ELEC-C7420 Basic Principles in Networking
Part II: Security

Assignment I: Introduction to basic cryptography using Arduino

Members: Beatriz Glaser
Thuy Määttä

1. Goals of the experiment

   The goal of this experiment is to create a basic encryption sketch that can convert a readable text message into an encrypted message and convert an encrypted message into a readable text message. The encryption is implemented on an Arduino MKR Wifi 1010 board and should provide some level of security to the transmitted messages, we chose to use base64 as our algorithm. The experiment demonstrates how to use the Arduino board for basic encryption tasks and provides a starting point for future practical applications where secure communication is essential.

2. Experimental setup

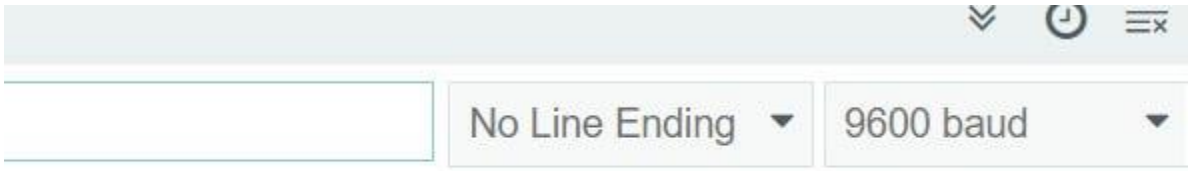
   For this experiment, we used the following equipment:
   - Arduino MKR WiFi 1010 board
   - Micro USB cable
   - Arduino IDE
   - Base64 library (by Densaugeo)

   We took the following steps to set up the experiment:
   1. Download and install the Arduino IDE
   2. Install SAMD in the Arduino IDE to support the MKR Wifi 1010 board
   3. Select/Connect the correct board and port in the IDE
   4. Create a sketch
      4.1 Import the library
      4.2 Write a “setup” function to establish serial communication between Arduino and the computer via a USB cable
      4.3 Implement code in a “loop” function
   5. Compile code and upload the sketch to the board
   6. Test with the Serial Monitor
      6.1 Open Serial Monitor
      6.2 Change the configuration from ‘New line’ to ‘No line Editing’
      6.3 Follow the instructions given by the program.

3. Results and Conclusion

3.1. Results

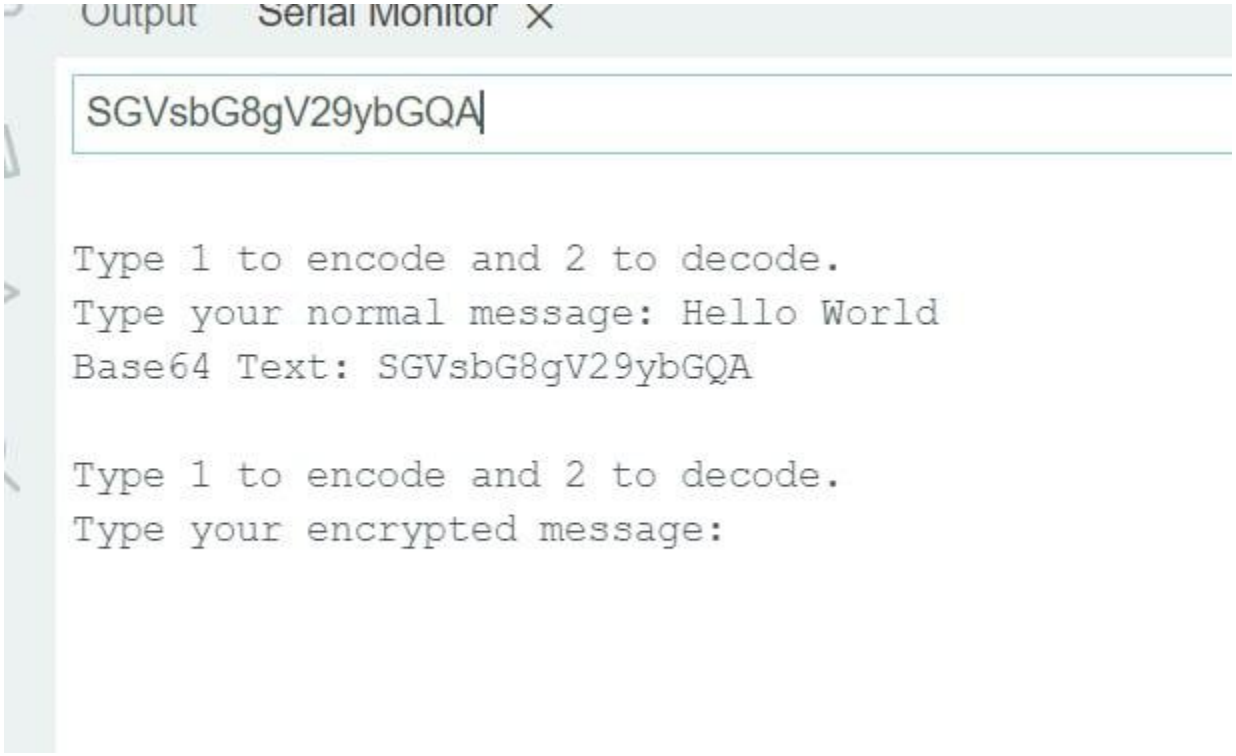
1. Make sure to change the Serial Monitor from ‘New line’ to ‘No Line Editing’



2. Type 1 to encode and type in the message to be encrypted, ‘Hello World’ in this test

3. The program successfully encodes the message and shows the encrypted text

4. Type 2 to decode and type the encrypted message



5. The program successfully decodes the message and shows the result

6. Error handling: if the input is different than 1 or 2, the program will print ‘Incorrect command’ until a correct one is entered.

3.2. Conclusion

In general, this experiment provides a basic understanding of encryption techniques and how they can be implemented on an Arduino board. Besides, encryption is a necessary tool to protect sensitive information and maintain data privacy and security. It is also important to learn and understand the syntax, after gaining a comprehensive understanding of the syntax, the implementation of the program becomes more manageable.

4. Bonus

4.1 Explanation

4.1.1 Creating keys

- The user enters two large prime numbers x and y, which are used to compute the modulus n = x * y.
- The totient function of n, denoted by ø(n), is calculated as ø(n) = (x-1) * (y-1).
- An integer ‘e’ is chosen such that it is coprime to ø(n) (i.e., the greatest common divisor of e and ø(n) is 1) and 1 < e < ø(n). This value of e is the public key exponent used for encryption.
- A value ‘d’ is chosen such that d * e mod ø(n) = 1, which means that d is the multiplicative inverse of e in mod ø(n). This value of d is the private key exponent used for decryption.

- The public key consists of the pair {e, n}, and the private key consists of the pair {d, n}.

4.1.2 Encrypting message

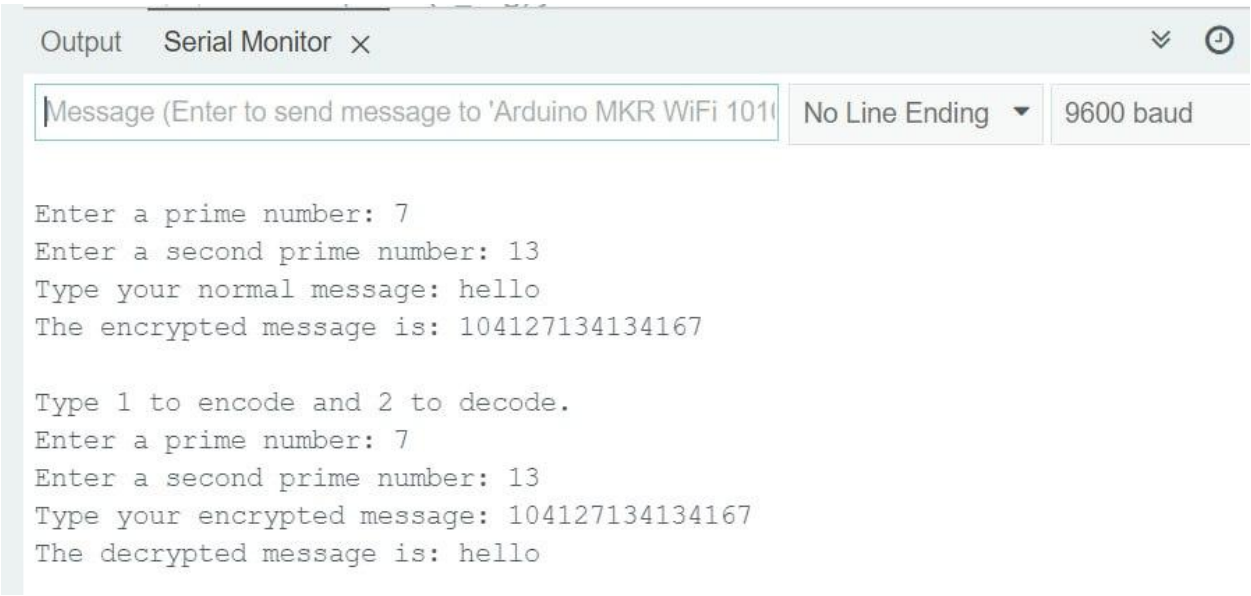
- To encrypt a message M, the program uses the public key {e, n}.
- The program computes the ciphertext C as C = M ^ e mod n, where ^ denotes exponentiation.
- The program prints out the ciphertext C.

4.1.3 Decrypting message

- To decrypt the ciphertext C, the program uses the private key {d, n}.
- The program computes the plaintext message M as M = C ^ d mod n.
- The program now prints out the original message M.

4.2 Result

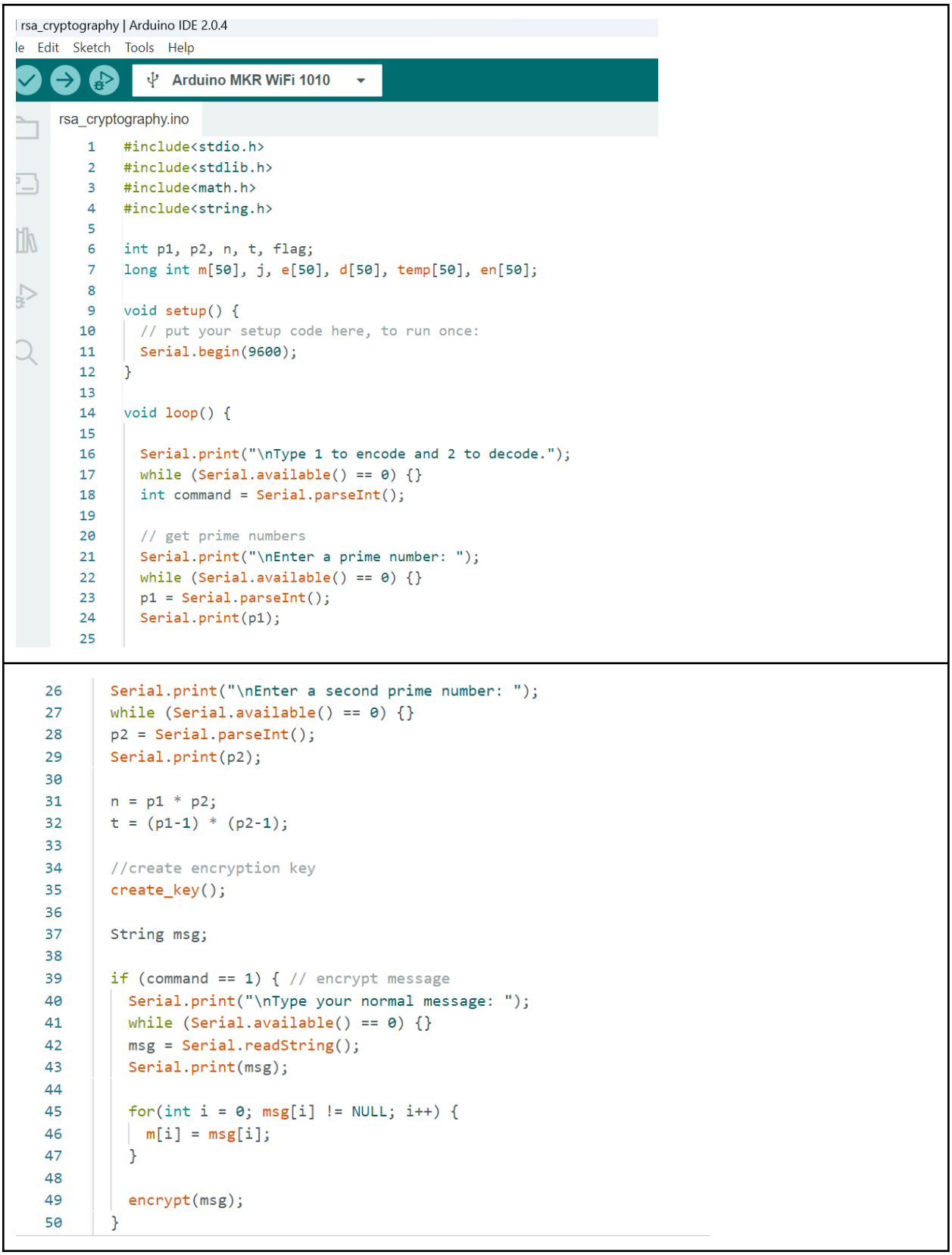
The program successfully encrypts the message and decrypts it back to the original message.



5. Annex

5.1 Source code for coding/decoding with base64.

5.2 Source code for RSA implementation.




```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>

int p1, p2, n, t, flag;
long int m[50], j, e[50], d[50], temp[50], en[50];

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {

  Serial.print("\nType 1 to encode and 2 to decode.");
  while (Serial.available() == 0) {}
  int command = Serial.parseInt();

  // get prime numbers
  Serial.print("\nEnter a prime number: ");
  while (Serial.available() == 0) {}
  p1 = Serial.parseInt();
  Serial.print(p1);

  Serial.print("\nEnter a second prime number: ");
  while (Serial.available() == 0) {}
  p2 = Serial.parseInt();
  Serial.print(p2);

  n = p1 * p2;
  t = (p1-1) * (p2-1);

  //create encryption key
  create_key();

  String msg;

  if (command == 1) { // encrypt message
    Serial.print("\nType your normal message: ");
    while (Serial.available() == 0) {}
    msg = Serial.readString();
    Serial.print(msg);

    for(int i = 0; msg[i] != NULL; i++) {
      m[i] = msg[i];
    }

    encrypt(msg);
  }
```

```
  else if (command == 2) { // decrypt message
    Serial.print("\nType your encrypted message: ");
    while (Serial.available() == 0) {}
    String e_msg = Serial.readString();
    Serial.print(e_msg);

    for(int i = 0; e_msg[i] != NULL; i++) {
      m[i] = e_msg[i];
    }

    decrypt(e_msg);
  }
  else {
    Serial.print("\nIncorrect command.");
  }
}

int is_prime(long int num) {
  int i;
  j = sqrt(num);
  for(i = 2; i <= j; i++)
  {
   if(num % i == 0)
    return 0;
  }
```

```
  return 1;
}

long int cd(long int num) {
  long int k = 1;
  while(1)
  {
    k = k + t;
    if(k % num == 0)
      return(k / num);
  }
}

void create_key() {
  int x = 0;

  for (int i = 2; i < t; i++) {
    if (t % i == 0) {
      continue;
    }

    flag = is_prime(i);
    if (flag == 1 && i != p1 && i != p2) {
        e[x] = i;
        flag = cd(e[x]);
```

```
            if (flag > 0) {
              d[x] = flag;
              x++;
            }

            if (x == 99) break;
          }
        }
      }

      void encrypt(String msg) {

        long int p, c, key = e[0], x;
        int i = 0;
        long int len = strlen(msg.c_str());

        while (i != len) {
          p = m[i];
          p = p - 96;
          x = 1;

          for (j = 0; j < key; j++) {
            x = x * p;
            x = x % n;
          }

          temp[i] = x;
          c = x + 96;
          en[i] = c;
          i++;
        }

        en[i] = -1;

        Serial.print("\nThe encrypted message is: ");
        for (i = 0; en[i] != -1; i++) {
          Serial.print(en[i]);
        }
        Serial.print("\n");
      }

      void decrypt(String msg) {

        long int p, c, key = d[0], x;
        int i = 0;

        while (en[i] != -1) {
          c = temp[i];
          x = 1;
```

```
151
152       for (j = 0; j < key; j++) {
153         x = x * c;
154         x = x % n;
155       }
156
157       p = x + 96;
158       m[i] = p;
159       i++;
160     }
161
162     m[i] = -1;
163
164     Serial.print("\nThe decrypted message is: ");
165     for (i = 0; m[i] != -1; i++) {
166       Serial.write(m[i]);
167     }
168     Serial.print("\n");
169   }
170
```

6. Reference

RSA algorithm:
http://www.trytoprogram.com/c-examples/c-program-to-encrypt-and-decrypt-string/

Base64 library:
https://github.com/Densaugeo/base64_arduino