ST25DV-I2C cryptographic demonstration

Introduction

This document shows how to run the "ST25DV-I2C Crypto Demo", using the ST25DV-I2C fast transfer mode (FTM) to establish a secure transfer channel (STC) over NFC between an STM32 microcontroller and an Android™ smartphone or iPhone (7 and up, with iOS13 or later version).

The ST25DV-I2C is a dynamic NFC Tag IC able to communicate with NFC readers and smartphones, and also with a microcontroller through an I^2C interface. The FTM feature speeds up the communication between these two interfaces.

This demonstration establishes an STC by using cryptography to perform a mutual authentication and to encrypt the communications over NFC. This STC is used during the demonstration to securely:

- Send and retrieve data
- Set the device settings
- Upload new firmware

Only the granted user / smartphone is able to communicate with the STM32 device to perform these operations.

The STC over NFC has applications in different sectors (such as industrial, home appliance and consumer) where the control of a device is restricted to authorized users, and when the personal data must be protected.

The following packages are available on www.st.com:

- STSW-ST25DV005 firmware
- STSW-ST25003 Android™ application
- STSW-ST25IOS003 iOS™ application

1 General information

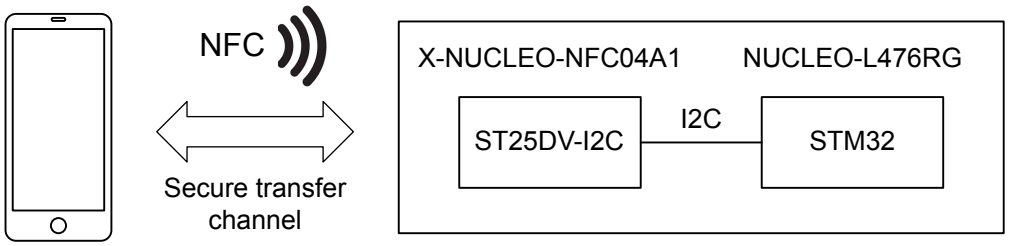
The application described in this document runs on the STM32L476 Arm®-based devices.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

1.1 Purpose and scope

The "ST25DV-I2C Crypto Demo" runs on the NUCLEO-L476RG board plus a X-NUCLEO-NFC04A1 shield, featuring a ST25DV-I2C tag connected to a STM32L476 device through the I²C bus. This kit represents an IOT sensor device, which is controlled by an Android™ smartphone or by an iPhone (with iOS13 or later version) through the NFC.

Figure 1. Connection scheme of the "ST25DV-I2C Crypto Demo"



When a communication between the device and a smartphone starts, a mutual authentication takes place. It ensures that:

- The user of the smartphone has a permission to communicate with the device
- The device is not counterfeited

Once the mutual authentication has taken place, all the communications between the microcontroller and the smartphone are encrypted, so the user can configure the product or retrieve data securely. Anyone who spies on the data exchanged on the NFC is unable to interpret them. The key used to encrypt the communication changes each time a mutual authentication is done, this action prevents someone from recording the encrypted content and replaying it.

In this demonstration, the first user of the device becomes the administrator so the device refuses requests from other users.

More details on the cryptographic processing used in the demonstration are provided in Section 2 Security processes.

Additional details on the implementation are provided in the AN5453 *"ST25DV-I2C crypto demonstration"*.

1.2 Acronyms

Table 1. **Acronyms**

Acronyms	Meaning
NFC	Near field communication
AES	Advanced encryption standard
ECC	Elliptic curve cryptography
FTM	Fast transfer mode
STC	Secure transfer channel
GCM	Galois/counter mode
GMAC	Galois message authentication code

1.3 Hardware equipment

The following hardware is needed for this demonstration:

- NUCLEO-L476RG board plus X-NUCLEO-NFC04A1 shield
- An iPhone (with iOS™ 13 or later) or an Android™ smartphone with at least the version 6.0 (Android™ Marshmallow)

1.4 Installation

This demonstration requires to download the firmware (STSW-ST25DV005) for the NUCLEO-L476RG board and the Android™ executable (APK, STSW-ST25003) the iOS™ application (STSW-ST25IOS003) to be used on a smartphone.

1.4.1 NUCLEO-L476RG and X-NUCLEO-NFC04A1 setup

The "ST25DV-I2C Crypto Demo" binary (“*SBSFU_UserApp.bin”*, available in STSW-ST25DV005 package) must be uploaded on the NUCLEO-L476RG board.

To program the NUCLEO-L476RG board, go through the following steps:

1. Install the ST-LINK USB driver, available on www.st.com.
2. Connect the NUCLEO-L476RG board to a PC with the USB-mini port.
3. The NUCLEO-L476RG board icon must appear in the PC directory.
4. Drag-and-drop the "SBSFU_UserApp.bin" binary to the NUCLEO-L476RG board icon.
5. Restart the NUCLEO-L476RG board by pressing the reset (black button).

1.4.2 Android™ APK application installation

The "ST25DV-I2C Crypto Demo" Android application is available at Google Play™ store.

The STSW-ST25DV005 package contains two firmwares in the UserApp Binary directory (UserApp.sfb and UserApp_NewPicture.sfb) to use for the secure firmware upgrade demonstration. These files have to be downloaded on the Android™ smartphone.

1.4.3 iOS™ application installation

This application is not available on Apple® store so it must be installed manually.

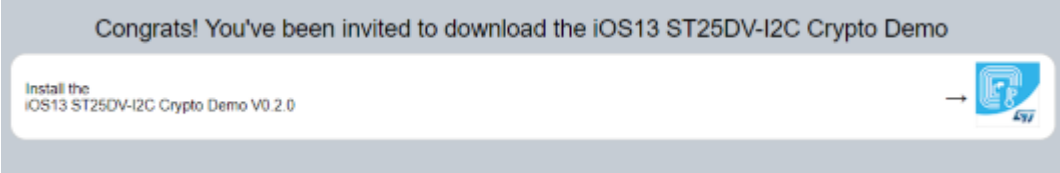
The user can download the application by the OTA (over the air) programming, a method for wireless distribution of an application and/or its updates to end-users.

On approval, application is automatically installed or it is updated if it is already installed.

Perform the following steps to install the application:

1. Use Safari® browser on your iPhone
2. Enter URL: *http://myst25.com/iOSST25DVCryptoDemo/*
3. Click on the blue icon on the right (see Figure 2)
4. Check iOS™ installation on your iPhone.

Figure 2. **Blue icon view**



When user opens an enterprise application (OTA installed), user is identified by one 'Untrusted Enterprise Developer' notification. In such case, user needs the 'Trust Developer' certificate allowing application to be installed.

This can be done by tracing the menu as follows: Settings→General→Device Management→Enterprise Apps→Trust Developer.

1.4.4 How to set the "Authorized User"

As described in Section 2.1 Public key exchange, the first smartphone establishing a secure session with the firmware is considered as the owner of the device. This means that only this smartphone is able to establish a secure session from now on, and all connections from other smartphone are rejected.

When no "Authorized User" has been set yet, the X-NUCLEO-NFC04A1 LED1 (green) is OFF.

After the start of the demonstration and set up of a secure channel (as described in section Section 3.2.1 Secure transfer channel setup), the STM32L476RG microcontroller saves the connection data (smartphone "Login" and "Public key") and only accepts connections with the same smartphone credentials.

Note: As these data are saved in the Flash memory of the STM32, the firmware restores them after a reset.

Once an "Authorized User" is set, the X-NUCLEO-NFC04A1 LED1 (green) is switched ON.

Any other user / smartphone trying to connect to this NUCLEO-L476RG board is rejected, and the X-NUCLEO-NFC04A1 LED3 (yellow) is switched ON..

It is possible to set a new "Authorized User" by pushing the user button (blue) of the NUCLEO-L476RG board (any previously stored "Authorized User" is erased by the firmware).

1.5 License scheme

The Android™ and iOS™ application and the associated firmware are provided under the SLA0052 license agreement, available on www.st.com.

The software components provided in this package come with different license schemes, as shown in Table 2.

Table 2. ST25DV-I2C cryptographic firmware - License scheme

Component	Copyright	License
Project application	STMicroelectronics	ST proprietary
LibNDEF		ST proprietary
Menu demo		ST proprietary
Board support package (BSP)		ST proprietary
LibJPEG		ST Liberty SW License Agreement V2
HAL STM32L4		Open source BSD
STM32_Cryptographic		Image V2 (object release only)
STM32_Secure_Engine		Ultimate Liberty (source and object release)
Cortex®-M CMSIS	Arm®	Open source BSD

2 Security processes

This section describes the security processes used to perform a mutual authentication and to establish a secure transfer channel (SFC) where all the communications are encrypted.

2.1 Public key exchange

The public keys exchange is done in two steps:

- The smartphone sends its "Login" and ECC "Public key".
- The firmware sends its ECC "Public key".

If the firmware has no registered "Login" yet, it saves the "Login" and the "Public key" in the static memory and considers this user to be the "Authorized User" of the product. It means that the firmware only accepts requests from this smartphone.

When the smartphone receives the firmware "Public key", it checks that this key is signed by a manufacturer key. This verification ensures that the product (represented by the NUCLEO-L476RG and X-CUBE-NFC04A1 kit here) is not counterfeited.

2.2 Definition of a "Shared Secret"

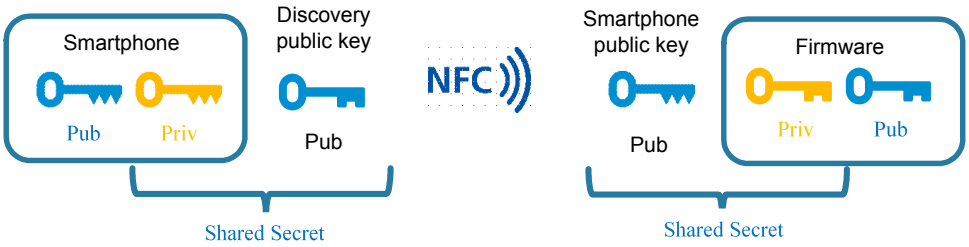
To establish an encrypted channel, the smartphone and the firmware have to agree on a symmetric key used to encrypt all the communications between the two devices. This key cannot be exchanged over NFC because someone may spy all the data exchanged and get the key.

Elliptic curve Diffie–Hellman (ECDH) is a “key agreement protocol” used to establish a “Shared Secret” over an insecure channel. Section 2.3 Derivation of a public key describes how this "Shared Secret" is used to define a symmetric key to encrypt all the communications of this session.

Both two communicating devices must have an ECC key pair. They exchange their public keys (the private key remains secret and is not shared). Each device uses ECDH scheme to combine its own private key with the public key of the peer device. Thanks to ECC, these two operations bring to the exact same result, which is called “Shared Secret” (see Figure 3).

Someone who has spied the communication has seen the public keys exchanged but this is not sufficient to find the "Shared Secret".

Figure 3. Elliptic curve Diffie-Hellman over NFC



The two devices have been able to define a "Shared Secret" that nobody else can find. Only the ones knowing the private keys can get the "Shared Secret".

2.3 Derivation of a public key

The "Shared Secret" can be used to encrypt the communications between the two devices but it has a weakness: the ECC key pairs of the smartphone and firmware do not change, so the "Shared Secret" is always the same. Someone can record the data exchanged over NFC and re-execute them. This is called "replay attack".

To avoid this problem, a key is derived from the "Shared Secret" plus a random number (changing every time). The key obtained is called “AES Session key” and it is used to encrypt all the exchanges between the two devices. The random number changes every time so the session key is different.

By convention, the random number used for key derivation is chosen by the firmware and shared (not encrypted with the Android™ or iOS™ phone).

In this demonstration, an AES-256-GCM encryption is used. GCM (Galois counter mode) permits the authentication of the encrypted messages received (GMAC). Each encrypted message is authenticated so the receiver detects if the received encrypted message has been modified.

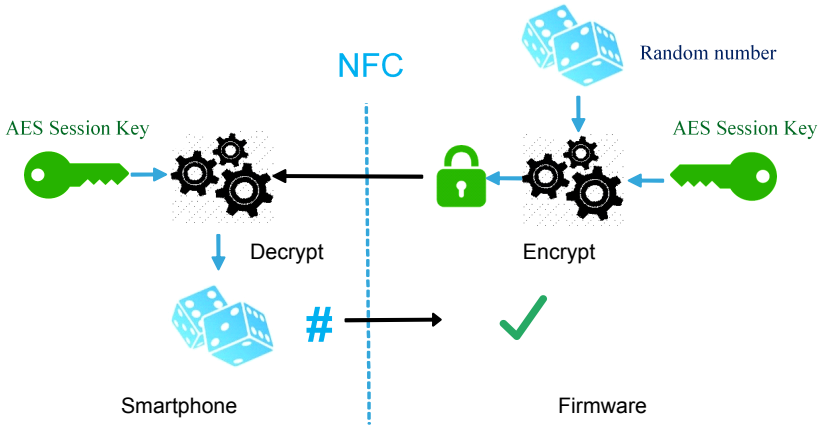
2.4 Authentication of the smartphone

When the communication between the smartphone and the firmware starts, the smartphone sends a “Login” to the firmware. This "Login" corresponds to the "Login" received by the firmware during the keys exchange phase when the product has been used for the very first time. The NUCLEO-L476RG board has saved this "Login name" and the corresponding "Public key" in its static memory.

The firmware challenges the smartphone to check if it really knows the "Private key" corresponding to this "Public key":

1. The firmware generates a random number, encrypts it with the AES session key and sends it to the smartphone.
2. If the smartphone owns the "Private key" corresponding to the "Login name", it computes the "AES Session key" and decrypts the message received.
3. The smartphone sends a SHA256 hash of the random number in order to prove that it has been able to decrypt the challenge.
4. The firmware also computes the SHA256 hash and then knows if the answer is correct.

Figure 4. Smartphone authentication over NFC



This authentication protects the device from someone trying to usurp the "Login" of a valid user. A hacker may know the "Login" and the associated "Public key" (since they are exchanged not encrypted over NFC) but does not know the "Private key" so the "Shared Secret" or the "AES Session key" cannot be computed.

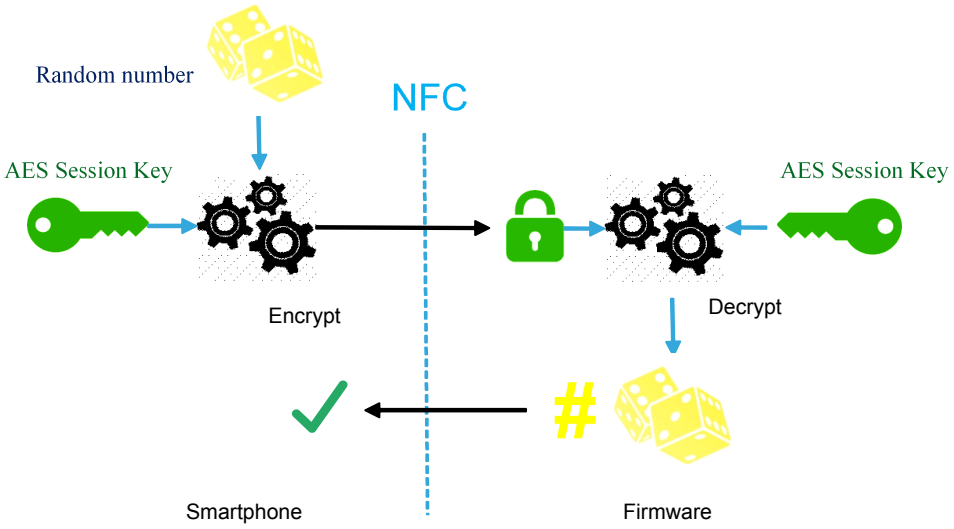
2.5 Authentication of the connected device

The smartphone performs an authentication of the firmware. This is done to be sure that the product is genuine and corresponds to the "Public key" that has been saved in the smartphone during the key exchange phase.

The procedure is the same but in the opposite direction: now the smartphone generates a challenge, encrypts it with the "AES Session key" and sends it to the firmware.

The firmware decrypts it and sends a SHA256 hash to prove that the decryption is correct.

Figure 5. **Firmware authentication over NFC**



This authentication protects from counterfeited products containing a valid "Public key" taken on a valid product. However it does not contain the "Secret Key" that is stored in the product and that is not readable. The counterfeited product is not able to compute the "Shared Secret" nor the "AES Session key", so it fails this authentication phase.

2.6 Encrypted data transfer

Once the mutual authentication has been run, all the imminent communications over NFC are encrypted using the current AES session key, which means:

- Someone spying the NFC communication is not able to decrypt the transmitted data (because the current "AES Session key" is unknown).
- A message not encrypted with the current "AES Session key" is rejected
- A valid message (encrypted with the current "AES Session key") maliciously modified is rejected (thanks to the message authentication).

The AES encryption is performed by using the GCM.

This encryption method requires to transmit additional metadata along with the encrypted data:

1. An initialization vector (12-bytes) required to initialize the decryption process. This initialisation vector changes for every new encrypted message.
2. A GMAC of 16 bytes used to ensure the message integrity and source.

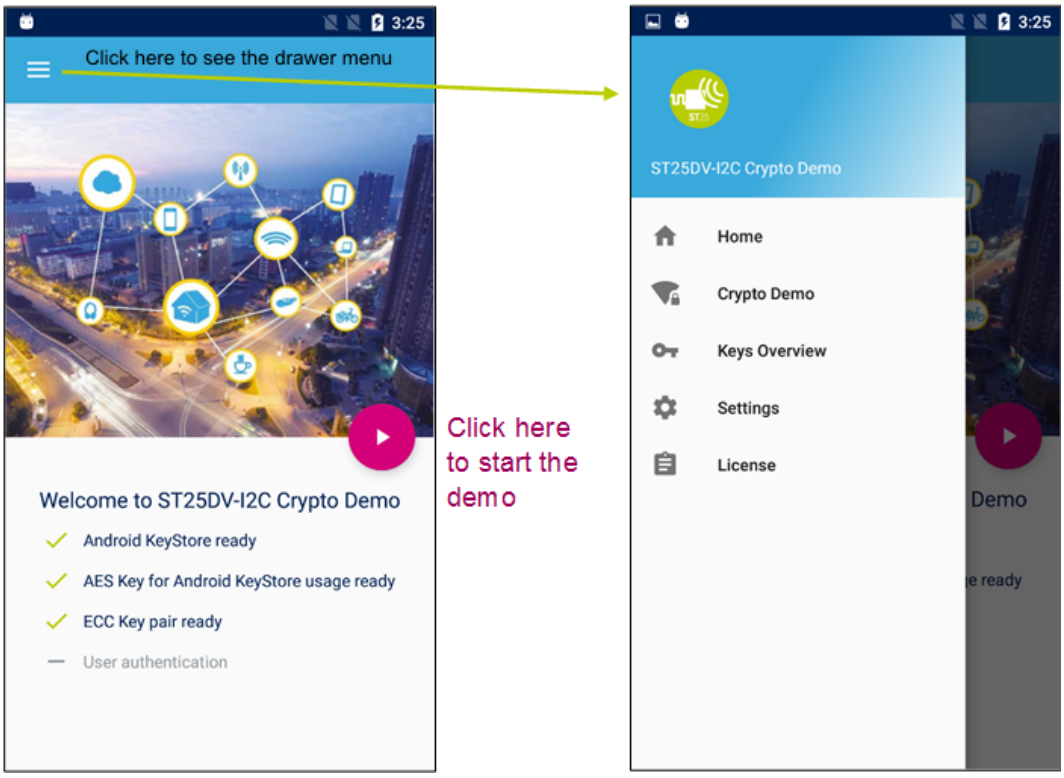
Note: No block-padding is required by this encryption method.

3 "ST25DV-I2C Crypto Demo" application screens

3.1 Home screen

User manually launches the application “ST25DV-I2C-Crypto Demo” or simply taps the ST25DV-I2C NFC tag, Android™ automatically launches the “ST25DV-I2C-Crypto Demo” application.

Figure 6. "ST25DV-I2C Crypto Demo" - Android™ home screen



When the application starts, it initializes the Android™ KeyStore and some cryptography elements.

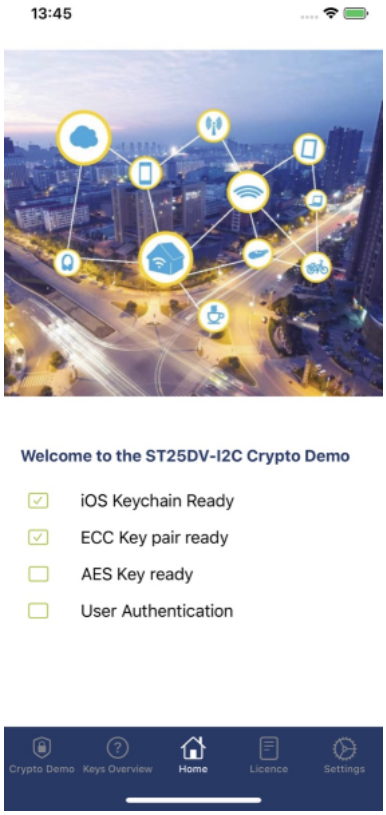
By default, the “User authentication” is disabled but this can be changed in the "Settings" menu. If enabled, the user has to enter its pin-code or fingerprint every time this application starts.

On iOS™ application, user must launch the application manually since NFC is only enabled on demand by user application.

When the iOS™ application starts, a home screen is displayed and a tab bar appears at the bottom of the application screen (see Figure 7). It provides the ability to quickly switch between different sections of an application.

Push on section "Crypto Demo" to start demonstration.

Figure 7. **"ST25DV-I2C Crypto Demo" - iOS™ home screen**



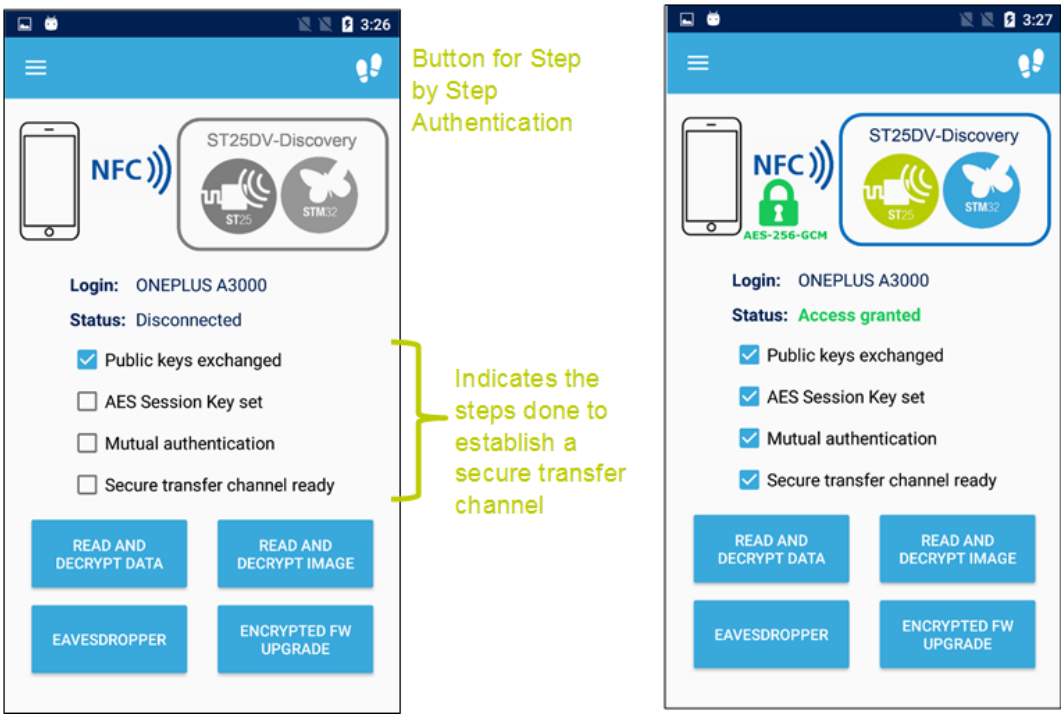
3.2 "ST25DV-I2C Crypto Demo" application main screen

This section describes how the "ST25DV-I2C Crypto Demo" application works.

3.2.1 Secure transfer channel setup

When tapping the ST25DV-I2C tag, the "ST25DV-I2C Crypto Demo" application automatically executes all the steps described from Section 2.1 Public key exchange to Section 2.5 Authentication of the connected device to setup a secure transfer channel.

Figure 8. "ST25DV-I2C-Crypto Demo" - Android™ secure transfer channel setup

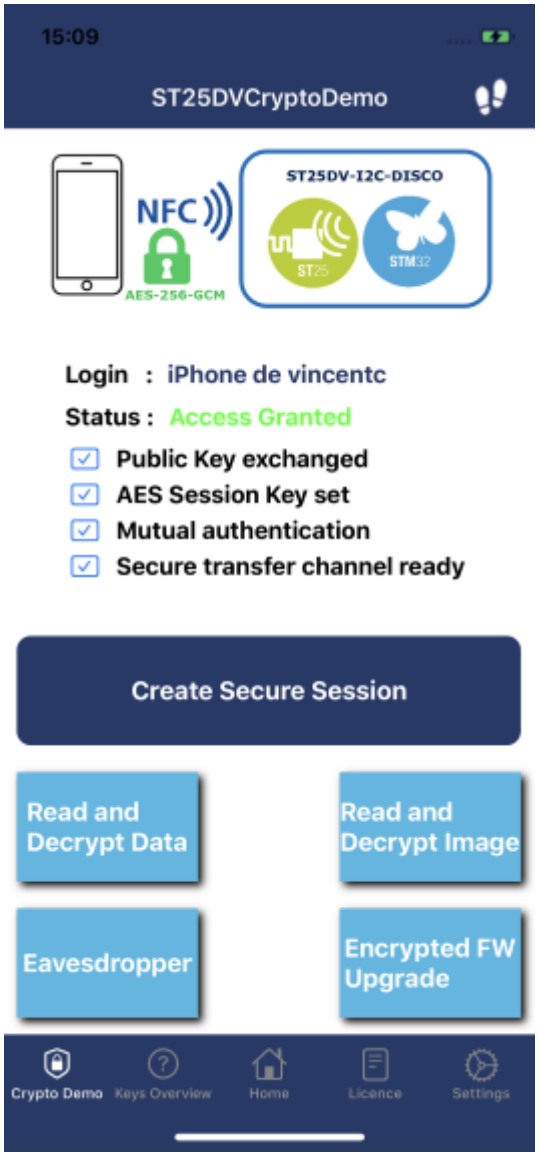


On the X-NUCLEO-NFC04A1, the LED2 (blue) is switched ON to indicate that the secure transfer channel is setup.

On the iOS™ application, the "Crypto Demo" includes a button to create the secure session, as NFC is only enabled on demand.

When pushing button "Create Secure Session", the iOS™ application starts the setup of a secure transfer channel. Secure session remains valid until the firmware closes the session .

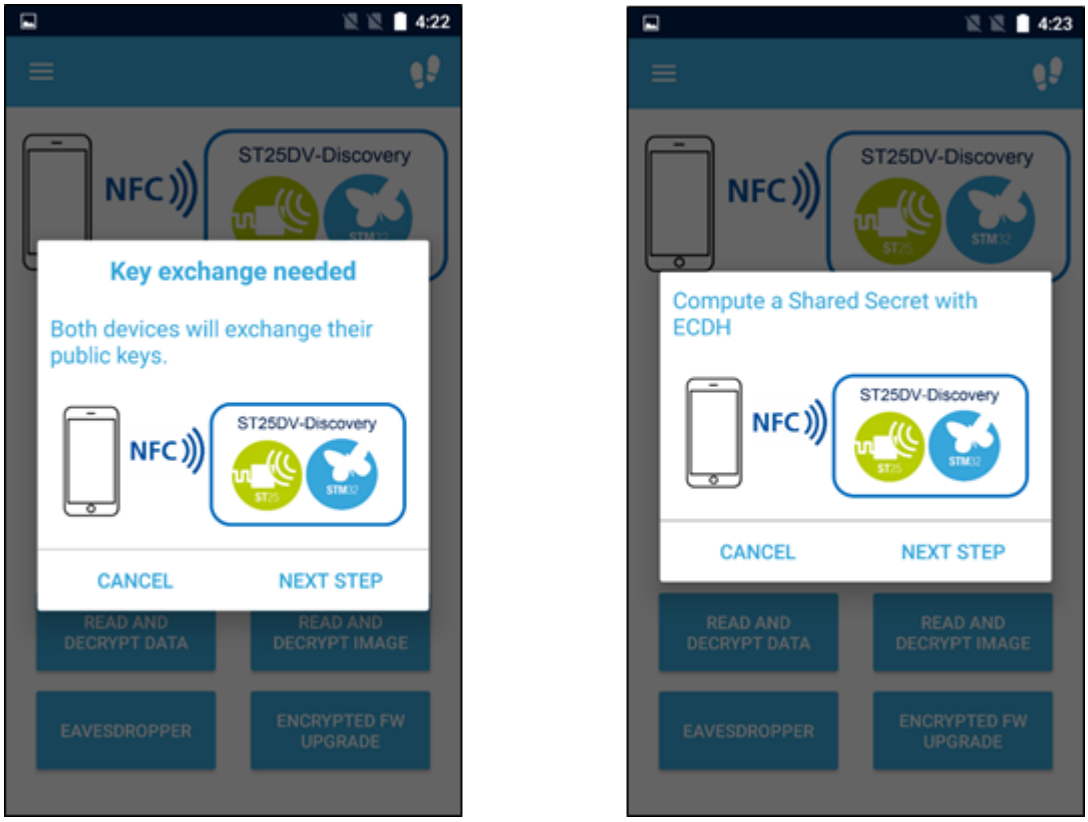
Figure 9. **"ST25DV-I2C-Crypto Demo" - iOS™ secure transfer channel setup**



3.2.2 Step by step mode

In the step by step mode, the user is guided through all the “keys exchange” and “mutual authentication” process.

Figure 10. "ST25DV-I2C-Crypto Demo" - step by step mode



3.2.3 “Read and decrypt data” button

Once the STC is ready, click on “read and decrypt data” button to download some data from the firmware.

This data (an array of points) is generated by the firmware running on the NUCLEO-L476RG board (it is not a real measurement) and is stored in the STM32L476 memory. This demonstration emulates the behaviour of a sensor regularly recording, for instance, the temperature and transferring this data to a smartphone through NFC.

The transfer occurs in the following steps:

1. The smartphone writes an encrypted command to the ST25DV-I2C FTM buffer mailbox to request the reading of data.
2. The firmware reads the mailbox buffer and decrypts the command.
3. The firmware encrypts and writes the data through I^2C bus to the ST25DV-I2C FTM mailbox buffer.
4. The smartphone reads over NFC and decrypts the mailbox buffer content to finally display the corresponding graph.

Note: The shape of the curve changes after each transfer. The user then restarts the transfer to receive the new set of points.

Figure 11. **"ST25DV-I2C Crypto Demo" - data transfer**



Android iOS

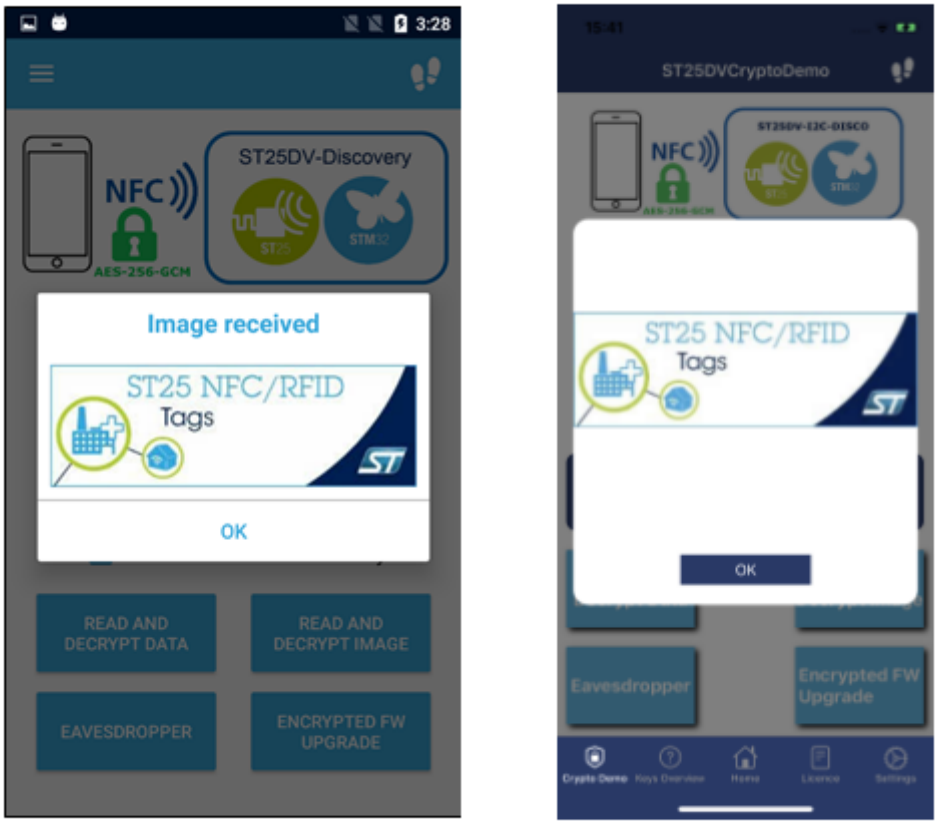
3.2.4 "Read and decrypt image" button

Once the STC is ready, the user must click on "read and decrypt image" button to download a picture from the ST25DV-I2C-DISCO.

The transfer occurs by following steps:

1. The smartphone writes an encrypted command to the ST25DV-I2C FTM mailbox buffer to request an image.
2. The firmware reads the mailbox buffer and decrypts the command.
3. The firmware encrypts the picture and writes all the chunks of the picture through I^2C to the ST25DV-I2C FTM mailbox buffer.
4. The smartphone reads the chunks over NFC and decrypts the encrypted picture to finally display it.

Figure 12. **"ST25DV-I2C Crypto Demo" - picture transfer**

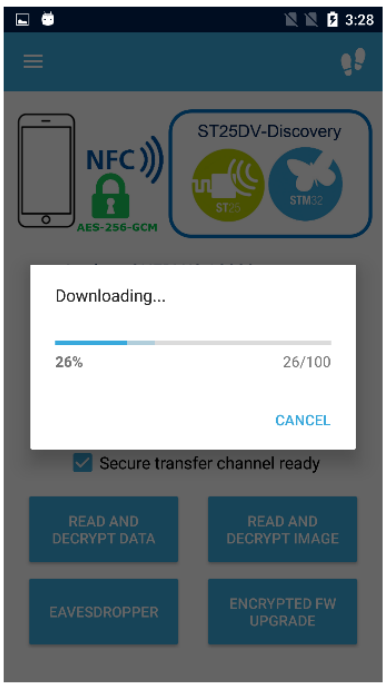


Android iOS

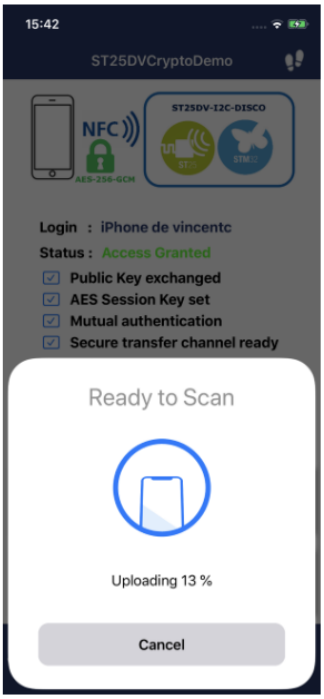
Note: A progress bar is displayed during the download. On the smartphone, two progress bars are visible, a blue one and a light blue one.

- *The blue one corresponds to the data received and acknowledged (data integrity has been checked)*
- *The light blue one corresponds to data received but not yet acknowledged. They might be trashed if the integrity verification detects an error.*

Figure 13. **"ST25DV-I2C-Crypto Demo" - progress bar**



Android



iOS

3.2.5 "Encrypted firmware upgrade" button

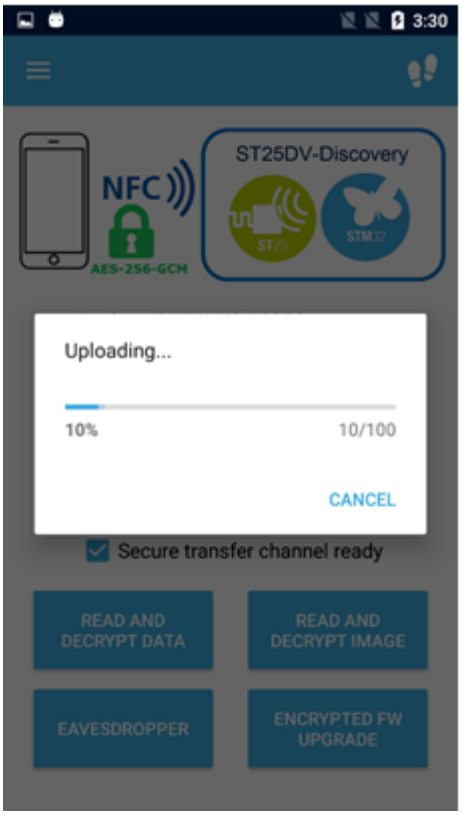
When user pushes "Encrypted firmware upgrade" button, the firmware is chosen in the phone internal memory. Firmwares are provided in the STSW-ST25DV005 package, see Section 1.4.2 Android™ APK application installation.

After selecting the file, the new firmware is uploaded, encrypted to the NUCLEO-L476RG board, by writing chunks to the ST25DV-I2C FTM buffer mailbox. The "ST25DV-I2C Crypto Demo" firmware checks the authenticity of the uploaded firmware chunks (thanks to the GMAC) and flashes them in the STM32L476RG static memory.

Once the new firmware has been successfully received, a reset is triggered and the secure boot installs and starts executing it.

Note: The firmwares provided for upload are the same demonstration using different pictures for the "Read Decrypt Image demo".

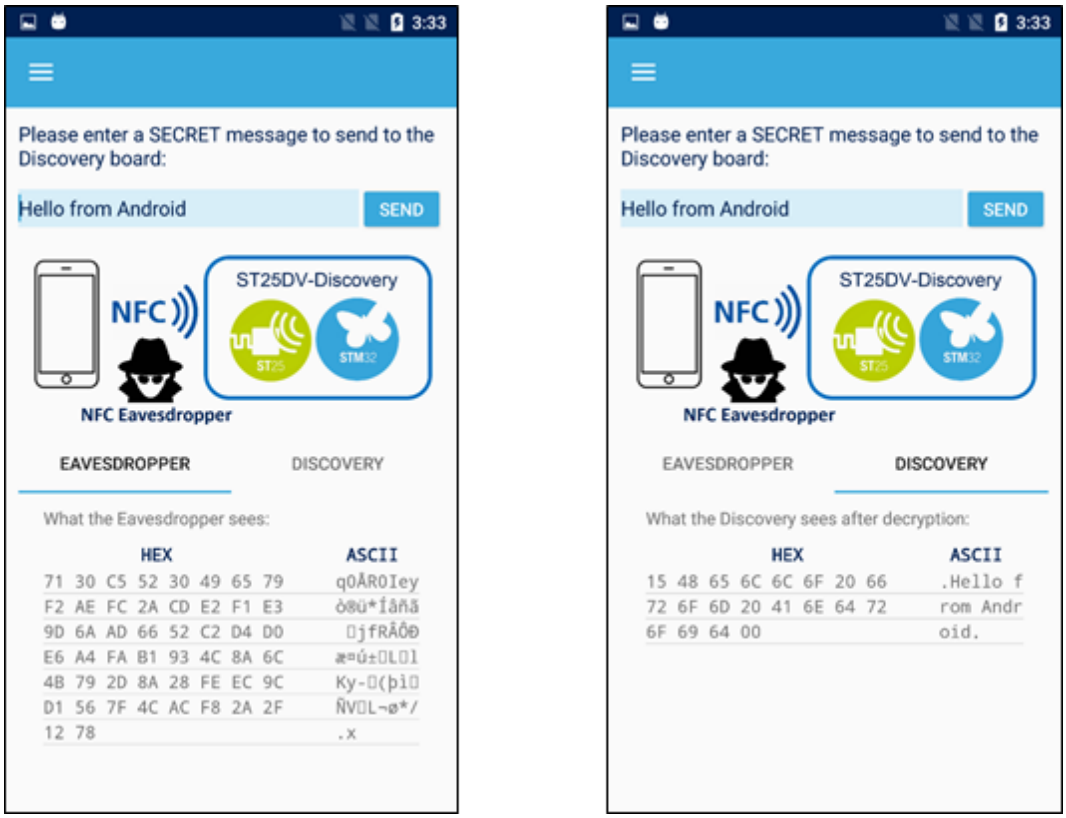
Figure 14. **"ST25DV-I2C Crypto Demo" - firmware upgrade**



3.2.6 Eavesdropper screen

This screen is used to show what someone sees when spying the NFC connection. It permits to see the encrypted data that are exchanged through NFC and what the firmware gets after decryption.

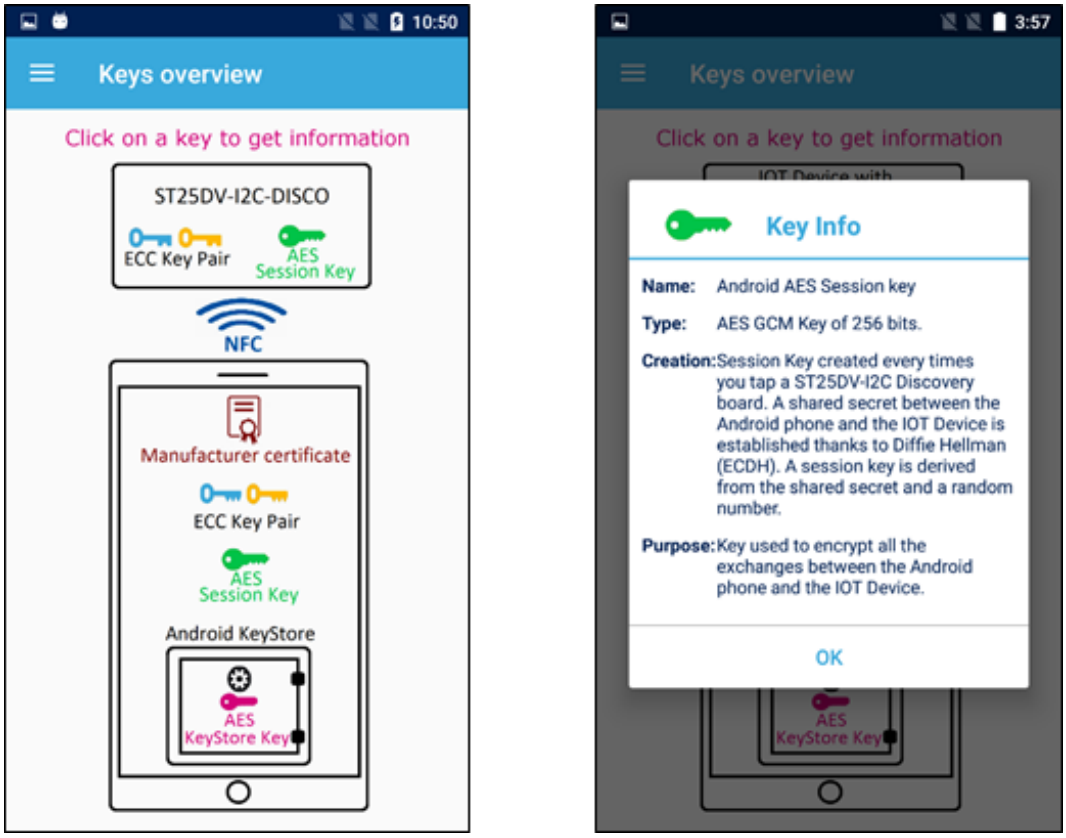
Figure 15. "ST25DV-I2C Crypto Demo" - eavesdropper



3.3 Keys overview

Various cryptographic keys are used in the "ST25DV-I2C Crypto Demo". This screen helps the user to understand the role of each key. In the application, click on a key to get information about it.

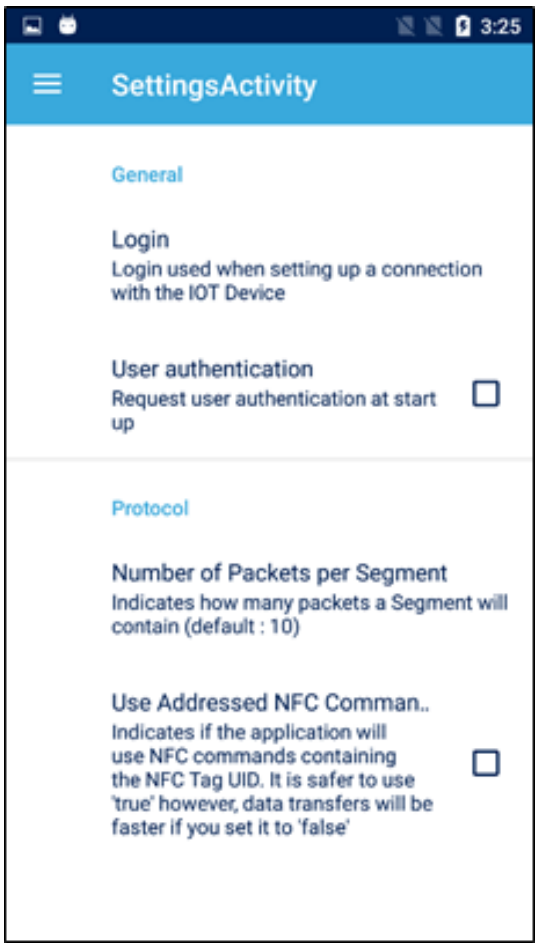
Figure 16. **"ST25DV-I2C-Crypto Demo" - keys overview**



3.4 Setting screen

This screen is used to change some settings of the "ST25DV-I2C Crypto Demo" application.

Figure 17. "ST25DV-I2C-Crypto Demo" - settings



Role of each setting field:

- ***Login***: When setting up a connection , the smartphone model name is used as "Login". This "Login" can be changed here.

Note: A hacker may want to change the login name to usurp the identity of the authorized user. It is impossible because the STM32L476RG saves the "Login" name plus the associated "Public key". The hacker may use the "Login" and the "Public key" of the authorized user but cannot pass the authentication phase because the hacker unknowns the "Private key".

- **User authentication**: Indicates if user pin-code or fingerprint is requested every time the user starts this application. This is an extra security to ensure that only the owner of the smartphone can use this Android™ application to communicate with the IOT device (represented by the NUCLEO-L476RG plus X-NUCLEO-NFC04A1 kit in this demonstration).
- **Number of Packets per Segment**: The smartphone and the firmware are exchanging data through the ST25DV-I2C FTM mailbox (which contains up to 256 bytes). When sending data bigger that 256 bytes, the data is split in blocks of data fitting in the mailbox that are called “Packets”. A group of N packets is called “Segment”. Each segment is encrypted separately, it has its own GMAC for authentication and integrity verification, and it is acknowledged by the receiver (if integrity is correct). This setting is able to change the number of packets per segment used when sending data to the firmware. This is the case for instance when doing a firmware upgrade.

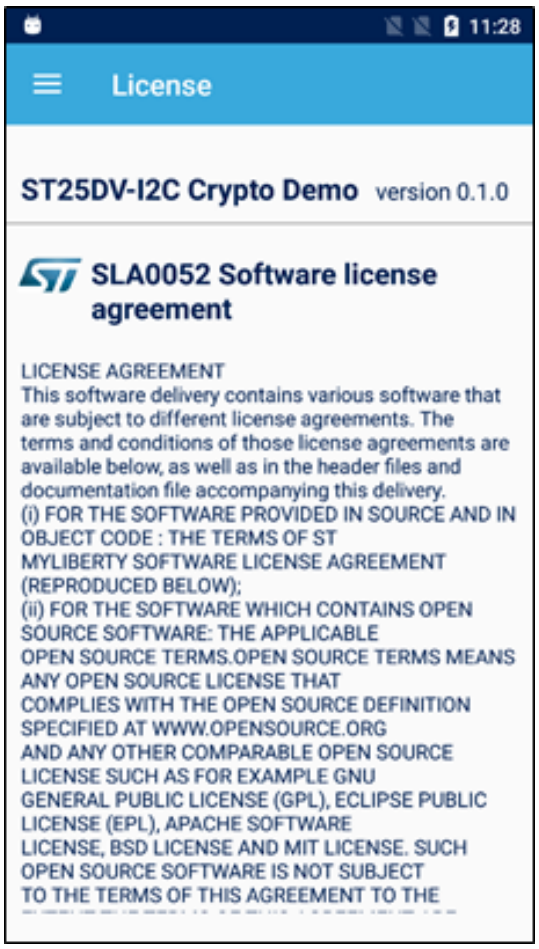
- **Use Addressed NFC command**: Communication between the smartphone and the firmware is done through NFC Type 5. Two modes are available for NFC Type 5 commands (this option is disabled in iOS™ application, as only non-adressed mode is available for iOS13 onward):
  - Addressed
  - Non-addressed

  When using addressed commands, the NFC Tag UID (on 8 bytes) is included in the command. This is safer because only the targeted tag processes the command but the communication is slower due to those extra bytes to send for every packet exchanged.

3.5 License screen

This screen shows the software license agreement that user accepts when using this program. It also indicates the version number of the program.

Figure 18. "ST25DV-I2C-Crypto Demo" - license



Revision history

Table 3. **Document revision history**

Date	Version	Changes
07-May-2020	1	Initial release.
25-Nov-2020	2	Updated Section 1.4.1 NUCLEO-L476RG and X-NUCLEO-NFC04A1 setup and Section 1.4.2 Android™ APK application installation

Contents

1 **General information** 2
1.1 Purpose and scope 2
1.2 Acronyms 3
1.3 Hardware equipment 3
1.4 Installation 3
1.4.1 NUCLEO-L476RG and X-NUCLEO-NFC04A1 setup 3
1.4.2 Android™ APK application installation 3
1.4.3 iOS™ application installation 3
1.4.4 How to set the "Authorized User" 4
1.5 License scheme 5
2 **Security processes** 6
2.1 Public key exchange 6
2.2 Definition of a "Shared Secret" 6
2.3 Derivation of a public key 6
2.4 Authentication of the smartphone 7
2.5 Authentication of the connected device 7
2.6 Encrypted data transfer 8
3 **"ST25DV-I2C Crypto Demo" application screens** 9
3.1 Home screen 9
3.2 "ST25DV-I2C Crypto Demo" application main screen 11
3.2.1 Secure transfer channel setup 11
3.2.2 Step by step mode 13
3.2.3 “Read and decrypt data” button 13
3.2.4 “Read and decrypt image” button 14
3.2.5 “Encrypted firmware upgrade” button 16
3.2.6 Eavesdropper screen 18
3.3 Keys overview 19
3.4 Setting screen 20
3.5 License screen 21

Revision history 22

List of tables

Table 1. Acronyms . . . 3
Table 2. ST25DV-I2C cryptographic firmware - License scheme . . . 5
Table 3. Document revision history . . . 22

List of figures

Figure 1. Connection scheme of the "ST25DV-I2C Crypto Demo" . . . 2
Figure 2. Blue icon view . . . 4
Figure 3. Elliptic curve Diffie-Hellman over NFC . . . 6
Figure 4. Smartphone authentication over NFC . . . 7
Figure 5. Firmware authentication over NFC . . . 8
Figure 6. "ST25DV-I2C Crypto Demo" - Android™ home screen . . . 9
Figure 7. "ST25DV-I2C Crypto Demo" - iOS™ home screen . . . 10
Figure 8. "ST25DV-I2C-Crypto Demo" - Android™ secure transfer channel setup . . . 11
Figure 9. "ST25DV-I2C-Crypto Demo" - iOS™ secure transfer channel setup . . . 12
Figure 10. "ST25DV-I2C-Crypto Demo" - step by step mode . . . 13
Figure 11. "ST25DV-I2C Crypto Demo" - data transfer . . . 14
Figure 12. "ST25DV-I2C Crypto Demo" - picture transfer . . . 15
Figure 13. "ST25DV-I2C-Crypto Demo" - progress bar . . . 16
Figure 14. "ST25DV-I2C Crypto Demo" - firmware upgrade . . . 17
Figure 15. "ST25DV-I2C Crypto Demo" - eavesdropper . . . 18
Figure 16. "ST25DV-I2C-Crypto Demo" - keys overview . . . 19
Figure 17. "ST25DV-I2C-Crypto Demo" - settings . . . 20
Figure 18. "ST25DV-I2C-Crypto Demo" - license . . . 21

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2020 STMicroelectronics – All rights reserved