Technical specificiation

All requests must be done over HTTPS. It increases the difficulty for a potential attacker greatly.

Generating shared secret

Using a “cryptographically secure” random number generator (RNGCryptoServiceProvider in ASP.net), generate 32 characters (A-Z, a-z, 0-9). This is the same length as a GUID, and the risk of randomly generating two identical GUIDs is sufficiently close to 0 to be disregarded.

This should be hashed before storing, same as with any password. I propose using SHA256 or better yet SHA512.

Encrypting the shared secret for local storage in app

Use AES-256 encryption. Use the users password as key. The encrypted data must be stored in a way that one can tell what user the encrypted data belongs to. How exactly this is done does not matter, as long as it is internally consistent; stored data should not be decipherable, and decrypted stored data should be identical to the initial data.

I suggest using using the bouncycastle library for android. Perhaps use RNCryptor for swift. Of course, any library that implements AES-256 will suffice.

Timeouts

The “holding screens” should wait for one minute before returning. This goes for the backend too; an authentication attempt sent after a minute is considered old/untimely.