unique to HUMAN. So, when the HUMAN through username buys something from the web page, it will trigger the server App (see the first line of Apps requirements to identify this) which then will trigger all the associated devices to that username for the HUMAN. The HUMAN can then choose any ONE device and its User App (client App - see the first line of Apps requirements to identify this) to complete the buying process on the website.

This is to demonstrate how the User App provides an authentication method – a factor of authentication in the MFA.

Q: Which Android, iOS or Windows client app to authenticate? Need to provide a list of clients to select?

Ans: See above.