University of Cape Town

EEE3097S

Engineering Principles: Electrical And Computer Engineering

Second Progress Report

Authors:
David Young
Caide Spriestersbach

Student Numbers:
YNGDAV005
SPRCAI002

October 2022

Table of Contents

Table of Figures ..... *- 1 -*

List of Tables ..... *- 1 -*

Administrative Details ..... *- 2 -*

Individual Contributions ..... **- 2 -**

Project Management Tool ..... **- 2 -**

Development Timeline ..... **- 2 -**

GitHub Link ..... **- 2 -**

1. IMU Module ..... *- 3 -*

1.1. Additional Features ..... **- 3 -**

1.2. Testing of the IMU ensuring the system can be extrapolated to the buoy: ..... **- 3 -**

1.3. Validation test for the IMU module: ..... **- 3 -**

2. Experiment Setup ..... *- 6 -*

2.1. Experiments to Check Overall Functionality of the System ..... **- 6 -**

2.2. Experiments for Compression Block ..... **- 6 -**

2.3. Experiments for Encryption Block ..... **- 7 -**

2.4. Experiments for Checksum Block ..... **- 8 -**

2.5. Expected Data to be Retrieved and Returned From Each Block ..... **- 8 -**

3. Results ..... *- 9 -*

3.1. Results of Experiments to Check Overall Functionality of the System ..... **- 9 -**

3.2. Results of Experiments for Compression Block ..... **- 10 -**

3.3. Results of Experiments for Encryption Block ..... **- 12 -**

3.4. Results of Experiments for Checksum Block ..... **- 14 -**

3.5. Effects of Changing the Data Provided to the System ..... **- 16 -**

Under Sampling Experiment ..... - 16 -

High Sampling Rate Experiment ..... - 18 -

Gaussian Noise Experiment ..... - 19 -

4. Practical Data Acceptance Test Procedures ..... *- 21 -*

4.1. Compression ATPs ..... **- 21 -**

4.2. Encryption ATPs ..... **- 21 -**

4.3. Checksum ATPs ..... **- 22 -**

4.4 New Specifications ..... **- 22 -**

Compression ATP ..... - 22 -

Encryption ATP ..... - 22 -

References ..... *- 22 -*

Table of Figures

Figure 1 – Screenshot of project management tool front page ..... - 2 -
Figure 2 – Development timeline for the project ..... - 2 -
Figure 3 – The SparkFun 9DoF IMU Breakout ..... - 4 -
Figure 4 – STM2 CLI output for the overall functionality experiment ..... - 9 -
Figure 5 – STM1 CLI output for the overall functionality experiment ..... - 9 -
Figure 6 – Compression block input data size versus computational speed ..... - 11 -
Figure 7 – Encryption block input data size versus computational speed ..... - 13 -
Figure 8 – STM2 CLI output for the under-sampling experiment ..... - 16 -
Figure 9 – STM1 CLI output for the under-sampling experiment ..... - 16 -
Figure 10 – STM2 CLI output for increased sensor sampling rate experiment ..... - 18 -
Figure 11 – STM1 CLI output for increased sensor sampling rate experiment ..... - 18 -
Figure 12 – STM2 CLI output for gaussian noise experiment ..... - 19 -
Figure 13 – STM1 CLI output for gaussian noise experiment ..... - 19 -

List of Tables

Table I – Table of individual contributions made by each member ..... - 2 -
Table II – Raw sensor measurements from the accelerometer ..... - 4 -
Table III – Raw sensor measurements from the gyroscope ..... - 5 -
Table IV – Raw sensor measurements from the magnetometer ..... - 5 -
Table V – Results for the overall functionality experiment ..... - 10 -
Table VI – Raw results for the compression block experiments ..... - 10 -
Table VII – Calculated results for the compression block experiments ..... - 11 -
Table VIII – Raw results for the encryption block experiments ..... - 12 -
Table IX – Calculated results for the encryption block experiments ..... - 13 -
Table X – Raw results for the checksum block experiments ..... - 14 -
Table XI – Average checksum generation speed for checksum algorithm ..... - 14 -
Table XII – Raw results for the checksum block experiments ..... - 15 -
Table XIII – Under-sampling experiment compression results ..... - 16 -
Table XIV – Under-sampling experiment encryption results ..... - 17 -
Table XV – Increased sampling rate experiment compression results ..... - 18 -
Table XVI – Increased sampling rate experiment encryption results ..... - 19 -
Table XVII – Gaussian noise experiment compression results ..... - 20 -
Table XVIII – Gaussian noise experiment encryption results ..... - 20 -
Table XIX – Practical data ATPs for the compression block ..... - 21 -
Table XX – Practical data ATPs for the encryption block ..... - 21 -
Table XXI - Practical data ATPs for the checksum block ..... - 22 -
Table XXII – new specification for compression algorithm vs old specification ..... - 22 -
Table XXII – new specification for encryption algorithm vs old specification ..... - 22 -

Administrative Details

Individual Contributions

Both members worked equally on the project and contributed to each section. However, some areas were focused more heavily on by a specific member. These sections are listed below.

Name	Sections	Pages
David Young YNGDAV005	1.3, 2.2, 2.4, 3.1, 3.2, 3.4, 3.5, 4.1, 4.4	3, 6-12, 14-22
Caide Spriestersbach SPRCAI002	1.1, 1.2, 2.1, 2.3, 2.5, 3.3, 4.2, 4.3, 4.4	3, 6-8, 12-13, 21-22

Table I – Table of individual contributions made by each member

Project Management Tool

Below is a screenshot of the front page of our project management tool as of Saturday, 1st October 2022.

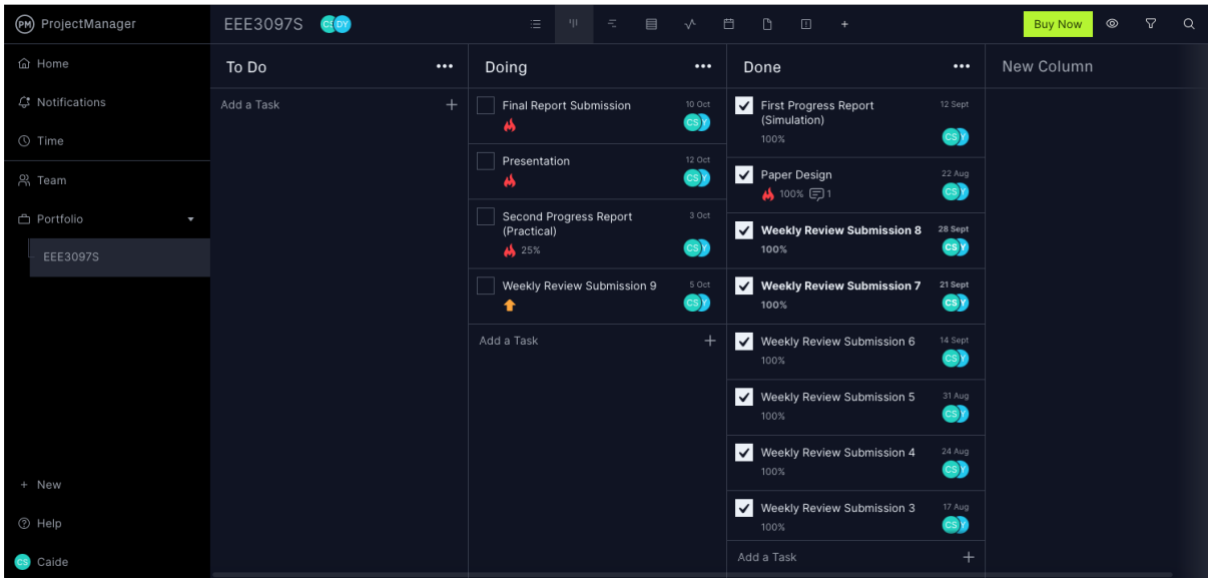

Figure 1 – Screenshot of project management tool front page

Development Timeline

As can be seen by the timeline below, the development is on time.

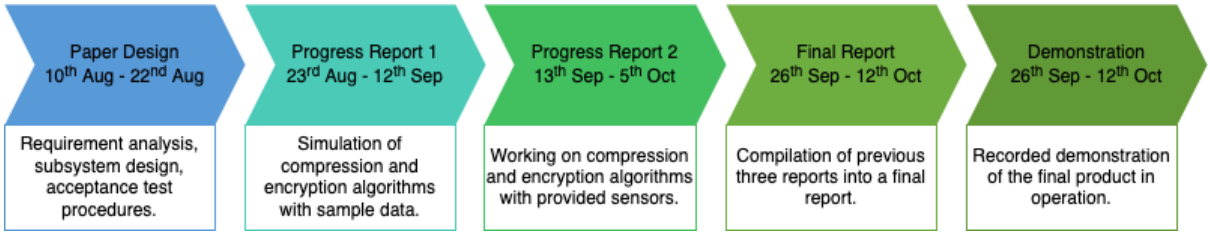

Figure 2 – Development timeline for the project

GitHub Link

The members made use of a GitHub git repository used throughout this project. Below is a link to the GitHub repository.
https://github.com/the-user-created/EEE3097S-Project

1. IMU Module

1.1. Additional Features

The system used the ICM-20948 MEMS motion tracking unit provided on the SparkFun 9DoF IMU Breakout. The ICM-20948 is the same sensor provided by the Sense Hat B. The decision was made to use only the MEMS motion tracking unit as it provides the system with only the needed data and requires less setup and configuration to communicate with the STM microcontroller. The Sense Hat B was provided to the team after successfully communicating with the ICM-20948; therefore, it was decided that we would not be using the Sense Hat B, although the system can be configured to use either sensor.

The most noticeable feature of using the ICM-20948 was that it consumes significantly less power when compared to the Sense Hat B; it was concluded that since the Sense Hat B has multiple features, it consumes more power. It was then decided that since there was no utilisation of all the features of the Sense Hat B since it did not replicate what will be implemented on the SHARK buoy, the ICM-20948 would be the only sensor used for testing the system. The ICM-20948 will provide recordings similar to what will be used on the SHARK buoy.

1.2. Testing of the IMU ensuring the system can be extrapolated to the buoy:

The ICM-20948 used for testing the entire system is made by the same company as the ICM-20649 onboard the SHARK buoy. It was decided to perform all tests solely with the SparkFun IMU instead of using the Sense Hat B. The SparkFun IMU is a 9-axis sensor, whereas the ICM-20649 is a 6-axis sensor. Although the experiments in the following sections used all nine axes provided by the SparkFun IMU, these same results can be extrapolated for using only the six axes provided by the ICM-20649.

The ICM-20649 only records the gyroscopic and accelerometer data, whereas the SparkFun IMU records gyroscopic, accelerometer, and magnetic data. There are no other significant differences between the SparkFun IMU and the one on the shark buoy. The SparkFun IMU provided more data than necessary, and this was done to ensure that the system could be extrapolated effectively to work on the SHARK buoy. The climate differences could not be tested since we cannot reproduce the conditions of Antarctica in South Africa.

1.3. Validation test for the IMU module:

Tests were done on the IMU to ensure that the STM was successfully powered on. The connection between the IMU and the STM was tested using the computer's serial interface to ensure the code was run successfully on the STM. The IMU's data was then collected and translated from binary to decimal to ensure that the readings were accurate compared to the ambient surroundings. The IMU was then moved around and placed in different environments to test if the data set would change, indicating that the sensor was working as expected. This data was collected and tabulated for comparisons to be made and to verify the success of the entire system.

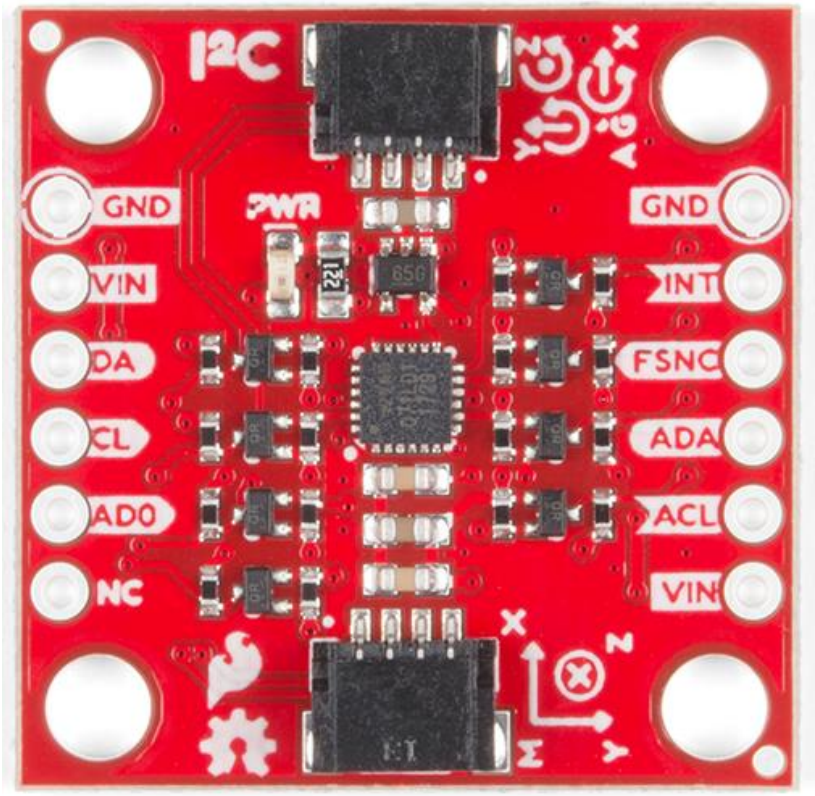

Figure 3 – The SparkFun 9DoF IMU Breakout

The table below shows the IMU first sitting flat on a table. The IMU was then placed so that its pins were perpendicular to the table, hence measuring gravity's downwards force on the sensor's x-axis. The IMU was then placed so that the pins were parallel to the table, hence measuring gravity's downwards force on the sensor's negative y-axis.

Reading Group	Time	Accel X	Accel Y	Accel Z
Control	0:00:00.845806	276	40	16451
	0:00:01.951676	278	43	16455
	0:00:03.057521	281	45	16460
	⋮	⋮	⋮	⋮
Perpendicular to table	0:00:11.901786	11842	73	5424
	0:00:13.007536	13847	68	3345
	0:00:14.112257	15892	73	1250
	⋮	⋮	⋮	⋮
Parallel to table	0:00:36.151303	247	-4492	13530
	0:00:37.257039	222	-6540	11816
	0:00:38.427122	211	-9918	9328

Table II – Raw sensor measurements from the accelerometer

The table below shows the IMU first sitting flat on a table. The IMU was then placed so that its pins were almost perpendicular to the table; hence the gyroscope's y-axis changed. The IMU was then placed so that the pins were parallel to the table; hence the gyroscope's x-axis changed.

Reading Group	Time	Gyro X	Gyro Y	Gyro Z
Control	0:00:00.325570	2	0	-1
	0:00:01.429950	2	1	0
	0:00:02.534451	1	1	0
	⋮	⋮	⋮	⋮
Perpendicular to table	0:00:04.744082	5	-148	32
	0:00:05.848769	4	-151	32
	0:00:06.953420	4	-144	32
	⋮	⋮	⋮	⋮
Parallel to table	0:00:25.739159	-147	-36	81
	0:00:26.844462	-161	58	80
	0:00:27.950040	-173	31	79

Table III – Raw sensor measurements from the gyroscope

The table below shows the IMU first sitting flat on a table without a strong magnet in its vicinity. A magnet was placed directly above the sensor (parallel with the sensor's z-axis). The magnet was then placed parallel to the sensor's pins (parallel with the sensor's x-axis). Lastly, the magnet was placed perpendicular to the sensor's pins (parallel with the sensor's y-axis).

Reading Group	Time	Magn X	Magn Y	Magn Z
Control	0:00:00.647816	-956	-119	44
	⋮	⋮	⋮	⋮
Magnet on z-axis	0:00:10.587180	-2338	-588	7831
	0:00:11.691698	-2021	-1962	10364
	0:00:12.796130	-1763	-3298	12830
	⋮	⋮	⋮	⋮
Magnet on x-axis	0:00:33.780674	-1683	-118	451
	0:00:34.885170	-2690	-182	717
	0:00:35.989632	-3695	-267	888
	⋮	⋮	⋮	⋮
Magnet on y-axis	0:00:54.765832	-1056	-446	137
	0:00:55.870350	-1151	-1164	290
	0:00:56.975013	-1243	-2174	556

Table IV – Raw sensor measurements from the magnetometer

2. Experiment Setup

2.1. Experiments to Check Overall Functionality of the System

The overall system uses a combination of encryption and compression functions compiled in C and managed by a python script for execution and testing. A checksum is performed on the collected sensor data from the IMU (stored on the STM microcontroller temporarily), the compression block will compress the data, and encryption is performed on the compressed data before being transmitted to the computer for testing. The encrypted data can then be decrypted and decompressed to retrieve the original data by passing the encrypted data back to a secondary STM, which has the decryption and decompression algorithm loaded; the STM will then perform the subsequent algorithms before returning the original data along with a checksum (which is generated on the secondary STM) for confirmation of the retrieval and storage of data to be confirmed – the checksum is used to ensure that the original sensor data is identical to the resultant data after decompression. Since both the compression and encryption algorithms are lossless, the input and output files should be identical. The checksum confirms this – this ensures that the efficiency of the overall setup is 100%.

A timer was used to determine the time taken for the system to collect a block of sensor data and perform the necessary algorithms on the data before transmission; this test was also performed on the decompression and decryption to ensure the data processing on the STM satisfies the requirements. The total time for the overall system to perform the above actions were used to verify the requirements that the encryption and compression blocks were efficient and did not exceed the time constraints outlined in the figure of merits. Once decompression and decryption had been performed, the original size of the sensor data was compared to the compressed data size to ensure that a compression ratio of at least 1.5 was achieved every iteration.

2.2. Experiments for Compression Block

Testing was done in phases to obtain values which could then be used for comparisons and verify the respective ATPs along with determining figures of merit. The compression algorithm was tested by performing compression on multiple sets of known data generated by a pseudorandom value generator using a seed. The seed used on the STM and the computer were the same – hence the data compressed on the STM and the data compressed on the computer were the same. This was done to validate that the algorithm is working on the STM as it was on the computer – confirming the algorithm is working as desired before the sensor data is passed to the compression block. Since the computer runs the same algorithm as the STM, the output file size and the output data must be the same as the ones returned by the computer. While the tests have peripheral objectives related to the ATPs, the main aim of the compression block is to reduce the physical space required to store the data whilst retaining all data inside the file.

The Lempel-Ziv-Storer-Szymanski (LZSS) compression algorithm was used for compression. The original sensor data set size is compared to the compressed data size for each iteration to determine the compression ratio – the compression ratio should be at least 1.5. The execution

time of the compression and decompression for each generated data set can be determined via the same methods used for the overall system; however, in this section, the file size was divided by the time taken to compression to ensure that at least one kilobyte of data is compressed every second.

Lastly, the input and output data sets are compared for each sample data set to determine if the compression and decompression are lossless – the checksum is also used to verify that the data collected by the sensor as well as the data generated by the STM using the seed is the same as the data in the output file following decompression. The process described above was repeated using data collected by the IMU and stored on the STM; the results from the generated data set were used as a control to verify that the compression block performed as expected.

2.3. Experiments for Encryption Block

Testing the encryption block consisted of various stages. Testing was done in phases to obtain values which could then be used for comparisons and verify the respective ATPs. The encryption algorithm was tested by performing encryption on multiple sets of known data generated by a pseudorandom value generator using a seed. Since the computer used for testing is a 64-bit processor while the STM microcontroller has a 32-bit processor, the machines' processors perform substitutions of the data into the S-boxes differently. This raised the issue that the computer cannot be used to verify the algorithm. To solve this problem, the sensor data encrypted by the STM is transferred to the client's computer along with the padding length used for the encryption and the checksum of the sensor data generated by the STM. The client can then see all the information related to the encryption before sending the data to another STM for decryption.

The generated data was used to confirm that the algorithm is working as desired before the user of real-world sensor data is passed to the encryption block – this was done as the generated data remained constant and was easily repeatable. The time taken for the computer to perform encryption on the generated data was recorded using a timer. This execution time is used to verify that the encryption block works as desired.

Since the STM generated a checksum when the data was collected before being compressed, this checksum can be used for comparison against the checksum generated following decryption to ensure that the encryption is lossless and that the client receives all data. While these tests have peripheral objectives related to the ATPs, the primary function of the encryption block is to convert plaintext data into ciphertext, which cannot trivially be decrypted without the password. The encryption algorithm used was a custom modified blowfish algorithm which Bruce Schneier originally developed – Twofish was unable to be implemented on the hardware of the STM. Hence, we were forced to revert to Twofish's predecessor, which was designed for hardware use. The original data was compared to the encrypted data for each data set to determine whether the encryption was successful. The execution time for encryption and decryption of each data set was determined using a timer. Determining if the encryption was lossless was done by comparing the compressed data provided to the encryption algorithm and the output data following decryption; the checksum was also used to verify as a secondary measure. The above information was then used to confirm that the figure of merits were satisfied to meet the user requirements. The encryption algorithm must be able to compress at least one kilobyte of data per second.

2.4. Experiments for Checksum Block

Testing the checksum block can be done by providing the CRC32 checksum algorithm with identical and different data sets and creating a checksum of each. To verify the CRC32 algorithm, a checksum of the sensor data collected by the remote STM is generated and transmitted with sensor data. Then the client's device receives the checksum and sensor data and sends it to their local STM. If the resultant 32-bit checksums of each data set are identical, then the data set has not changed. However, if the checksums are different, then the data is different, or in a rare case, the checksum was corrupted during transmission.

2.5. Expected Data to be Retrieved and Returned From Each Block

The compression block is expected to receive the raw IMU data in binary at a rate of 1 sample per 500 milliseconds, which will be compressed by executing the compression algorithm on the collected sensor data. The compressed data will also be binary and stored on the STM. The compressed binary data will be passed to the encryption block on the STM. The encryption block will then execute the encryption algorithm to convert the compressed sensor data into ciphertext before transmission to the client's computer. The client will then send the encrypted, compressed data in ciphertext to their STM for decryption. The decryption block will then execute its algorithm on the data which the client supplies. Following this, the decryption block will pass the decrypted data (compressed data) in binary to the decompression block, which will decompress the data so that the client can read the IMU's sensor data.

3. Results

3.1. Results of Experiments to Check Overall Functionality of the System

The system's overall functionality was tested on 132 bytes of collected sensor data. This data was then passed to the compression block. The compressed data was then passed to the encryption block, and the encrypted data was transmitted to the user. The above process is shown in Figure 4 below. The user is then asked if they want to decrypt and decompress the data. The outputs from the encryption block and compression block is shown in Figure 4 and 5 below.

```
SENSOR DATA:
Sensor data size: 132 bytes
0   ff ff ff ff 00 25 00 03  08 07 00 03 00 02 00 00
1   ff 91 00 07 00 0b ff ff  ff ff 00 48 00 08 10 0d
2   00 02 00 02 00 00 ff 24  00 0e 00 15 ff ff ff ff
3   00 6d 00 10 18 11 00 02  00 01 00 00 fe b7 00 15
4   00 20 ff ff ff ff 00 90  00 16 20 17 00 02 00 01
5   ff ff fe 4b 00 1a 00 2b  ff ff ff ff 00 b4 00 1b
6   28 1b 00 02 00 01 ff ff  fd dd 00 21 00 35 ff ff
7   ff ff 00 d8 00 21 30 1f  00 01 00 01 ff ff fd 70
8   00 28 00 40
CRC32 CHECKSUM:
0   b9 81 7e 70
COMPRESSED DATA:
Compression took: 189.0 milliseconds
Compressed data size: 103 bytes
0   ff 80 0a 01 25 80 40 e1  10 70 18 c0 20 40 f2 c8
1   c0 01 23 0b 0a a5 22 01  08 88 43 42 64 15 39 24
2   02 1d 00 8a 85 52 db 00  88 46 22 21 32 80 85 47
3   fd b7 09 8c 02 40 15 4c  84 02 2d 20 8b 85 4c 16
4   3f ea 5c 02 35 00 95 85  53 69 00 8d ca 23 61 55
5   fe f7 60 12 18 04 d4 2a  9d 80 48 cc 23 e3 f2 0a
6   9d c2 01 28 80 50 00
ENCRYPTED DATA:
Encryption took: 118.489 milliseconds
Encrypted data size: 104 bytes
0   de fa d8 d4 ad 4b 97 1a  39 30 c9 f1 8f 14 f3 4c
1   f4 a6 01 37 a4 8c 1f 9a  84 ad 35 64 b6 c0 ed 07
2   d4 be 12 c0 c0 55 ac 02  2b cf 8f 05 a4 6b 14 22
3   ec 60 a1 ae 15 dc 63 b0  b4 d8 88 f7 40 35 29 2d
4   0d f1 f2 42 61 90 bd 4e  40 c6 97 2c 7e 75 23 36
5   43 0f 09 7e e4 84 21 6d  a3 ce f7 5a c7 a2 56 76
6   ca 45 95 ee f0 e3 62 83
COMPRESSED DATA SIZE
0   67 00 00 00
PADDING LENGTH:
0   01
```

Figure 5 – STM1 CLI output for the overall functionality experiment

```
Decrypt and decompress? (y/n): y
PADDING LENGTH CONFIRMATION:
0   01
COMPRESSED DATA SIZE CONFIRMATION:
0   67 00 00 00
ENCRYPTED DATA CONFIRMATION:
0   de fa d8 d4 ad 4b 97 1a  39 30 c9 f1 8f 14 f3 4c
1   f4 a6 01 37 a4 8c 1f 9a  84 ad 35 64 b6 c0 ed 07
2   d4 be 12 c0 c0 55 ac 02  2b cf 8f 05 a4 6b 14 22
3   ec 60 a1 ae 15 dc 63 b0  b4 d8 88 f7 40 35 29 2d
4   0d f1 f2 42 61 90 bd 4e  40 c6 97 2c 7e 75 23 36
5   43 0f 09 7e e4 84 21 6d  a3 ce f7 5a c7 a2 56 76
6   ca 45 95 ee f0 e3 62 83
DECRYPTED DATA:
Decryption took: 117.606 milliseconds
Decrypted data size: 103 bytes
0   ff 80 0a 01 25 80 40 e1  10 70 18 c0 20 40 f2 c8
1   c0 01 23 0b 0a a5 22 01  08 88 43 42 64 15 39 24
2   02 1d 00 8a 85 52 db 00  88 46 22 21 32 80 85 47
3   fd b7 09 8c 02 40 15 4c  84 02 2d 20 8b 85 4c 16
4   3f ea 5c 02 35 00 95 85  53 69 00 8d ca 23 61 55
5   fe f7 60 12 18 04 d4 2a  9d 80 48 cc 23 e3 f2 0a
6   9d c2 01 28 80 50 00
DECOMPRESSED DATA:
Decompression took: 155.795 milliseconds
Decompressed data size: 132 bytes
0   ff ff ff ff 00 25 00 03  08 07 00 03 00 02 00 00
1   ff 91 00 07 00 0b ff ff  ff ff 00 48 00 08 10 0d
2   00 02 00 02 00 00 ff 24  00 0e 00 15 ff ff ff ff
3   00 6d 00 10 18 11 00 02  00 01 00 00 fe b7 00 15
4   00 20 ff ff ff ff 00 90  00 16 20 17 00 02 00 01
5   ff ff fe 4b 00 1a 00 2b  ff ff ff ff 00 b4 00 1b
6   28 1b 00 02 00 01 ff ff  fd dd 00 21 00 35 ff ff
7   ff ff 00 d8 00 21 30 1f  00 01 00 01 ff ff fd 70
8   00 28 00 40
CRC32 CHECKSUM:
Average checksum generation time: 0.067 milliseconds
0   b9 81 7e 70
```

Figure 4 – STM2 CLI output for the overall functionality experiment

The results for the overall functionality experiment are shown below in table form.

Algorithm	Input data size [bytes]	Output data size [bytes]	Execution time [milliseconds]
Compression	132	103	189.000
Encryption	103	104	118.489
Decryption	104	103	117.606
Decompression	103	132	155.795

Table V – Results for the overall functionality experiment

A padding length of 1 byte was used in the encryption and decryption, as can be seen in Figure 4 and Figure 5. This padding was added because the compressed data size was not a multiple of 8 bytes. This padding allows the entire input of the encryption algorithm to be encrypted. The checksum of the collected sensor data is b9817e70, as shown in Figure 4. This same checksum was generated for the output data from the overall system, as seen in the last line of Figure 5. Therefore, both the encryption and compression algorithms are lossless.

3.2. Results of Experiments for Compression Block

The results for the compression block experiments are shown below in table form.

Round	Uncompressed data size [bytes]	Compressed data size [bytes]	Compression time [ms]	Decompression time [ms]
1	66	52	105.228	82.075
2	176	131	227.086	205.773
3	286	169	292.021	327.105
4	396	225	367.486	446.712
5	506	269	425.389	567.747
6	616	351	557.499	693.522
7	726	374	585.581	814.379
8	836	439	686.253	935.235
9	946	422	655.423	1052.512
10	1012	430	654.141	1122.388

Table VI – Raw results for the compression block experiments

The values in the table below were calculated from those in the table above.

Round	Compression Ratio	Compression Speed [bytes/second]	Decompression Speed [bytes/second]
1	1.269	627.209	804.143
2	1.344	775.037	855.311
3	1.692	979.382	874.337
4	1.760	1077.592	886.477
5	1.881	1189.499	891.242
6	1.755	1104.935	888.220
7	1.941	1239.794	891.477
8	1.904	1218.210	893.893
9	2.242	1443.343	898.802
10	2.353	1547.067	901.649
Average	1.814	1120.207	878.555

Table VII – Calculated results for the compression block experiments

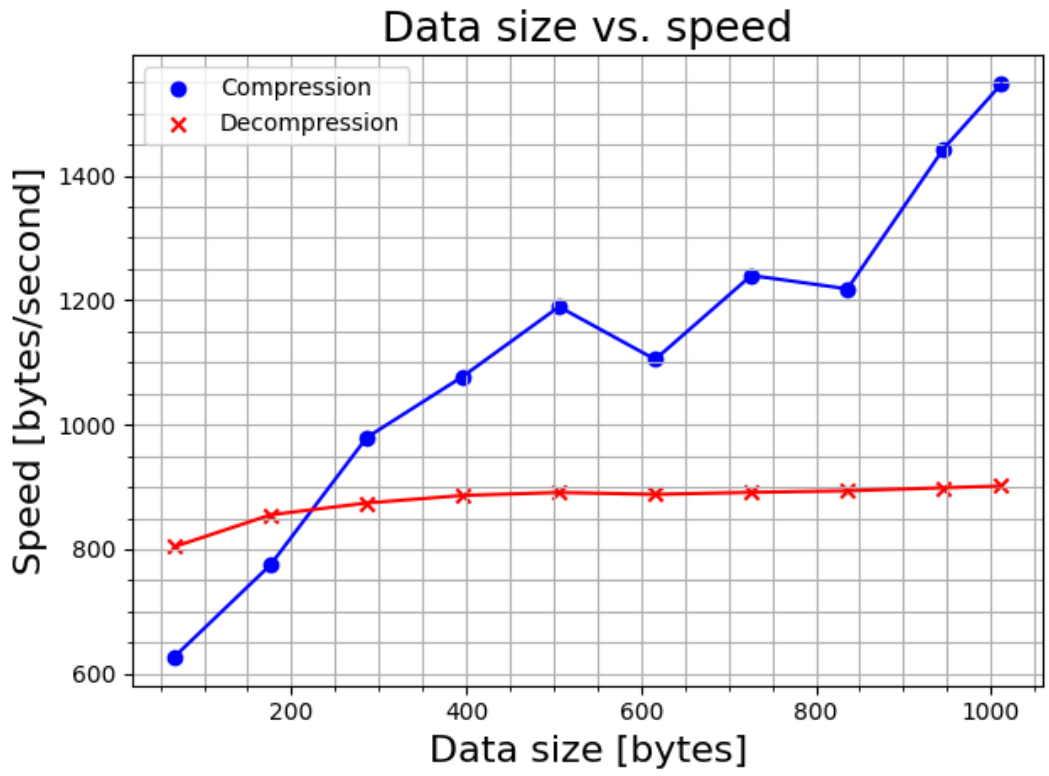

Figure 6 – Compression block input data size versus computational speed

The graph above indicates that the compression algorithm is more efficient than the decompression algorithm because as the size of the input data increases, the computational speed of the compression algorithm increases, reducing the system's total run time per block of sensor data. Whereas the decompression speed remains essentially constant, increasing the system's total run time per block of sensor data.

3.3. Results of Experiments for Encryption Block

The results for the encryption block experiments are shown below in table form.

Round	Decrypted data size [bytes]	Encrypted data size [bytes]	Encryption time [ms]	Decryption time [ms]
1	66	72	85.866	77.821
2	132	136	155.080	151.242
3	198	200	225.800	220.632
4	264	264	292.847	291.795
5	330	336	370.048	365.553
6	396	400	442.593	435.963
7	462	464	511.385	505.961
8	528	528	581.825	580.378
9	594	600	657.277	652.113
10	660	664	727.974	721.273
11	726	728	798.617	792.994

Table VIII – Raw results for the encryption block experiments

The values in the table below were calculated from those in the table above.

Round	Padding Length [bytes]	Encryption Speed [bytes/second]	Decryption Speed [bytes/second]
1	6	925.200	838.516
2	4	899.221	876.967
3	2	906.487	885.740
4	0	904.745	901.495
5	6	919.155	907.990

6	4	917.509	903.765
7	2	917.067	907.340
8	0	909.752	907.489
9	6	920.086	912.857
10	4	920.595	912.120
11	2	918.040	911.576
Average	-	914.351	896.896

Table IX – Calculated results for the encryption block experiments

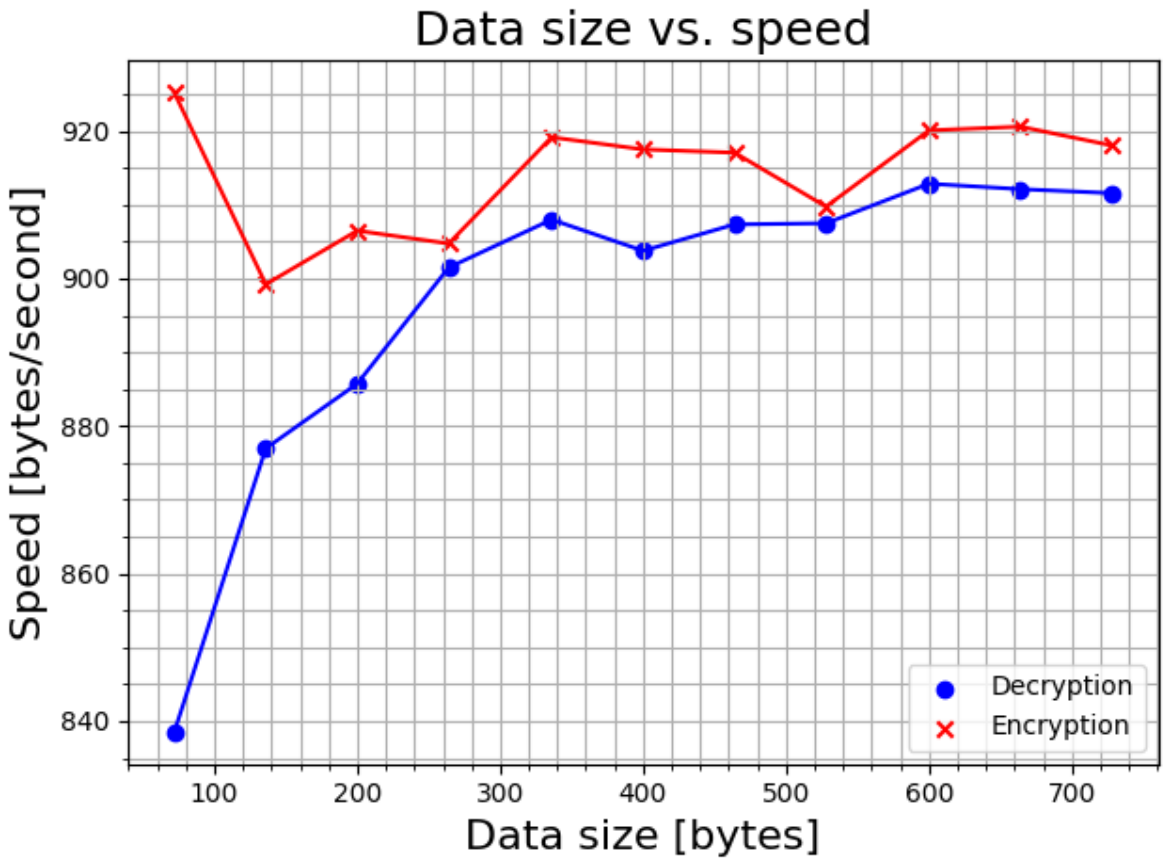

Figure 7 – Encryption block input data size versus computational speed

The graph above implies that as the size of the supplied data increases, the decryption and encryption speeds tend towards each other at about 915 bytes per second.

3.4. Results of Experiments for Checksum Block

The results for the checksum block experiments are shown below in table form.

Round	Data size	STM 1 Checksum	STM 2 Checksum	Average generation time [ms]
1	66	ea 54 a9 d5	ea 54 a9 d5	0.015
2	550	e5 a3 0b 4a	e5 a3 0b 4a	0.017
3	1034	bd 09 cb ed	bd 09 cb ed	0.019
4	1518	93 1b bb fe	93 1b bb fe	0.022
5	2002	f6 18 aa 7f	f6 18 aa 7f	0.024
6	2486	54 9d 79 3e	54 9d 79 3e	0.025
7	2970	5b f5 85 87	5b f5 85 87	0.029
8	3454	df 1b ae a0	df 1b ae a0	0.030
9	3938	0e 81 35 20	0e 81 35 20	0.033
10	4422	8c 39 d4 b7	8c 39 d4 b7	0.035

Table X – Raw results for the checksum block experiments

The checksums for every round match; therefore, the original sensor data matches the data received by the second STM. The values in the table below were calculated from those in the table above.

Round	Average generation speed [bytes/second]
1	4 400 000
2	32 400 000
3	54 400 000
4	69 000 000
5	83 400 000
6	99 400 000
7	102 000 000
8	115 000 000
9	119 000 000
10	126 000 000

Table XI – Average checksum generation speed for checksum algorithm

To demonstrate the effectiveness of the checksum algorithm, the table below shows the effect when the data is changed, causing STM2 to generate a different checksum.

Data size	STM 1 Checksum	STM 2 Checksum
66	74 51 92 4d	96 d1 38 a7
2002	fd 63 fa be	73 25 6a b6
4422	f3 fa 5f 39	a7 30 0d 27

Table XII – Raw results for the checksum block experiments

Only the first byte of each data set was changed after the sensor data was transmitted to the second STM before checksum generation, and even due to such a minuscule change, the checksums do not match. This shows how the checksum can be used reliably to indicate whether the algorithms are lossless and that no data is lost during transmission between the STMs and the client. However, if, during transmission, the checksum is affected (i.e., a single bit flips), then the checksum would indicate that the transmitted data has changed even though it may still be unchanged.

3.5. Effects of Changing the Data Provided to the System

Under Sampling Experiment

The CLI output of under sampling the data provided to the system is shown below.

```
SENSOR DATA:
Sensor data size: 132 bytes
0   ff ff ff ff 00 22 00 04  08 08 00 01 00 01 ff ff
1   ff 92 00 07 00 0b ff ff  ff ff 00 46 00 09 10 0e
2   00 01 00 01 ff ff ff 25  00 0e 00 16 ff ff ff ff
3   00 6a 00 0f 18 15 00 02  00 01 ff ff fe b8 00 16
4   00 21 ff ff ff ff 00 8b  00 14 20 1d 00 02 00 02
5   ff ff fe 4a 00 1e 00 2b  00 00 00 00 00 00 00 00
6   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
7   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8   00 00 00 00
CRC32 CHECKSUM:
0   d4 7e 8f e6
COMPRESSED DATA:
Compression took: 139.111 milliseconds
Compressed data size: 74 bytes
0   ff 80 0a 01 22 80 41 21  10 88 04 04 02 20 c2 c9
1   40 20 f0 08 58 55 28 d0  08 4c 42 1c 15 69 2c 00
2   12 31 60 aa 5a a0 10 f8  c4 56 01 02 0a 9f fb 70
3   13 18 04 84 2a 98 b8 04  52 41 1d 0a 94 08 2a 54
4   a8 04 7a 01 2b 33 f8 03  c0 16
ENCRYPTED DATA:
Encryption took: 93.338 milliseconds
Encrypted data size: 80 bytes
0   77 86 fd bc ec 1f 84 71  4d 9f 03 3d ad af 48 3a
1   b2 63 77 44 e3 06 3e b8  65 b7 cc d4 5c db 92 96
2   90 06 43 ce ec c1 12 b3  c9 d1 a7 87 cb de ab 57
3   90 1a a9 13 90 29 78 37  46 9f 26 88 59 b4 b8 1c
4   ff bd 5b b9 92 5b 68 03  3a c3 99 36 6e e8 f8 eb

COMPRESSED DATA SIZE
0   4a 00 00 00
PADDING LENGTH:
0   06
```

Figure 9 – STM1 CLI output for the under-sampling experiment

```
Decrypt and decompress? (y/n): y
PADDING LENGTH CONFIRMATION:
0   06
COMPRESSED DATA SIZE CONFIRMATION:
0   4a 00 00 00
ENCRYPTED DATA CONFIRMATION:
0   77 86 fd bc ec 1f 84 71  4d 9f 03 3d ad af 48 3a
1   b2 63 77 44 e3 06 3e b8  65 b7 cc d4 5c db 92 96
2   90 06 43 ce ec c1 12 b3  c9 d1 a7 87 cb de ab 57
3   90 1a a9 13 90 29 78 37  46 9f 26 88 59 b4 b8 1c
4   ff bd 5b b9 92 5b 68 03  3a c3 99 36 6e e8 f8 eb

DECRYPTED DATA:
Decryption took: 85.69 milliseconds
Decrypted data size: 74 bytes
0   ff 80 0a 01 22 80 41 21  10 88 04 04 02 20 c2 c9
1   40 20 f0 08 58 55 28 d0  08 4c 42 1c 15 69 2c 00
2   12 31 60 aa 5a a0 10 f8  c4 56 01 02 0a 9f fb 70
3   13 18 04 84 2a 98 b8 04  52 41 1d 0a 94 08 2a 54
4   a8 04 7a 01 2b 33 f8 03  c0 16
DECOMPRESSED DATA:
Decompression took: 153.784 milliseconds
Decompressed data size: 132 bytes
0   ff ff ff ff 00 22 00 04  08 08 00 01 00 01 ff ff
1   ff 92 00 07 00 0b ff ff  ff ff 00 46 00 09 10 0e
2   00 01 00 01 ff ff ff 25  00 0e 00 16 ff ff ff ff
3   00 6a 00 0f 18 15 00 02  00 01 ff ff fe b8 00 16
4   00 21 ff ff ff ff 00 8b  00 14 20 1d 00 02 00 02
5   ff ff fe 4a 00 1e 00 2b  00 00 00 00 00 00 00 00
6   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
7   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8   00 00 00 00
CRC32 CHECKSUM:
Average checksum generation time: 0.067 milliseconds
0   d4 7e 8f e6
```

Figure 8 – STM2 CLI output for the under-sampling experiment

STM1 was told to sample 132 bytes of sensor data,. However, it was forced only to sample 88 bytes of data to simulate under-sampling.

The table below shows a comparison between the results from the under-sampling experiment and the results from section 3.1. for compression.

	Uncompressed data size [bytes]	**Compressed data size [bytes]**	**Compression time [ms]**	**Decompression time [ms]**
Original	132	103	189.000	155.795
Under-sampled	132	74	139.111	153.784

Table XIII – Under-sampling experiment compression results

As seen from the table above, the under-sampling caused the compressed data to reduce the compressed data size due to more values in the buffer being null. The under-sampling also decreased the compression time because more values were null. However, the decompression remained essentially constant because the algorithm still had to expand the data to 132 bytes.

The table below shows a comparison between the results from the under-sampling experiment and the results from section 3.1. for encryption.

	Decrypted data size [bytes]	**Encrypted data size [bytes]**	**Encryption time [ms]**	**Decryption time [ms]**
Original	103	104	118.489	117.606
Under sampled	74	80	93.338	85.690

Table XIV – Under-sampling experiment encryption results

Since the size of both the decrypted data and encrypted data changed, there should be a change in the encryption and decryption time. This behaviour was observed as shown in the table above.

High Sampling Rate Experiment

The CLI output for sampling sensor data at a rate of one sample per 100 milliseconds is shown below.

```
SENSOR DATA:
Sensor data size: 132 bytes
0   ff ff ff ff 00 25 00 04  08 08 00 02 00 01 ff ff
1   ff 93 00 08 00 0c ff ff  ff ff 00 49 00 08 10 0e
2   00 02 00 01 ff ff ff 27  00 10 00 17 ff ff ff ff
3   00 6c 00 0c 18 14 00 03  00 01 00 00 fe bc 00 17
4   00 23 ff ff ff ff 00 8f  00 12 20 18 00 04 00 01
5   00 00 fe 4f 00 1f 00 2e  ff ff ff ff 00 b3 00 19
6   28 1e 00 04 00 01 00 00  fd e3 00 26 00 39 ff ff
7   ff ff 00 d5 00 1f 30 29  00 04 00 01 00 00 fd 76
8   00 2e 00 44
CRC32 CHECKSUM:
0   af 6e 0d 2e
COMPRESSED DATA:
Compression took: 185.24 milliseconds
Compressed data size: 98 bytes
0   ff 80 0a 01 25 80 41 21  10 88 04 0a 01 01 06 16
1   4e 00 09 18 60 55 29 20  91 88 43 82 ad 27 80 44
2   20 11 70 aa 5b 03 a3 18  8a 40 20 61 51 1d 8f fb
3   78 13 18 04 8c 2a 98 f8  04 4a 41 18 22 88 55 29
4   f0 08 fc 02 5c 15 4d 9c  02 33 28 8f 05 57 fb e3
5   80 49 a0 13 90 aa 75 43  e3 30 94 85 5a ec 29 18
6   05 10
ENCRYPTED DATA:
Encryption took: 118.252 milliseconds
Encrypted data size: 104 bytes
0   66 77 e2 5e 55 a3 5e f2  30 33 51 ca c6 8b cb 61
1   ec b6 1e 0c 3f 6b 7c 82  4f bb 49 50 82 b1 98 9d
2   ce 4e b9 b8 db 01 57 fc  16 7b d2 07 1e 60 31 9b
3   a3 a6 e2 e2 3d 2d 3f b8  1d e9 0a 1b e3 bd d9 ea
4   6c cd eb f2 1d 63 db db  f6 16 dd 80 8b ed 08 47
5   cc d2 49 e4 ce 79 c8 db  fb 9d 3c d9 58 7e 03 6e
6   44 57 2f 82 61 cb 31 9a
COMPRESSED DATA SIZE
0   62 00 00 00
PADDING LENGTH:
0   06
```

Figure 11 – STM1 CLI output for increased sensor sampling rate

```
Decrypt and decompress? (y/n): y
PADDING LENGTH CONFIRMATION:
0   06
COMPRESSED DATA SIZE CONFIRMATION:
0   62 00 00 00
ENCRYPTED DATA CONFIRMATION:
0   66 77 e2 5e 55 a3 5e f2  30 33 51 ca c6 8b cb 61
1   ec b6 1e 0c 3f 6b 7c 82  4f bb 49 50 82 b1 98 9d
2   ce 4e b9 b8 db 01 57 fc  16 7b d2 07 1e 60 31 9b
3   a3 a6 e2 e2 3d 2d 3f b8  1d e9 0a 1b e3 bd d9 ea
4   6c cd eb f2 1d 63 db db  f6 16 dd 80 8b ed 08 47
5   cc d2 49 e4 ce 79 c8 db  fb 9d 3c d9 58 7e 03 6e
6   44 57 2f 82 61 cb 31 9a
DECRYPTED DATA:
Decryption took: 112.211 milliseconds
Decrypted data size: 98 bytes
0   ff 80 0a 01 25 80 41 21  10 88 04 0a 01 01 06 16
1   4e 00 09 18 60 55 29 20  91 88 43 82 ad 27 80 44
2   20 11 70 aa 5b 03 a3 18  8a 40 20 61 51 1d 8f fb
3   78 13 18 04 8c 2a 98 f8  04 4a 41 18 22 88 55 29
4   f0 08 fc 02 5c 15 4d 9c  02 33 28 8f 05 57 fb e3
5   80 49 a0 13 90 aa 75 43  e3 30 94 85 5a ec 29 18
6   05 10
DECOMPRESSED DATA:
Decompression took: 157.531 milliseconds
Decompressed data size: 132 bytes
0   ff ff ff ff 00 25 00 04  08 08 00 02 00 01 ff ff
1   ff 93 00 08 00 0c ff ff  ff ff 00 49 00 08 10 0e
2   00 02 00 01 ff ff ff 27  00 10 00 17 ff ff ff ff
3   00 6c 00 0c 18 14 00 03  00 01 00 00 fe bc 00 17
4   00 23 ff ff ff ff 00 8f  00 12 20 18 00 04 00 01
5   00 00 fe 4f 00 1f 00 2e  ff ff ff ff 00 b3 00 19
6   28 1e 00 04 00 01 00 00  fd e3 00 26 00 39 ff ff
7   ff ff 00 d5 00 1f 30 29  00 04 00 01 00 00 fd 76
8   00 2e 00 44
CRC32 CHECKSUM:
Average checksum generation time: 0.067 milliseconds
0   af 6e 0d 2e
```

Figure 10 – STM2 CLI output for increased sensor sampling rate

The table below compares the results from the increased sampling rate experiment and the results from section 3.1. for compression.

	Uncompressed data size [bytes]	Compressed data size [bytes]	Compression time [ms]	Decompression time [ms]
Original	132	103	189.000	155.795
Fast sampling	132	98	185.240	157.531

Table XV – Increased sampling rate experiment compression results

As seen from the table above, sampling the sensor data at an increased rate decreased the compressed data size from 103 to 98 bytes. This means that more repetitive sequences in the sensor data were found. The compression and decompression took essentially the same length of time to run to completion.

The table below compares the results from the increased sampling rate experiment and the results from section 3.1. for encryption.

	Decrypted data size [bytes]	Encrypted data size [bytes]	Encryption time [ms]	Decryption time [ms]
Original	103	104	118.489	117.606
Under sampled	98	104	118.252	112.211

Table XVI – Increased sampling rate experiment encryption results

As seen from the table above, the increased sampling rate did not significantly affect the encryption algorithm.

Gaussian Noise Experiment

The CLI output for adding Gaussian noise to the sensor data is shown below.

```
SENSOR DATA:
Sensor data size: 132 bytes
0   00 00 ff 00 00 22 00 03  08 08 01 02 01 01 ff ff
1   ff 99 01 08 01 0e ff ff  ff 00 01 46 00 0a 11 10
2   00 01 01 00 ff fe ff 31  01 0f 00 18 ff 00 00 00
3   01 6c 00 10 18 16 01 02  01 00 00 fe fe c8 01 14
4   00 25 ff 00 ff 00 01 8e  01 19 21 1b 01 02 00 01
5   ff fe ff 5f 01 1c 01 32  ff 00 00 ff 01 b0 00 1e
6   29 20 01 02 01 00 ff fe  fd f4 00 22 00 3f 00 ff
7   00 ff 01 d3 00 23 30 26  01 02 01 01 00 fe fd 89
8   00 28 01 4b
CRC32 CHECKSUM:
0   df 33 ed 7f
COMPRESSED DATA:
Compression took: 198.368 milliseconds
Compressed data size: 120 bytes
0   00 0f fc 04 32 28 04 0e  11 08 80 c0 a0 30 1f f8
1   00 73 30 10 48 c3 80 e5  00 80 d1 a0 10 a8 8c 40
2   0c 30 11 08 ff bf f3 18  0c 3e 01 18 14 90 38 6d
3   90 08 84 62 2c 2b 20 50  ff bf dc 88 0c 52 01 25
4   0a 88 ac b1 d0 18 cc 86  36 15 10 40 8a ca bf 01
5   8e 40 66 42 b2 ff c0 76  10 08 f4 a6 40 2b 30 a8
6   ff 7e 86 52 9f 8a 88 2a  3d 38 04 8e 61 26 36 99
7   04 7f b8 98 04 a2 03 4b
ENCRYPTED DATA:
Encryption took: 136.115 milliseconds
Encrypted data size: 120 bytes
0   74 f4 eb 60 60 40 33 60  c9 7d 34 9f cb 0f 02 5c
1   2c 87 d9 43 a6 01 90 ca  af 5d fc 21 ca 53 be 99
2   c3 d0 2a 83 ba f4 0c 29  7d ff 34 82 73 84 d3 d3
3   19 6b 4b 84 9b b2 6b 6d  01 cb 3c 89 44 4d 4a f3
4   a1 32 90 32 52 ad a9 f9  34 7a d1 53 0d 3d 92 f0
5   f6 37 1a 9b 1b 53 68 ce  b3 d1 bb 69 93 b5 2f 36
6   05 e6 5b 1a 33 d6 7f 0d  cc 80 3f fb 0c 21 a1 ef
7   46 7a ff 42 d5 f9 d4 1d
COMPRESSED DATA SIZE
0   78 00 00 00
PADDING LENGTH:
0   00
```

Figure 13 – STM1 CLI output for gaussian noise experiment

```
Decrypt and decompress? (y/n): y
PADDING LENGTH CONFIRMATION:
0   00
COMPRESSED DATA SIZE CONFIRMATION:
0   78 00 00 00
ENCRYPTED DATA CONFIRMATION:
0   74 f4 eb 60 60 40 33 60  c9 7d 34 9f cb 0f 02 5c
1   2c 87 d9 43 a6 01 90 ca  af 5d fc 21 ca 53 be 99
2   c3 d0 2a 83 ba f4 0c 29  7d ff 34 82 73 84 d3 d3
3   19 6b 4b 84 9b b2 6b 6d  01 cb 3c 89 44 4d 4a f3
4   a1 32 90 32 52 ad a9 f9  34 7a d1 53 0d 3d 92 f0
5   f6 37 1a 9b 1b 53 68 ce  b3 d1 bb 69 93 b5 2f 36
6   05 e6 5b 1a 33 d6 7f 0d  cc 80 3f fb 0c 21 a1 ef
7   46 7a ff 42 d5 f9 d4 1d
DECRYPTED DATA:
Decryption took: 134.875 milliseconds
Decrypted data size: 120 bytes
0   00 0f fc 04 32 28 04 0e  11 08 80 c0 a0 30 1f f8
1   00 73 30 10 48 c3 80 e5  00 80 d1 a0 10 a8 8c 40
2   0c 30 11 08 ff bf f3 18  0c 3e 01 18 14 90 38 6d
3   90 08 84 62 2c 2b 20 50  ff bf dc 88 0c 52 01 25
4   0a 88 ac b1 d0 18 cc 86  36 15 10 40 8a ca bf 01
5   8e 40 66 42 b2 ff c0 76  10 08 f4 a6 40 2b 30 a8
6   ff 7e 86 52 9f 8a 88 2a  3d 38 04 8e 61 26 36 99
7   04 7f b8 98 04 a2 03 4b
DECOMPRESSED DATA:
Decompression took: 159.722 milliseconds
Decompressed data size: 132 bytes
0   00 00 ff 00 00 22 00 03  08 08 01 02 01 01 ff ff
1   ff 99 01 08 01 0e ff ff  ff 00 01 46 00 0a 11 10
2   00 01 01 00 ff fe ff 31  01 0f 00 18 ff 00 00 00
3   01 6c 00 10 18 16 01 02  01 00 00 fe fe c8 01 14
4   00 25 ff 00 ff 00 01 8e  01 19 21 1b 01 02 00 01
5   ff fe ff 5f 01 1c 01 32  ff 00 00 ff 01 b0 00 1e
6   29 20 01 02 01 00 ff fe  fd f4 00 22 00 3f 00 ff
7   00 ff 01 d3 00 23 30 26  01 02 01 01 00 fe fd 89
8   00 28 01 4b
CRC32 CHECKSUM:
Average checksum generation time: 0.067 milliseconds
0   df 33 ed 7f
```

Figure 12 – STM2 CLI output for gaussian noise experiment

The table below compares the results from the gaussian noise experiment and the results from section 3.1. for compression.

	Uncompressed data size [bytes]	Compressed data size [bytes]	Compression time [ms]	Decompression time [ms]
Original	132	103	189.000	155.795
Gaussian Noise	132	120	198.368	159.722

Table XVII – Gaussian noise experiment compression results

As seen from the table above, adding Gaussian noise to the sensor data increased the compressed data size from 103 to 120 bytes and slightly increased the compression time and decompression time. This is because compression looks for repetitive values while compressing to reduce the overall size of the data. Since noise is added to the input data set, fewer values will be repetitive. Hence the algorithm will not be as effective as one with less or no noise in the sensor data.

The table below compares the results from the gaussian noise experiment and the results from section 3.1. for encryption.

	Decrypted data size [bytes]	Encrypted data size [bytes]	Encryption time [ms]	Decryption time [ms]
Original	103	104	118.489	117.606
Gaussian Noise	120	120	136.115	134.875

Table XVIII – Gaussian noise experiment encryption results

As the table above shows, adding Gaussian noise to the sensor data increased the encryption and decryption computation times. The Gaussian noise affected the substitution in the S-boxes; the algorithm took longer to process the data blocks since the data was more random. The same effect occurred with decryption, as the algorithm took longer to process the substituted blocks of data. The compressed size of the data set also increased; this, in turn, also affects the algorithm's speed.

4. Practical Data Acceptance Test Procedures

4.1. Compression ATPs

ATP	Description	ATP achieved	Comment
Compression time	The compression algorithm must not take more than 5 seconds per 10 kilobytes of data. (2 kilobytes per second)	NO	The decompression and compression fails to meet the ATP requirement, this is shown by one of the calculations done to verify the working of the compression algorithm.
Data loss	No data must be lost during the compression of the data file.	YES	The checksum indicates that the entire system is lossless and therefore the compression block must be lossless.
Compression effectiveness	The compression ratio of the original file size and the compressed file size must be at least 1.5.	YES	As indicated in table VII, the compression was at least 1.5 or higher when the rounds were greater than 3.

Table XIX – Practical data ATPs for the compression block

Compression Time Calculation:

To decompress 1012 bytes, it will take 1122.388ms. This indicates that it will take 2.218s to decompress 2 kilobytes of data – this fails to meet the ATP.

4.2. Encryption ATPs

ATP	Description	ATP achieved	Comment
Encryption time	The encryption and decryption execution time should not exceed 10 seconds per 10 kilobytes of data.	NO	The encryption and decryption took longer than what was outlined by the ATP requirement.
Data loss	No data must be lost during the encryption and decryption of the data file.	YES	The checksum indicates that the entire system is lossless and therefore the encryption block must be lossless.
Encryption security	The encryption must be strong enough to prevent decryption.	YES	Since the key is static and known only by the client and the STMs. It will only be possible for the client to be able to decrypt.
Encryption integrity	The original data in the file must be identical to the decrypted data in the output file.	YES	The checksum was used to verify the data received by the client following decryption is identical to the data collected by the STM.

Table XX – Practical data ATPs for the encryption block

4.3. Checksum ATPs

ATP	Description	ATP achieved	Comment
STM checksum	The STM must create a checksum of the original sensor data successfully.	YES	As can be seen in section 3.4., the checksum algorithm successfully generates a checksum on the first STM.
Client checksum	The client's STM must successfully create a checksum of the decrypted and decompressed data.	YES	As can be seen in section 3.4., the checksum algorithm successfully generates a checksum on the client's STM.
Checksum comparison	The two STMs must generate identical checksums.	YES	As can be seen by the results shown in section 3.1. and 3.4., the checksums are identical

Table XXI - Practical data ATPs for the checksum block

4.4 New Specifications

Compression ATP

ATP	Old Version	New Version
Compression time	The compression algorithm must not take more than 5 seconds per 10 kilobytes of data. (2 kilobytes per second)	The compression algorithm must not take longer than 1 second to execute on half a kilobyte of data.

Table XXII – new specification for compression algorithm vs old specification

Encryption ATP

ATP	Old Version	New Version
Encryption time	The encryption and decryption execution time should not exceed 10 seconds per 10 kilobytes of data.	The encryption algorithm must not take longer than 1 second to execute on half a kilobyte of data.

Table XXIII – new specification for encryption algorithm vs old specification

References

There are no sources in the current document.