Biometrics Authentications Approach

Nhan Cao – nhancv92@gmail.com

***Abstract*—Modern security is a battle between high security and low friction. Users, faced with the challenge of remembering a number of different passwords, want to use the simplest password or passcode possible. This makes it easier to remember, and makes access to the app or site quicker, but is fatal for online security. One way to combine both performance and security is to take advantage of fingerprint sensors that are now prevalent on high-end smartphones and other mobile devices.**

I. Introduction

THIS document is a biometrics authentication approach which especially for fingerprint on Android or touch id on iOs. Fingerprints have a number of benefits over passwords:

--Performance: Modern capacitive touch fingerprint sensors (such as the Touch ID [1] sensor on the latest Apple iPhones and sensors on recent Android devices) recognize the fingerprint and unlock in less than a second, quicker than inputting an extensive password.

--Security: Fingerprints are unique (even among identical twins), impossible to guess, and difficult to fake without significant effort. Modern fingerprint authentication uses the fingerprint to create an encrypted key, which is sent for server authentication.

--Permanence: Complex passwords are hard to remember, leading to the majority of people reusing passwords on multiple sites. As a physical feature, fingerprints are unforgettable.

II. HOW FINGERPRINT AUTHENTICATION WORKS

When a user elects to use fingerprint authentication, the library will generate a **KEY**. **KEY** will be stored on Keystore of the device and server also.

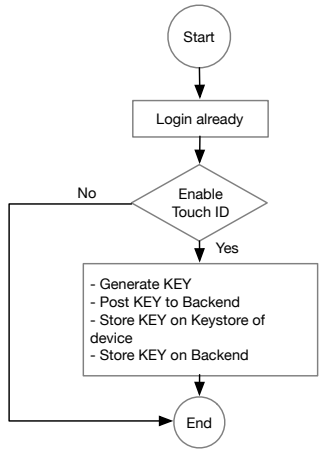

Fig. 1. Enable Fingerprint diagram.

When a user tries to authenticate with a valid fingerprint and get a success; Touch ID retrieves the **KEY** from the Keystore. A JSON Web Token (JWT [2]) will be created then signed with the **KEY**. The app sends this signed JWT to backend. Backend will verify signature with **KEY** and returns an id_token.

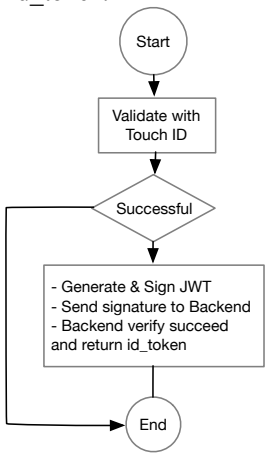

Fig. 2. Fingerprint Authentication diagram.

A. Approach 1: Verify signature with HMACSHA256

verify(signature, verificationKey)

In systems using HMAC signatures, verificationKey will be the server's secret signing key (since HMAC uses the same key for signing and verifying). Function above can be write as below:

verify(signature, serverHMACSecretKey)

With HMAC, KEY is unique and using for sign JWT and verify on backend also.

B. Approach 2: Verify signature with RSASHA256

verify(signature, verificationKey)

In systems using an asymmetric algorithm, verificationKey will be the public key against which the token should be verified:

verify(signature, serverRSAPublicKey)

With RSA, KEY are private key and public key. Private key store on mobile device and Public key store on server. This approach is more secure than HMAC because keys are

separate and verification need both of them.

III. Protect Data With Fingerprint

As jailbroken device can access your iOS Keychain/ Android shared preference and key store in plain text, it is necessary to add another layer of protection so even jailbreaking will not leak your data:

--For iOS it is implemented though Access Control [3]. Everytime when app wants to access the protected keychain item, a prompt by iOS will show up. Only when authentication success will the app get the keychain item.

--For Android it is implemented though FingerprintManager [4] + Keystore [5]. Keystore has a function called setUserAuthenticationRequired which makes Keystore requires user authentication before getting value.

IV. Conclusion

With above approachs, can implement authentication system for login and purchase also without using 3rd party library. The level authentication security decide by hardware and accepted by user.

References

[1] Wikipedia, 2018 Available: https://en.wikipedia.org/wiki/Touch_ID

[2] JSON Web Tokens, Available: https://jwt.io/

[3] Apple Developer, 2017 Available: https://developer.apple.com/documentation/security/secaccesscontrol

[4] Android Developer, Available: https://developer.android.com/reference/android/hardware/fingerprint/FingerprintManager.html

[5] Android Developer, Available: https://developer.android.com/reference/java/security/KeyStore.html