Encryption and Decryption of Messages on Android using NFC Tags

Andrei DRAGAN
IT&C Security Master
Department of Economic Informatics and Cybernetics
The Bucharest University of Economic Studies
ROMANIA
adragan@hollowdesign.ro

Abstract: Working with NFC technology and mobile devices brings a faster and more secure way of doing things like giving out contact information, automating certain tasks or transmitting data, as easy as touching the two. What this article focuses on is the use of NFC tags on the Android platform to store random generated keys and automate the encryption and decryption of messages.

Key-Words: Near Field Communication, Encryption, Android Mobile Devices, Software Security

1. Introduction

In recent years automatic identification procedures (Auto-ID) have become very popular in many service industries, purchasing and distribution logistics, industry, manufacturing companies and material flow systems. Automatic identification procedures exist to provide information about people, animals, goods and products in transit [1].
NFC or Near Field Communication which is a radio technology standard which works over limited ranges and powers smartphones as well as other devices to communicate with each other when they are brought together to a close proximity, allowing the transfer of small amounts of data between the two.
We live in a world where privacy invasion is a constant growing threat. Encrypted data transmission is a must in fields like healthcare, military applications, financial transactions or classified corporate data exchange to ensure quality standards for restricted access information. These are just a few examples that require increased privacy and it can be achieved in two ways:

- securing the communications channel used between the parties involved;
- hiding the message from unwanted parties by encrypting the information.

The need for privacy has always been a main concern when transmitting information over various types of mobile messaging platforms like Facebook Messenger, WhatsApp, Viber and Google Hangouts or even over simple SMS.
What this article focuses on, is an application which combines NFC technology with message encryption in order to provide the users with a safe way of transmitting messages with reduced risk of sensitive data being compromised. The NFC tag is used to store a key string generated with the application and it will be given to another user to import the key into their application. By manually giving the tag to another person to import the key it prevents possible attempts to intercept the message or duplicate the key. The encrypted message can then be shared through various applications and sent to the intended recipient or copied and used where necessary. The user can also decrypt messages received from other recipients provided they all use the same generated key previously shared via NFC tags and an agreed password.

2. Proposed Model

The solution discussed in this article is based on three main components:

- an NFC Forum type 4 tag;
- an NFC enabled Android mobile device;
- the Android cryptography class.

ISSN 2067 – 4074

Since most of the mobile devices released on the market today come equipped with an NFC interface and the fact that Android, Windows Phone and iOS come with a cryptography architecture already built in, it was an affordable and interesting idea to implement of combining the two ideas to produce a method to encrypt messages and a safe way to share the key with trusted people.
This model however concentrates on the use of Android as the operating system and the cryptography class that's already available.

2.1 NFC Forum Type 4 tag

There are 5 types of NFC tags; this article proposes the use of commonly found tags which are Type 4 with the following specifications:

- based on NXP DESFire tag (ISO-14443A) specification;
- configured by factory to be read-only, or read/write capable;
- contains 2, 4 or 8KB of memory;
- three communication speeds: 106, 212, or 424Kbps;
- anti-collision support.

For this, a writable NFC tag created by Samsung with a size of 884 bytes was sufficient as it support the IsoDep, NfcA and Ndef technologies. The application developed takes advantage of the Ndef (NFC Data Exchange Format) technology to write on the tag the key that will be generated by the application.
All the types of tags can be seen in the picture below.

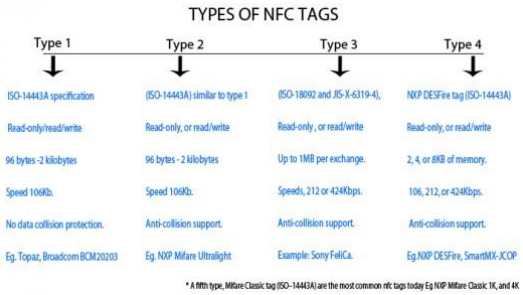

Figure 1. NFC Forum Type tags

2.2 NFC enabled Android mobile device

As a short-range wireless communication technology that potentially facilitates the mobile phone usage of billions of people over the world, NFC offers an enormous number of use cases – including credit cards, debit cards, loyalty cards, car keys, and access keys to hotels, offices, and houses – and has the potential eventually to integrate all such materials into one single mobile phone [2].
There are two main modes in which NFC technology can be used, active and passive.
In Active mode NFC interfaces alternately emit magnetic fields for data transmission, one of the NFC devices activates the transmitter interface and thus sets itself as the initiator, while the other device activates the receiving interface and sets itself as the receiver. When the receiver needs to respond it will be converted into a transmitter while the other device that was first an initiator will now become the receiver of the data.

In Passive mode the initiator also emits magnetic fields towards the NFC target and expects a response from it, however, compared to the active mode once the whole target is read and data transmitted there is no other response that can be expected.
The NFC interface that is the target is also able to establish, in addition to other NFC interfaces, the communication to compatible passive transponders (e.g. according to ISO/IEC 14443) that the NFC target supplies with power and that, via load modulation, can transmit data to the NFC interface. […] As the NFC interface in this case behaves similar to an RFID reader, this option is also called 'reader mode' or 'reader-emulation mode' [3].
A figure can be seen below on how an NFC device operates on both modes.

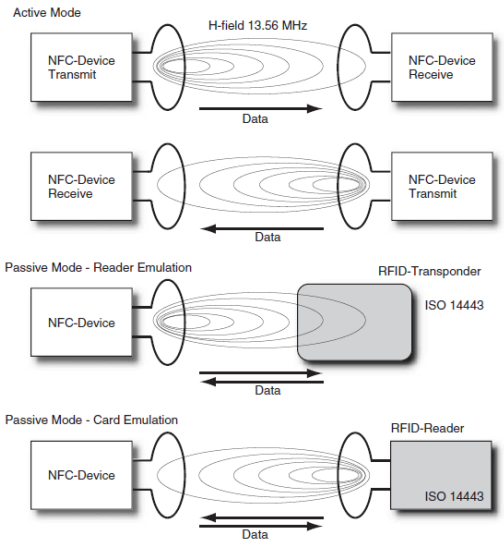

Figure 2. NFC passive and active modes

2.3 Android Cryptography class

The Android `javax.crypto` package provides the classes and interfaces for cryptographic applications implementing algorithms for encryption, decryption or key agreement [4].
The authentication used in the solution is based on the MAC (Message Authentication Code) with a HMAC (Hash MAC) with a SHA-1 hash function.
The solution proposed implements a cypher, an algorithm for performing encryption or decryption, based on the AES (Advanced Encryption Standard) with an ECB (Electronic Code Book) mode and a PKS5 style padding scheme.

There are several Android application on the Google Play Store that perform encryption and decryption of data based on AES or DES algorithms but none of them allows the generation of a random 128, 192 or 256 key and write it on an NFC tag. The generated key in combination with a chosen password offers extra protection to the encrypted text making it a lot harder for an attacker to decrypt the text without the correct credentials.

3. Model Solution

The application was developed for the Android ecosystem and as such, a capable mobile device with NFC technology implemented is required. The minimum SDK that the application runs on is version 14 which is Ice Cream Sandwich 4.0.

```
    defaultConfig {
        applicationId
"andreidragan.crypto"
        minSdkVersion 14
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
    }
```

This specific SDK version was chosen because most of the devices today are running this update or a higher one as it can be seen from the below distribution:

Table 1. Android distribution (04.05.2015)

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.3%
4.1.x	Jelly Bean	16	15.6%
4.2.x		17	18.1%
4.3		18	5.5%
4.4	KitKat	19	39.8%
5.0	Lollipop	21	9.0%
5.1		22	0.7%

Android Studio 1.2.0 was the software used in the development of the application which requires the following permissions to function correctly:

```
<uses-permission
android:name="android.permission.NFC"
/>
<uses-permission
android:name="android.permission.READ_E
XTERNAL_STORAGE" />
<uses-permission
android:name="android.permission.WRITE_
EXTERNAL_STORAGE" />
<uses-feature
  android:name="android.hardware.nfc"
    android:required="true" />
```

Permissions are required in order to make use of certain Android API functionalities, in this case we ask to READ and WRITE on from external disk (which, depending on the device can be a micro SD card or in our case use the SharedPreferences

ISSN 2067 – 4074

class) and make use of the NFC capabilities.
If for some reason a user will try to install the APK (application package) on a phone that does not have NFC capabilities, the application will exit and present the user with an appropriate message.
The mobile device used to test the developed application was an OnePlus ONE (A0001) running Android API 21 (Lollipop) version 5.0.2 and in the following picture can be seen the main activity (the first screen that the user is presented when entering the application).

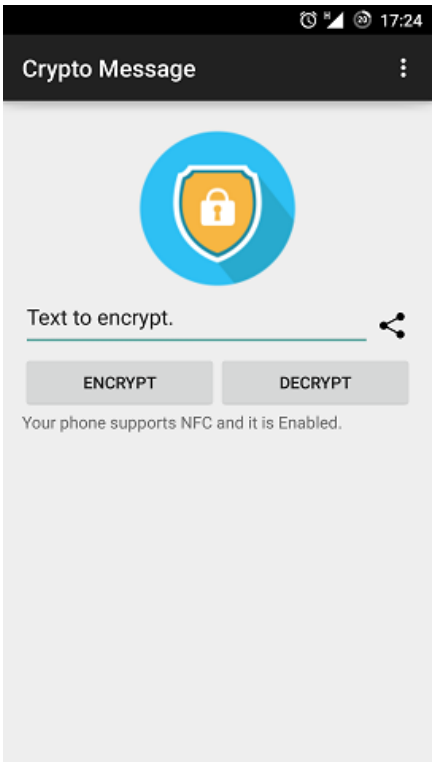

Figure 3. Main activity

The elements that can be found in the picture are as follows:

- the status bar with alarm option, GSM connectivity and data type, battery meter and current time;
- the action bar with the application name and the settings menu which is activated by the three vertical points;
- the application logo;
- a text field element where the user can insert the message that needs to be encrypted or decrypted;
- a share button shown as an image which allows the user to share the encrypted/decrypted text through various applications;
- an encrypt button which encrypts the text written in the previous field;
- a decrypt button which decrypts the text written in the previous field;
- a text view element which informs the user if NFC is enabled or not.

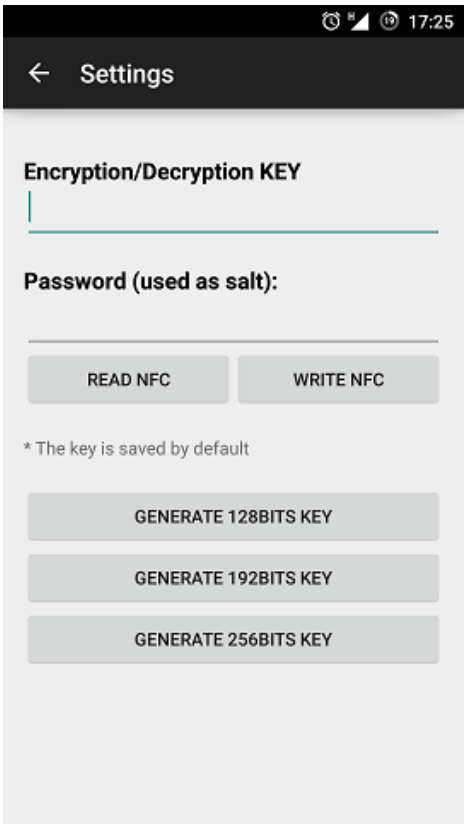

Figure 4. The settings menu

If the user opens the application with the NFC option disabled, the application will minimize and open the phone's settings menu in the Wireless & networks area and through a toast message it will inform that the phone supports NFC and it should be enabled before using the application. After enabling NFC the user can return to the application by pressing the back button on the device.
Activating the settings menu allows the user to choose make changes to the following options:
The Encryption/Decryption Key text field shows the user if there is already a random generated key, if there isn't one, the text field is empty.
Password is the field where a string can be chosen in order to use as salt in the encryption algorithm to encrypt and

decrypt data. This will be given to the users of the application who are intended receivers or transmitters of messages.
Write NFC allows the user to write the contents of the Encryption/Decryption Key to an NFC tag in Ndef format by showing a dialog which informs the user on the steps required to perform the action as it can be seen in the picture below:

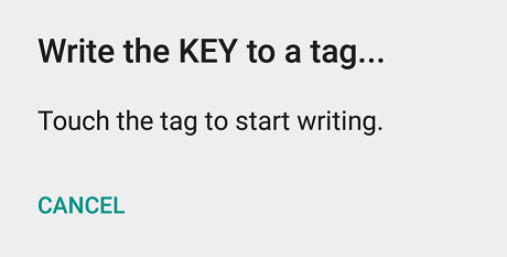

Figure 5. Write to NFC dialog

If there is no key generated, then the user is informed that first he needs to generate a key before writing anything on a tag.
Read NFC allows the user to read the written key from an NFC tag by popping up a dialog which informs the user of what steps need to be taken in order to read the tag. Below is an example of how it works:

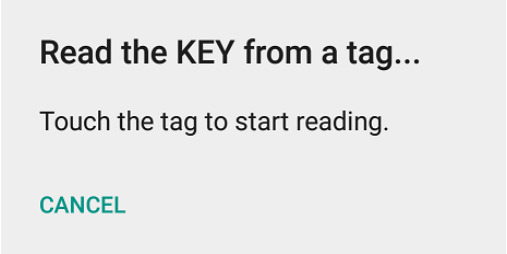

Figure 6. Read from NFC dialog

An NFC tag needs to be brought near the mobile device and the generated key that was written previously on the card will be displayed on the key text field.
The next three buttons call the `generateKey` method which based on the choice made, it will generate a 128, 192 or a 256 bit key, which is a combination of lower and uppercase alphabet characters and numbers. The next picture shows a 128 bit generated key as well as a chosen password that will be used in the encryption and decryption process:

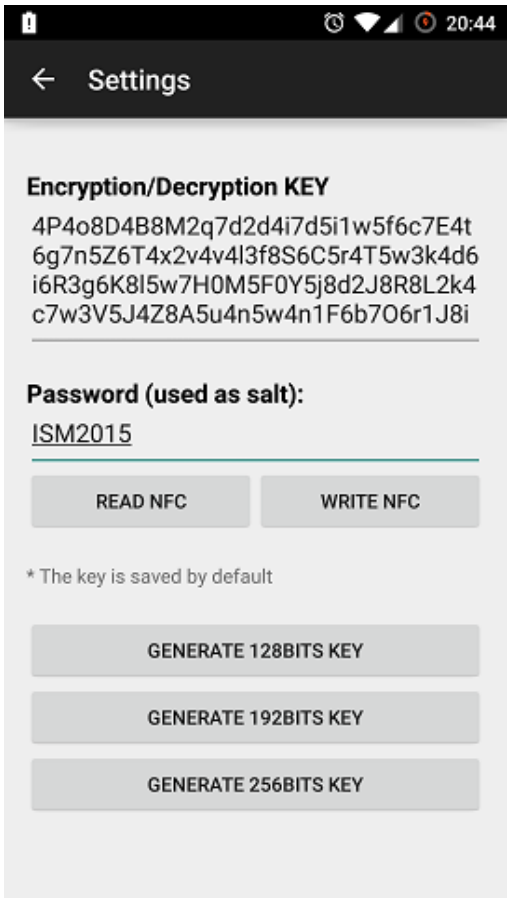

Figure 7. Generated key and password

All the settings are saved automatically with the help of the shared preferences class when the user presses the return (back) button (hardware or software) on the device. Pressing the back button at the top of the screen, allows the user to discard all changes done in the settings menu.
With the key generated and written on the tag and a common password chosen, the application is ready to encrypt any text.
Next we can see a sample text in clear form as well as encrypted with a 128 bit key randomly generated and the ISM2015 password:

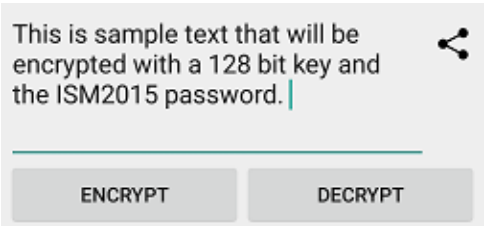

Figure 8. Text in clear

ISSN 2067 – 4074

Pressing the encrypt button calls the `encryptText` method which using an AES/ECB/PKCS5padding cipher and the salt and key generated will encrypt the text.

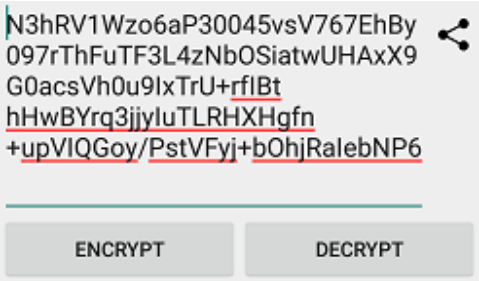

Figure 9. Encrypted text

With the text encrypted, it is now ready to be shared with the intended recipient provided that they were given the tag that was written with the encryption key and the chosen password. Pressing the share image button will give the user the option to share the text on any preferred application.

5. Conclusion

This document provided an overview of the solution proposed on encrypting and decrypting messages using NFC tags and AES encryption with a random generated key and a commonly chosen password.
Some people might find having to personally trade the NFC tag to someone a bit of a problem because they need to physically move to a location and personally hand the written tag to the intended recipient, however when sensitive information has to be secure, there should be no barrier on the means to achieve this goal.
However there is a safety concern that needs to be addressed. As with any RFID enabled device it is possible to copy the contents of the tag and replicate it by bringing it in close proximity with another NFC enabled device, but there is a smaller chance of this happening as the hijacker needs to be less than four centimeters distance from the tag itself, than having the key transmitted over the internet and the third party simply writing the said key themselves on a new tag.
In the future the solution can be brought to other operating systems like Windows Phone 8 or iOS 6 as both support NFC technology and have a similar cryptography class.

Acknowledgement

Parts of this paper were presented at The 8th International Conference on Security for Information Technology and Communications (SECITC 2015), Bucharest, Romania, 11-12 June 2015.

References

[1] John Wiley & Sons Ltd, *RFID Handbook 3rd Edition*, Wiley, 2010
[2] Vedat Coskun, Kerem Ok, Busra Ozdenici, Professional NFC Application Development, Wrox, 2013
[3] John Wiley & Sons Ltd, *RFID Handbook 3rd Edition*, Wiley, 2010, pg. 59
[4] developer.android.com, javax.crypto | Android Developer, http://developer.android.com/reference/javax/crypto/package-summary.html, retrieved on 2015-05-06