Encrypit

Calculator Password Vault

Final Writeup

Steven Guarino

Abstract

This application is meant to provide a user with a hidden encryptable list. Hidden behind a functional calculator UI that would only reveal the vault portion of the application if the correct expression was entered into the calculator's display. On hitting the "=" key within the calculator the expression the user has input is checked against a hidden code that if correct will load the second activity of the application which is the encryption portion. If the calculator's expression field does not match the code the expression is resolved and displayed. Within the encryption application the user defines a key alias which stores the users key used to encrypt their password list data. The user needs to input the same key used for encryption in order to decrypt.

Introduction

This application provides users with an encryptable password list while providing a hidden location for its storage. This can provide a user with an added layer of security as someone with malicious intent may be browsing the phone in search of the users notes application, where some of the more cavaliere users store their passwords. The application adds an additional layer of security by erasing the key alias a user has input after encryption has taken place. This requires the user to re-enter the same key alias used during encryption for proper decryption. This key storage system is accomplished through Android's Keystore library.

Methodology

This application was built using the Android SDK, Android's Cryptography library and keystore system. The Keystore library provides a secure container for storing cryptographic keys. These keys can be used for cryptographic purposes while remaining non-exportable and have the ability to be password protected. This keystore system is what provides this application with a secure system for storing the cryptographic keys used in encryption and decryption of the password list. Below is an example of key generation and storage under the name of the user defined key alias "aliasHolder." After which the cipher is initialized with said key, encryption mode, block cipher mode, and padding specifications.

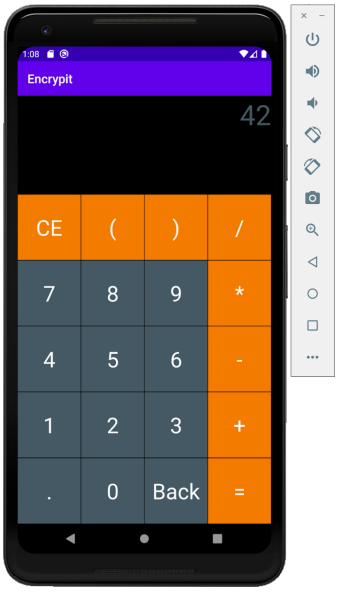
```
// Instance of Androids KeyGenerator
val keyGenerator = KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_AES,  provider: "AndroidKeyStore")

// build KeyGenParameterSpec with parameters of key, storing in keystore using keystore alias
val keyGenParameterSpec = KeyGenParameterSpec.Builder(
    // aliasHolder is user defined key alias
    aliasHolder,  purposes: KeyProperties.PURPOSE_ENCRYPT or KeyProperties.PURPOSE_DECRYPT)
    .setBlockModes(KeyProperties.BLOCK_MODE_GCM)
    .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_NONE)
    .build()

keyGenerator.init(keyGenParameterSpec)
val secretKey = keyGenerator.generateKey()

val cipher = Cipher.getInstance( transformation: "AES/GCM/NoPadding")
cipher.init(Cipher.ENCRYPT_MODE, secretKey)
```

Decryption is much the same as initialization except it needs the same IV passed from the cipher from encryption passed to the decryption cipher.

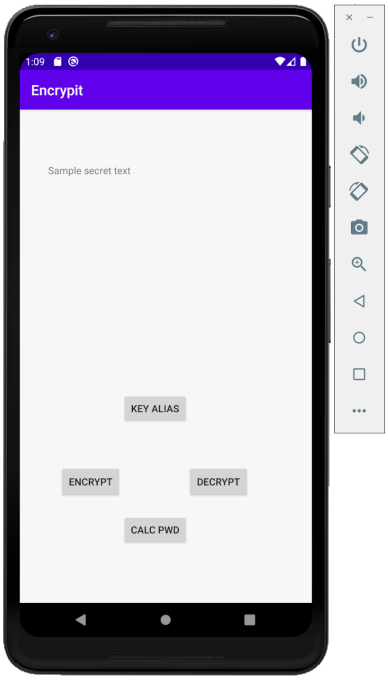
```
// Creating keystore instance
val keyStore = KeyStore.getInstance( type: "AndroidKeyStore")
keyStore.load( param: null)

// Gets our secret key from keystore with alias
val secretKeyEntry = keyStore
    .getEntry(aliasHolder,  protParam: null) as KeyStore.SecretKeyEntry

val decrsecretKey: SecretKey = secretKeyEntry.secretKey
val cipher =
    Cipher.getInstance( transformation: "AES/GCM/NoPadding")
// Passing encryption IV to decryption cipher
val spec = GCMParameterSpec( tLen: 128, iv)
cipher.init(Cipher.DECRYPT_MODE, decrsecretKey, spec)
```

Application

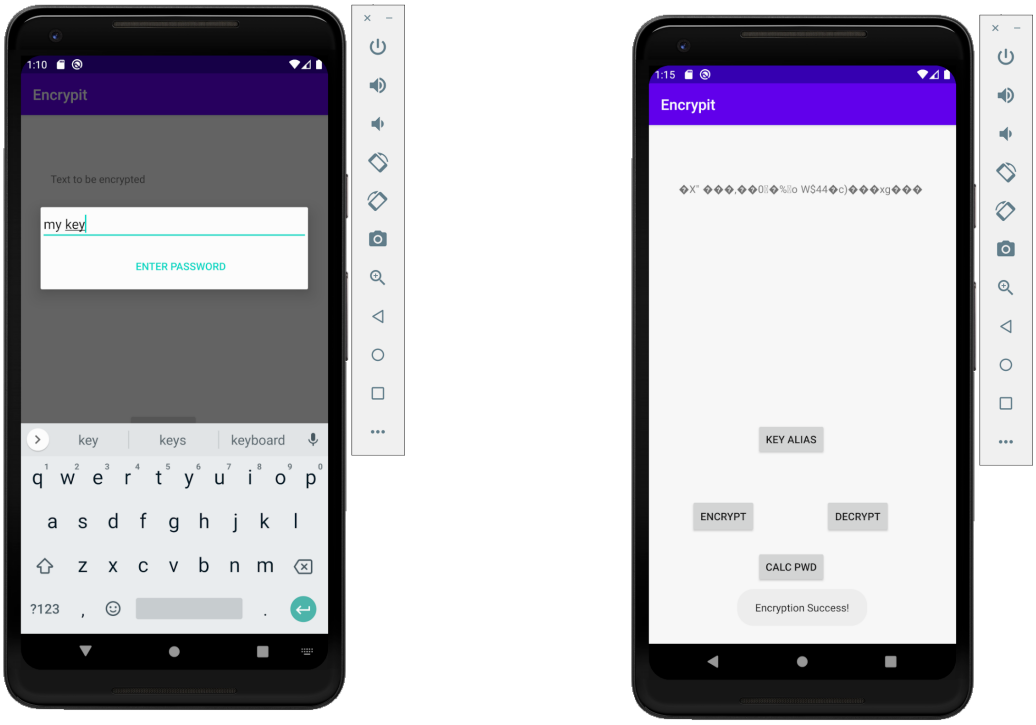
The first screen seen is the calculator.



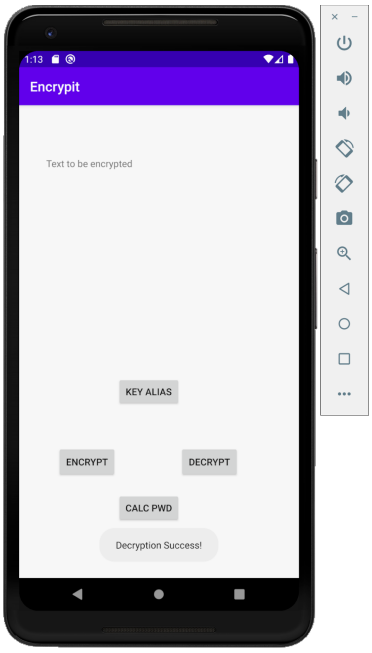
Given the correct expression Encrypit is shown.



The user is able to enter a key alias, and encrypt and decrypt the contents.



Once given the correct key alias, the text becomes decrypted.



Conclusion

The limitations of the application are most notable when restarting the app, when the application is closed and opened the reference to the encryption cipher's IV is lost. This can be circumvented through the use of androids shared preferences, allowing persistent storage of variables and files. This could also be extended to the calculator password button which receives an updated calculator passcode that the user can define from within the encryption application. The application works as expected, providing the user with a calculator UI with and a hidden application that provides the user with encryption and decryption capabilities.

References

https://developer.android.com/guide/topics/security/cryptography

https://developer.android.com/training/articles/keystore

https://developer.android.com/reference/javax/crypto/Cipher