P&O EAGLE

Rafael Galvez Vizcaino Vincent Rijmen

February 20, 2017

Contents

1 Introduction

A drone can fly autonomously given some vision capabilities and a set of instructions that discover the desired path. The students will work in a scenario where the instructions are navigation commands spread along the floor through QR code printouts. The navigation commands contain the coordinates from both the current station and the next waypoint.

These two-dimensional barcodes will contain several navigation commands, each for a different group. If the drone read the wrong coordinates, or an adversary could predict the trajectory and set up an attack or fake the real localization, the drone's autonomy would be greatly compromised.

Therefore, we need to protect that information against attacks against its integrity (modifying the navigation command) and its confidentiality (predicting the waypoints).

The cryptography submodule aims to provide such protections. Using a symmetric cipher, we will authenticate the data and keep it secret for anybody who does not have the correct key. A traditional cipher only protects the confidentiality of the information: the correct key is needed in order to have access to it; but Authenticated Encryption ciphers address both integrity and confidentiality: only if the contents have not been tampered with, the decryption operation returns the information to the holder of the key.

There are multiple printouts of the QR codes, one per station. If we used the same key for all the stations, the adversary would be able to read and potentially modify all the QR codes by cracking only one key. To avoid this problem, we will use a different key for each QR code. Similarly, if we were to use the same keys for all the student groups, any group could read and potentially modify the commands of the others; for that reason, we will have a key per QR code per group, so that each navigation command is readable only by the corresponding group.

This solution introduces a new problem: how do students get the keys for each QR code? Instead of having them hardcoded in the software itself, we will derive them from a single password, different for each group, combined with information from the QR code (its ID). In this way, we will obtain a key that decrypts and authenticates only the command to the group holding the password.[1]

This command will inform "LQR navigation control" how to arrive at the next station. By providing the coordinates of the next station, this submodule will make sure the drone progresses through the correct path.

2 Task

The goal of the submodule is to produce a system that handles a cropped photograph of a station QR code, authenticates the data in it and then decrypts it to hand the coordinates of the next station over to the "LQR navigation control" subsystem.

To get the data from the QR code, the students are provided with ZBar, a software library that decodes its cropped photograph and outputs the data

[1] Actually, because the QR codes can contain only a few bytes, three teams will share the same passwords and keys.

as a string. Each QR code will have 3 encrypted navigation commands, each for a different group: students must ignore those which are not for them. To obtain their own command, the students will have to derive the appropriate key using a Key Derivation Scheme: Argon2; and decrypt-authenticate it using an Authenticated Encryption cipher: ASCON. The decrypted command will then be sent as input to the "LQR navigation control" submodule.

The students will first implement a prototype of the complete crypto subsystem in a PC. This will make the development phase faster and easier, due to the absence of physical and resources constraints present in the drone. To test the prototype, they will use the example provided in this document. Once this milestone is achieved, they will deploy the system in the Zybo, making sure it remains functional, as well ensuring that the communication paths between "Vision" and "LQR navigation control" subsystems function properly.

Figure 1 gives a block diagram shows the overall description of the task.

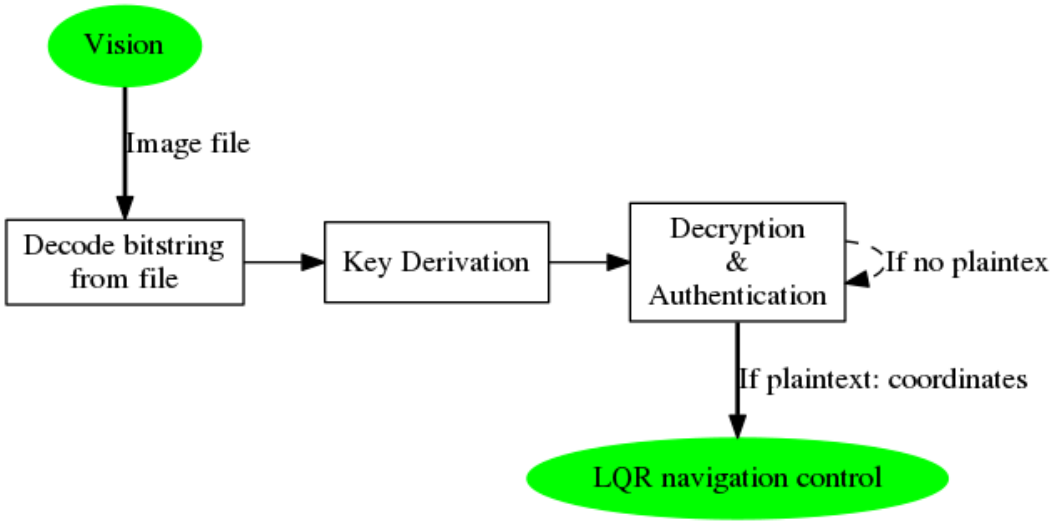

Figure 1: Task diagram

2.1 QR code generation example

This section describes the process of encryption of a QR code. It must be used to test the decryption system step by step. The QR code contains

binary data encoded in base 64. The students will have to decode it first, and then extract the real binary data.

In a single QR code, there is a different command per set of three groups. Table 1 gives a conceptual view. A more detailed description of the information in the QR codes will follow later in this document.

Table 1: Example QR code identifier and commands (conceptual view)

QR code identifier	1
Command1	0206
Command2	-0-0
Command3	-0-0

To decrypt the commands, we need the station keys. We first use Argon2 to derive the master keys from the passwords. Each of the sets of 3 groups has a different password. Table 2 shows the salt for the master key derivation (which is also the salt to be used in the project), 3 example passwords and the resulting master keys. Both the salt and the password are given as the actual inputs for Argon2 (no base-64 decoding should be done). The master key output is given in base-64 encoded format; only the first 16 characters are shown.

Table 2: Master key generation: examples

Master key salt	Password	Master key (base 64)
5X7X7Q6HXN82RWDB	KYGLYS3D4FXN9LKJ	uh3ChrcC801CwaDo
5X7X7Q6HXN82RWDB	LUC9FLJ2Y38COXRD	thIeHMX6EGVp4UmQ
5X7X7Q6HXN82RWDB	8AIUJQFTML4JGRK6	Dt1BVhdPimlBPPrK

Once the master key is derived we use again Argon2 to derive the station keys from the master key and thet station key salts. Table 3 gives a few examples. The station key salt is given as actual input for Argon 2 (no base-64 decoding should be done). The master key input and the station key output is given in base-64 encoded format; only the first 16 characters are shown.

We use ASCON to encrypt the commands and generate their authenticity tags using the station keys. Since ASCON is a symmetric cipher, the same keys will also be used to decrypt and authenticate the commands. Table 4 shows the first 10 characters of the nonce, the encrypted commands and the first 10 characters of the authenticity tags that will be the input for ASCON decryption.

Table 3: Station key generation: examples

Master key (base 64)	Station key salt	Station key (base 64)
uh3ChrcC801CwaDo	salt from qr01	tW0enq3S8yxiBzJp
thIeHMX6EGVp4UmQ	salt from qr01	E8zJOx8hfMRQOy9E
Dt1BVhdPimlBPPrK	salt from qr01	pFn/z8cNxFE+r5Ru

Table 4: Example nonce, encrypted commands and authenticity tags

Nonce (base64)	Encrypted commands (base64)	Authenticity tags (base64)
+HSP8++EYN	ZHy2yQ==	MnwGPhLMHl
+HSP8++EYN	Uy06TQ==	3k3oZ7UKmN
+HSP8++EYN	8b3CQw==	vzJRVzI6zu

The example QR code in Figure 2 has the nonce used with ASCON, the QR code identifier, the encrypted commands and their authenticity tags in the format detailed in Section 6. To decrypt and authenticate the navigation commands, use the station keys.

Figure 2: Example QR code

Decrypted command	0206
Decrypted command	-0-0
Decrypted command	-0-0

3 Provided for the students

The students will have:

- A software implementation of ASCON as a Python library
- A software implementation of Argon2 as a Python library
- A group password and the salt string to derive the master key.
- A software implementation of ZBar, a QR code reader.

4 Different sub-tasks

- Material reading: 1 person, 1 week
- QR code reader in a PC: 1 person, 1/2 week
- Implementation of the key derivation scheme in a PC: 1 week
- Implementation of the decryption in a PC: 1 person, 1 weeks
- Deployment of the complete decryption system in the drone: 1 person, 1/2 week
- QR code reader in the drone: 1 person, 1/2 week
- Implementation of its input and output interfaces: 1 person, 2 weeks
- Test with real QR codes in flight, 1 person, 2 weeks

5 Milestones

- Decryption on PC finished -> **T4**
  - QR code reader implementation in a PC
  - Implementation of the key derivation system in a PC
  - Implementation of the decryption system in a PC
  - Implementation of the decryption system in a PC
- Functional implementation on the drone -> **T5**
  - Deployment of the complete decryption system in the drone

  - QR code reader in the drone
  - Implementation of the input and output interfaces
    * Read the QR code with the built in camera
    * Pass along the decrypted command to the "LQR navigation control" subsystem.

- Final system fully integrated -> **T6**
  - Test with real QR codes in flight
    * Test landing
    * Test short path of 2 or 3 stations
    * Test complete path

6 Sub-tasks: detail

What needs to be done for each sub-task?

- QR code reader implementation in a PC: install ZBar to obtain the zbarimg command (already installed in the ESAT machines from 02.53 and 02.54), use it to read the Example QR code. Once decoded the base 64 string, the QR code data follows the format shown in Table 5.

Table 5: Format of the data encoded in a station QR code

Auxiliary information	Command1	Command2	Command3
22 bytes	20 bytes	20 bytes	20 bytes

- Implementation of the key derivation system in a PC: install the Argon2 python package and derive the both the master keys and the station keys in the example QR code. To derive new keys, Argon2 needs the original key and a random *salt* to randomize the new key. Students will derive the master key from their password, along with the salt specified in Table 2. To derive the station keys from the master keys, the salt will be the string "salt from qrN", N being the QR code identifier (e.g. 1). The students will find the QR code identifier in the *Auxiliary information*, cf. Table 6.

- Implementation of the decryption system in a PC: install the ASCON python package, decrypt the navigation commands in the Example

QR code and check their correctness with the results of the Example section.

To decrypt data, ASCON uses 128 bit keys, which will be the first 128 bits of the station key. A public number called *nonce*, used only once per key and per message, is required for further security. To authenticate the identifier of the QR code, we will write it to another parameter, the *associated data*. Finally, ASCON needs a cryptographic *tag* that enables the authentication of the encrypted data.

Students will find the nonce, the QR code identifier and its position in the *Auxiliary information*, cf. Table 6. As for the encrypted command and the authenticity tag, students will find them in the *Command* field, cf. Table 7.

Table 6: Auxiliary information format

Nonce	Identifier	Current station
16 bytes	2 bytes	4 bytes

Table 7: Encrypted command and authenticity tag

Encrypted command	Authenticity tag
4 bytes	16 bytes

Once the ciphertext is decrypted and the tag authenticated, the X-Y coordinates of the next station will be in the plaintext, cf. Table 8. For regular next-state coordinates, the bytes X_0, X_1, Y_0, Y_1 contain ASCII encodings of decimal digits. X_0 and Y_0 are the most significant digits of the coordinates. There are two special sets of next-station coordinates, that are used to denote other commands. For these special coordinates, both X_0 and Y_0 are the ASCII encoding of "-". If in addition both X_1 and Y_1 are equal to the ASCII encoding of "1", then the drone should land on top of the QR code. Alternatively, if both X_1 and Y_1 are equal to the ASCII encoding of "0", then the command is the "No OPeration (NOP) command"; the drone should ignore it.

Table 8: Decrypted Command format

$X_0X_1Y_0Y_1$
Next station

- Deployment of the complete decryption system in the drone: move the code to the drone, setup the libraries and re-test the system from the Zybo.
- Implementation of the input and output interfaces: integrate the QR reader and the decryption system in the workflow of the drone, to receive QR code images from the "Vision" subsystem as (e.g. PNG) files, decrypt them and hand the results to the "LQR navigation control" subsystem.
- Test with real QR codes in flight: test the decryption of a landing QR code, a series of 2 or 3 normal QR codes in a sequence, and the full path.

7 IP explanation

- Software implementation of ASCON (`https://github.com/meichlseder/pyascon`) and Argon2 (`https://pypi.python.org/pypi/argon2`): students will use these libraries to implement their complete decryption system of the station QR codes.
- Software implementation of ZBar, a QR reader: `https://sourceforge.net/projects/zbar/files/zbar/0.10/`. ZBar will parse the images obtained by the PSI subsystem.
- Example with intermediate results: this document describes what are the steps to obtain the navigation commands from a QR code. It also shows intermediate results that allow to test block by block the correctness of the module.

8 Documentation

- About Key derivation functions: `https://en.wikipedia.org/wiki/Key_derivation_function`. This Wikipedia page introduces the concept of a *Key Derivation Function*. It helps to undestand what is Argon2 and its use cases.
- About Authenticated Encryption: `https://en.wikipedia.org/wiki/Authenticated_encryption`. This Wikipedia page introduces the concept of *Authenticated Encryption*. Helpful to understand the problem that *Authenticated Encryption* ciphers solve.

- Argon2 python library documentation: `https://github.com/flamewow/argon2_py/blob/master/README.rst`. This file describes how to install and use the Argon2 Python library.
- ASCON python library documentation: `https://github.com/meichlseder/pyascon/blob/master/ascon.py`. The documentation of each function can be found directly in the source code. The only function that the students will use is $\text{ascon}_{\text{decrypt}}$. A description of the parameters can be found in the line 37.
- Python Virtual Environments. It is good practice to install Python libraries in a Virtual Environment. This document shows why and how to use it.
- Python Subprocess management. This module provides functionality to call external binaries from Python code.
- Cloning a repository from GitHub: `https://help.github.com/articles/cloning-a-repository/`. This documentation page from GitHub explains how to download (*clone*) repositories from GitHub, where the ASCON Python library is hosted.
- Unicode In Python, Completely Demystified: `http://farmdev.com/talks/unicode/`. This presentation helps to understand how we can use unicode to represent different kinds of strings (including binary data). Introduces the transformation, the encode and the decode operations that students will need to handle the binary data in the QR codes.
- Python Unicode: Encode and Decode Strings (in Python 2.x): `http://pythoncentral.io/python-unicode-encode-decode-strings-python-2x/`. This document helps to understand how the encodings are used in the encode and decode operations.