

CORSO DI
NETWORK SECURITY

Università degli Studi di Brescia
Dipartimento di Ingegneria
dell'Informazione
Via Branze, 38
25231 – Brescia –



Comunicazione cifrata tra iPhone e Arduino

Relazione del Progetto di Network Security

GRUPPO:

Nome: Sansoni Davide
Matricola: 86042

Nome: Trivella Emanuele
Matricola: 86505

Sommario

| | |
|--|----------|
| Capitolo 1: Introduzione..... | 3 |
| 1.1 Obiettivi del progetto | 3 |
| 1.1.1 Software utilizzato | 3 |
| 1.1.2 Hardware utilizzato | 3 |
| Capitolo 2: Preparazione ambiente..... | 4 |
| 2.1 Ambienti di sviluppo | 4 |
| 2.1.1 Xcode..... | 4 |
| 2.1.2 Arduino..... | 4 |
| 2.2 Librerie | 5 |
| 2.2.1 Bignumbr | 5 |
| 2.2.2 Collegamento BLE..... | 6 |
| 2.2.3 AES..... | 6 |
| Capitolo 3: L'applicazione iOS e la contro parte Arduino..... | 7 |
| 3.1 iOS | 7 |
| 3.1.1 Collegamento con Arduino | 8 |
| 3.1.2 Generazione dei parametri di Diffie-Hellman | 8 |
| 3.1.3 Calcolo della chiave Diffie-Hellman..... | 8 |
| 3.1.4 Invio e ricezione di messaggi cifrati tramite AES..... | 9 |
| 3.2 Arduino | 10 |
| 3.2.1 Collegamento con iPhone | 10 |
| 3.2.2 Generazione dei parametri e calcolo chiave Diffie-Hellman | 10 |
| 3.2.3 Invio e ricezione di messaggi cifrati tramite AES..... | 10 |

Capitolo 1: Introduzione

1.1 Obiettivi del progetto

Il progetto si pone come obiettivo la realizzazione di un sistema per lo scambio di messaggi cifrato tra un iPhone ed un Arduino.

I messaggi scambiati sono protetti da chiave simmetrica¹ calcolata da entrambe le parti con il protocollo Diffie-Hellman².

1.1.1 Software utilizzato

L'implementazione del progetto è divisa in due parti principali: una realizzata specificatamente per iPhone e l'altra per Arduino.

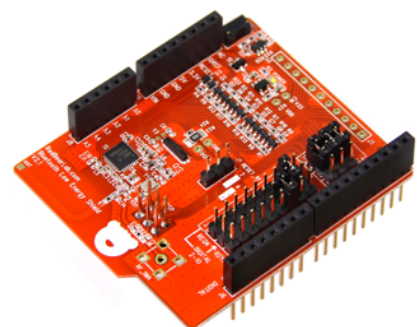
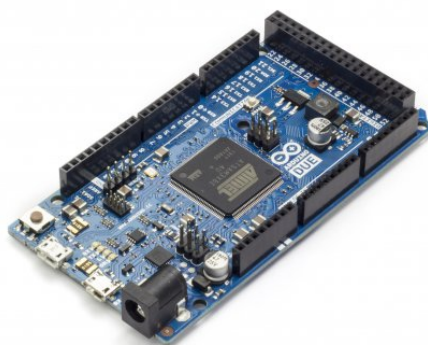
Per quanto riguarda lo smartphone, il codice è stato scritto in Objective-C con il supporto dell'ambiente di sviluppo Xcode³.

Per il microcontrollore, invece, è stato utilizzato l'IDE Arduino⁴ ed il linguaggio creato ad-hoc per questo dispositivo, che presenta molte analogie con il C.

1.1.2 Hardware utilizzato

I dispositivi fisici utilizzati in questo progetto sono tre:

- Apple iPhone 6 con iOS 9⁵
- Arduino Due con processore 32bit ARM⁶
- RedBearLab BLE Shield 2.1⁷



¹ https://en.wikipedia.org/wiki/Symmetric-key_algorithm

² https://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange

³ <https://developer.apple.com/xcode/>

⁴ <https://www.arduino.cc>

⁵ <http://www.apple.com/it/ios/whats-new/>

⁶ <https://www.arduino.cc/en/Main/ArduinoBoardDue>

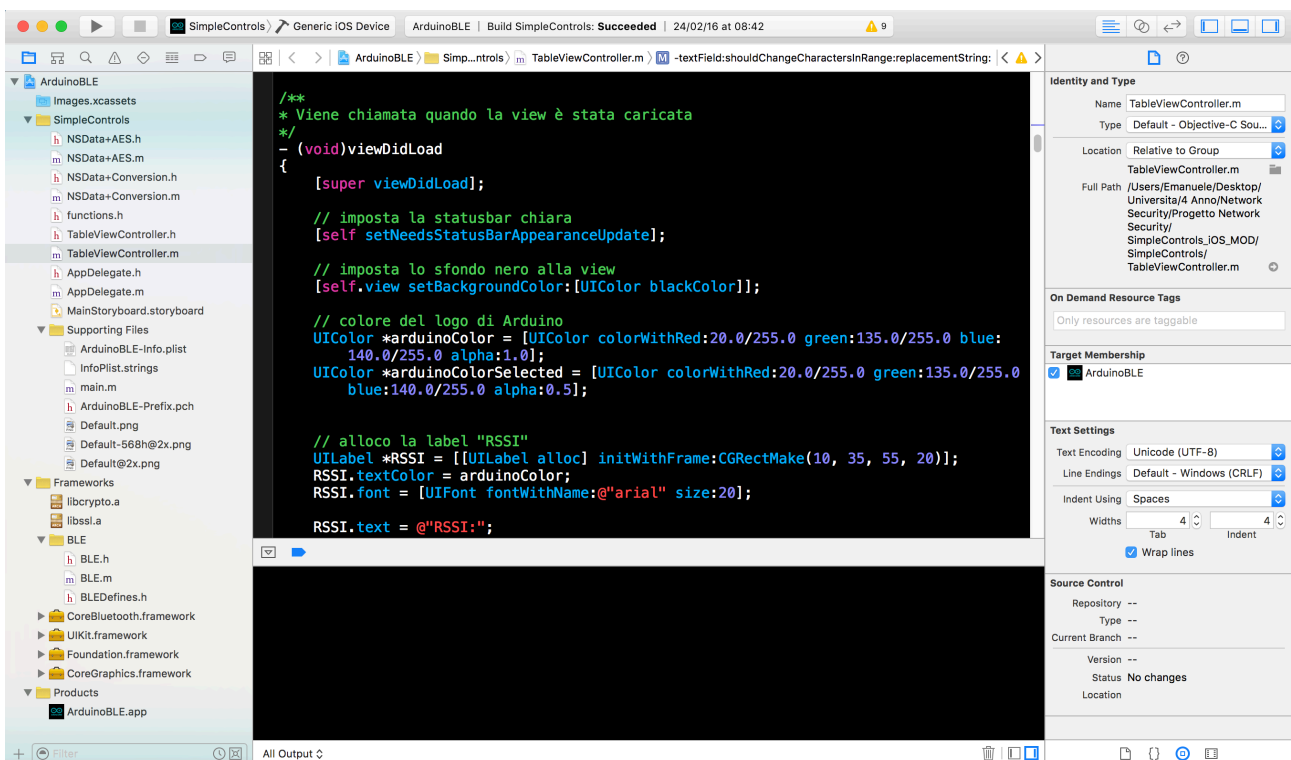
⁷ <http://redbearlab.com/bleshield/>

Capitolo 2: Preparazione ambiente

2.1 Ambienti di sviluppo

2.1.1 Xcode

Xcode è un ambiente di sviluppo integrato (IDE) contenente una suite di strumenti utili allo sviluppo di software principalmente per i sistemi OSX e iOS. È completamente sviluppato e mantenuto da Apple ed è possibile scaricarlo gratuitamente dal Mac App Store. Il linguaggio utilizzato è Objective-C sia per i sistemi desktop che per quelli mobile, ma per questi ultimi è disponibile anche il linguaggio open source Swift.

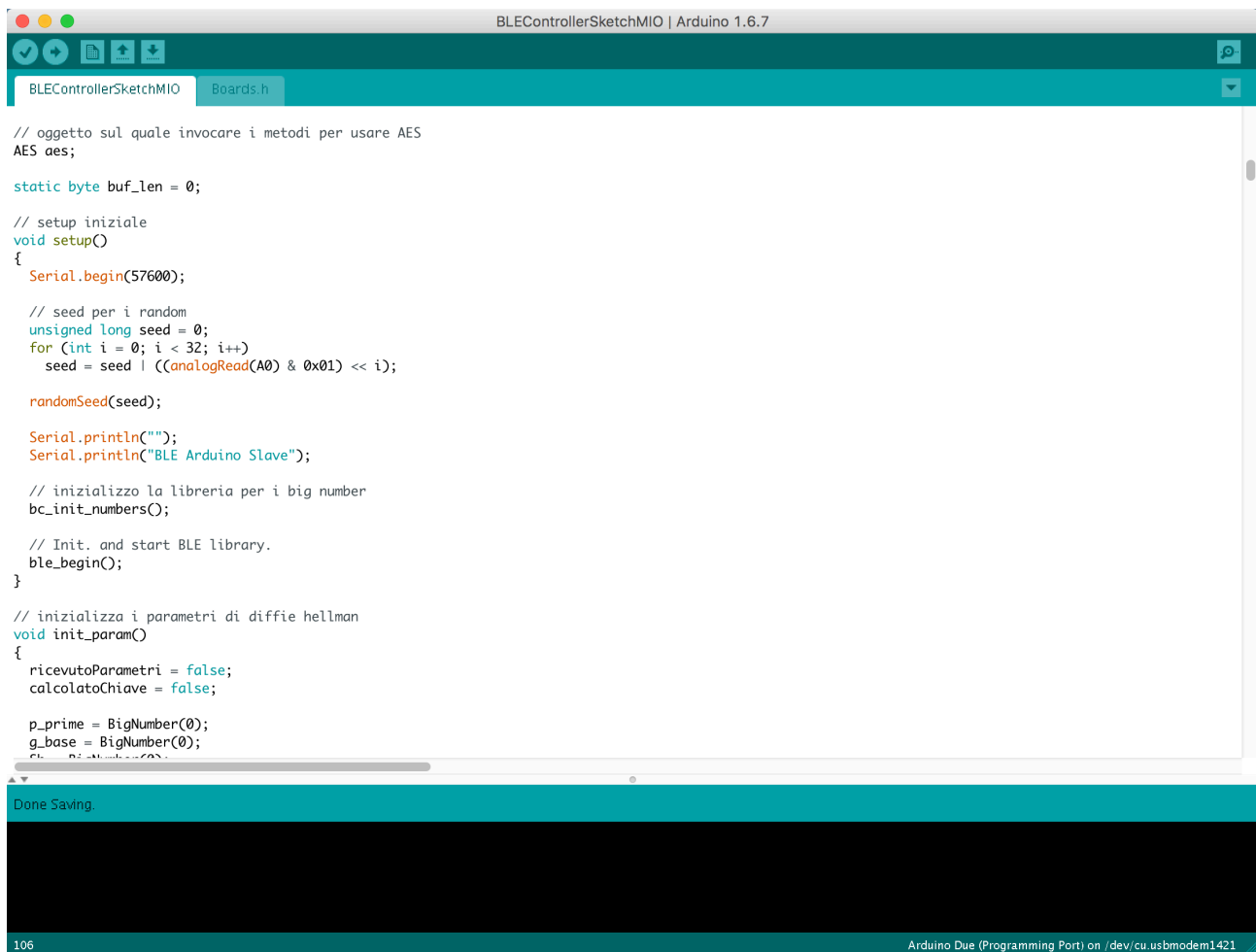


2.1.2 Arduino

L'ambiente di sviluppo integrato (IDE) di Arduino è un'applicazione multiplatforma in javascript, ed è derivata dall'IDE creato per il linguaggio di programmazione Processing e per il progetto Wiring.

L'editor è in grado di compilare e lanciare il programma eseguibile in una sola passata e con un solo click senza bisogno di creare alcun makefile.

L'ambiente di sviluppo integrato di Arduino è fornito di una libreria software C/C++ chiamata "Wiring": la disponibilità della libreria rende molto più semplice implementare via software le comuni operazioni di input/output.



```

BLEControllerSketchMIO | Arduino 1.6.7

// oggetto sul quale invocare i metodi per usare AES
AES aes;

static byte buf_len = 0;

// setup iniziale
void setup()
{
  Serial.begin(57600);

  // seed per i random
  unsigned long seed = 0;
  for (int i = 0; i < 32; i++)
    seed = seed | ((analogRead(A0) & 0x01) << i);

  randomSeed(seed);

  Serial.println("");
  Serial.println("BLE Arduino Slave");

  // inizializzo la libreria per i big number
  bc_init_numbers();

  // Init. and start BLE library.
  ble_begin();
}

// inizializza i parametri di diffie hellman
void init_param()
{
  ricevutoParametri = false;
  calcolatoChiave = false;

  p_prime = BigNumber(0);
  g_base = BigNumber(0);
}

```

Done Saving.

106 Arduino Due (Programming Port) on /dev/cu.usbmodem1421

2.2 Librerie

In questo capitolo verranno presentate le librerie necessarie per la corretta generazione dei parametri Diffie-Hellman e per la comunicazione cifrata con AES tra iPhone e Arduino.

2.2.1 BigNumber

Le librerie BigNumber sono utilizzate per gestire numeri più grandi dei tipi di dato predefiniti messi a disposizione dagli IDE (*int* e *long*). Per questo motivo abbiamo dovuto appoggiarci ad esse per la generazione della chiave Diffie-Hellman su entrambi i dispositivi.

Per quanto riguarda l'iPhone, i BigNumber sono definiti nella libreria OpenSSL⁸ e si possono istanziare con la keyword **BIGNUM**.

```

// Ta = (g^Sa) modp
BIGNUM *Ta;

// mandato da B (Arduino) = (g^Sb) modp
BIGNUM *Tb;

```

⁸ <https://github.com/st3fan/ios-openssl>

L'Arduino, invece, sfrutta un'altra libreria che è stata sviluppata da Nick Gammon⁹. Un esempio di utilizzo di BigNumber è qui sotto riportato:

```
// Diffie Helman param
BigNumber p_prime;
BigNumber g_base;
BigNumber Sb;

// iPhone
BigNumber Ta;

// Arduino
BigNumber Tb;

// DH KEY
BigNumber dh_key;
```

2.2.2 Collegamento BLE

Per quanto riguarda il collegamento Bluetooth tra i due dispositivi abbiamo utilizzato la libreria BLE^[10,11] disponibile sul sito di RedbearLab¹².

Inoltre per interfacciare Arduino Due con la BLE Shield è stato necessario installare la libreria RBL_nRF8001¹³.

Le librerie Arduino fornite da RedbearLab sono sicuramente funzionanti con Arduino Uno, mentre non è garantita la piena compatibilità con la versione Due del controllore. Per questo motivo, qualora dovessero verificarsi dei problemi durante la compilazione, mettiamo a disposizione i file da sostituire in entrambe le librerie¹⁴.

2.2.3 AES

Infine, per implementare la cifratura dei messaggi scambiati, abbiamo installato la libreria AES su Arduino¹⁵, mentre su iPhone ci siamo appoggiati sulla libreria di sistema *CommonCryptor.h*.

⁹ <http://forum.arduino.cc/index.php?topic=85692.0>

¹⁰ Arduino BLE SDK: <https://github.com/cheong2k/ble-sdk-arduino>

¹¹ iOS BLE Library: <https://github.com/RedBearLab/iOS>

¹² Repository RedbearLab Libraries: <https://github.com/RedBearLab>

¹³ <https://github.com/RedBearLab/nRF8001>

¹⁴ https://github.com/dsansoni/NetSecProject/blob/master/Arduino-Project/Libraries/BLE_DUE_Compatible.zip

¹⁵ Arduino AES Library: <http://utter.chaos.org.uk/~markt/AES-library.zip>

Capitolo 3: L'applicazione iOS e la contro parte Arduino

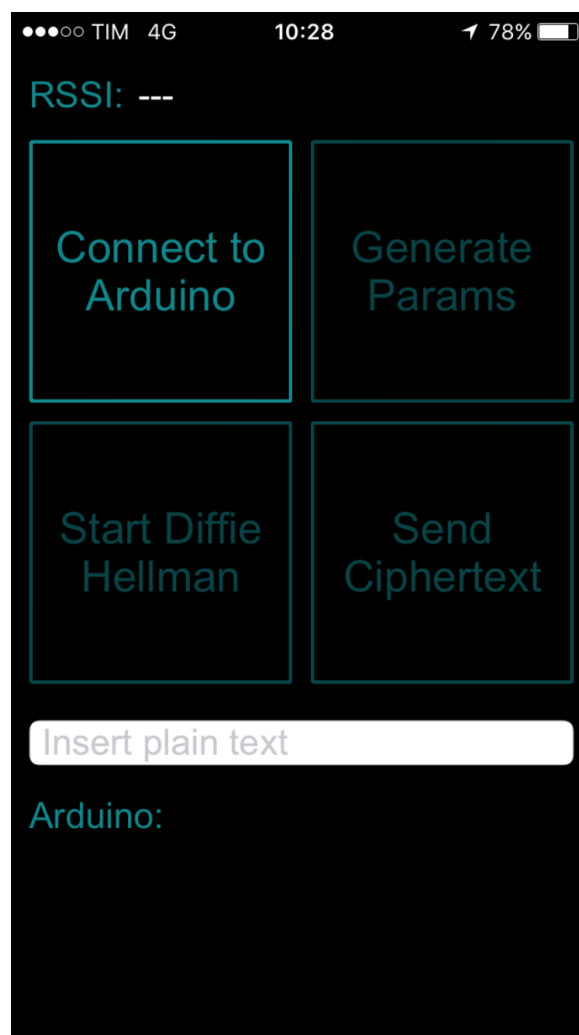
3.1 iOS

L'applicazione realizzata per iPhone ha un'interfaccia semplice e intuitiva: presenta quattro bottoni principali, un campo di testo modificabile e uno non modificabile.

I quattro bottoni sono utili per connettersi all'Arduino tramite tecnologia Bluetooth, per generare i parametri necessari al corretto funzionamento di Diffie-Hellman, per avviare il protocollo di derivazione della chiave e, infine, per inviare un messaggio cifrato con la chiave appena calcolata.

Nel campo di testo modificabile è possibile inserire del testo e successivamente inviare il cipher corrispondente all'Arduino, tramite il bottone apposito.

Il campo di testo non modificabile, invece, è stato previsto per visualizzare sullo schermo dell'iPhone la risposta dell'Arduino.



3.1.1 Collegamento con Arduino

Il collegamento tra i due dispositivi viene avviato facendo click sul bottone “Connect to Arduino”. Non appena cliccato, l’iPhone inizia la ricerca di una periferica BLE¹⁶ nelle vicinanze tramite il metodo *btnScanForPeripherals*. Nel dettaglio, questo metodo avvia un timeout entro il quale il dispositivo iOS deve trovare un accessorio BLE disponibile al collegamento e connettersi. Se l’intera procedura di connessione è andata a buon fine, l’iPhone e l’Arduino possono iniziare a comunicare tramite Bluetooth e scambiarsi le informazioni preliminari per avviare il protocollo Diffie-Hellman.

3.1.2 Generazione dei parametri di Diffie-Hellman

Una volta connessi i dispositivi, viene abilitato dall’applicazione il bottone “Generate Params” che permette la generazione dei parametri necessari al corretto funzionamento di Diffie-Hellman tramite il metodo *generateDHParams*.

Il primo parametro è il numero primo p calcolato con il metodo di libreria *BN_generate_prime_ex()* su 128 bit.

In seguito viene estratto un numero random Sa compreso tra 1 e p .

Quest’ultimo viene utilizzato per calcolare il numero Ta che verrà in seguito inviato dall’iPhone all’Arduino. Ta è calcolato elevando la base $g = DH_GENERATOR_5$ (costante di libreria) all’esponente Sa , il tutto modulo p . Ossia $Ta = g^{Sa} \bmod p$.

L’Arduino, come vedremo dopo, utilizzerà Ta per derivare la chiave simmetrica.

3.1.3 Calcolo della chiave Diffie-Hellman

Dopo aver assegnato tutti i parametri, viene abilitato il bottone “Start Diffie Hellman” che procede con l’invio degli stessi all’Arduino cui l’iPhone è connesso, sfruttando il metodo *startDH*. Ogni parametro è inviato con un messaggio distinto (metodo *sendPkt*), in particolare abbiamo optato per l’invio di quattro messaggi:

1. inizializzazione
2. parametro p
3. parametro g
4. parametro Ta

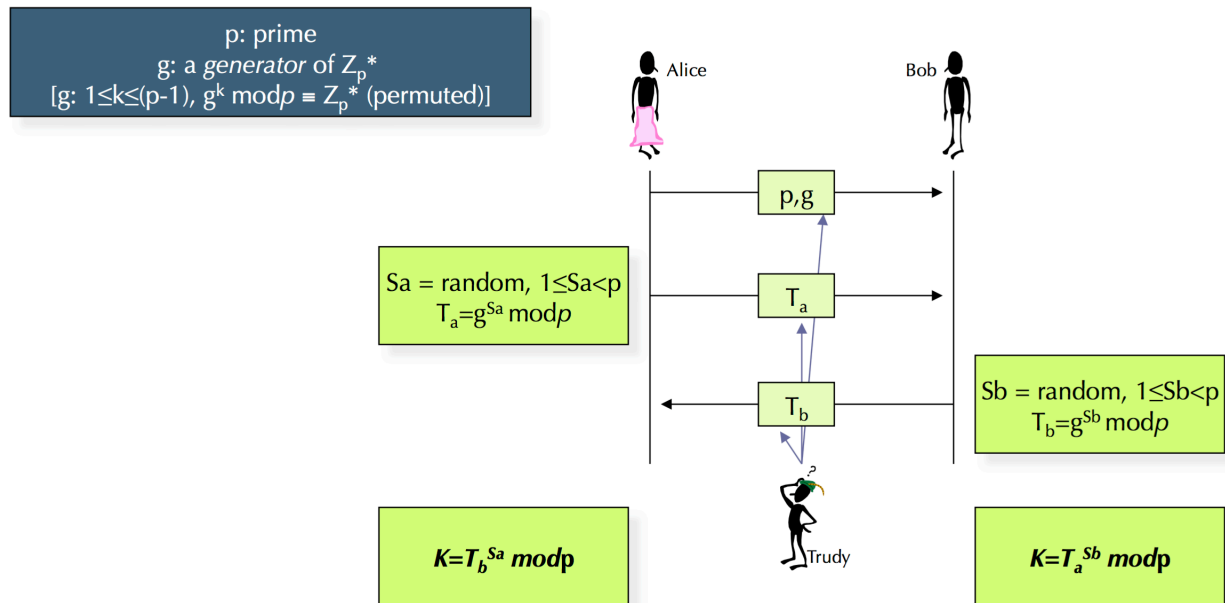
Ciascun messaggio è strutturato nella forma seguente:

| CMD | LML | MSG L. | Message (Payload) |
|--------|--------|----------|-------------------|
| 1 byte | 1 byte | LML byte | MSG L. byte |

¹⁶ Controllare che il service UUID sia 713D0000-503E-4C75-BA94-3148F18D941E

Dove CMD è il comando, ovvero un carattere identificativo del messaggio inviato¹⁷, PAYLOAD è il contenuto del messaggio, MSG Length è la lunghezza del payload e LML è la lunghezza del campo MSG Length.

Ricevuti i parametri, l'Arduino risponde all'iPhone inviando il proprio T_b . A questo punto è possibile per entrambi i dispositivi calcolare la chiave da utilizzare per cifrare i messaggi: $key = T_b^{S_a} \bmod p$ nel caso dell'iPhone, e $key = T_a^{S_b} \bmod p$ nel caso dell'Arduino.



3.1.4 Invio e ricezione di messaggi cifrati tramite AES

L'ultima fase è quella che prevede l'invio di un messaggio cifrato con AES dall'iPhone all'Arduino. Questo è possibile solamente se la chiave è stata precedentemente calcolata su entrambi i dispositivi.

Utilizzando la tastiera dello smartphone è possibile inserire un massimo di quindici caratteri e inviarli cifrati tramite il metodo *sendCipher*. Quest'ultimo non fa altro che cifrare con AES (metodo *AES128EncryptedDataWithKey*) il testo inserito dall'utente per poi inviarlo sfruttando il metodo *sendPkt* già utilizzato in precedenza per inviare i parametri Diffie-Hellman.

Una volta ricevuto e decifrato il messaggio, l'Arduino invia all'iPhone una risposta cifrata anch'essa con AES che viene ricevuta e decifrata dallo smartphone (metodo *AES128DecryptedDataWithKey*) e in seguito mostrata a video nel campo di testo non modificabile.

¹⁷ I possibili CMD sono: 'I' per l'inizializzazione, 'P' per l'invio del numero primo p , 'G' per l'invio della base g , 'T' per l'invio di T_a e 'M' per l'invio del messaggio cifrato.

3.2 Arduino

L'applicazione realizzata per Arduino non ha un'interfaccia grafica e ha il compito di ricevere i messaggi inviati dall'iPhone. In base al comando contenuto nei pacchetti ricevuti, il controllore esegue un'opportuna elaborazione del dato con conseguente invio della risposta al dispositivo iOS.

E' di particolare importanza evidenziare che l'Arduino riceve i pacchetti inviati dall'iPhone in modalità seriale, ovvero può leggere un singolo byte alla volta¹⁸. La funzione di libreria adibita a questo compito è *ble_read*. Essa, inoltre, si occupa della gestione dei byte persi nella comunicazione.

Per riuscire a leggere una sequenza di più byte, abbiamo realizzato la funzione *mergeHexString* che, a partire da *ble_read*, concatena i singoli dati in un'unica stringa in notazione esadecimale.

3.2.1 Collegamento con iPhone

Il collegamento tra i due dispositivi può essere avviato solamente dall'iPhone cliccando sul bottone "*Connect to Arduino*". Non appena cliccato, l'iPhone inizia la ricerca di una periferica BLE disponibile nelle vicinanze e successivamente stabilisce una connessione con essa.

3.2.2 Generazione dei parametri e calcolo chiave Diffie-Hellman

Il principale attore della generazione dei parametri Diffie-Hellman è l'iPhone. Esso, infatti, genera il numero primo p , la base g e li invia all'Arduino insieme al Ta ¹⁹. Il controllore, dopo aver ricevuto i tre parametri, deve estrarre un numero random Sb compreso tra 1 e p (funzione *randomCharArray*), calcolare il proprio $Tb = g^{Sb} \bmod p$ e successivamente inviarlo allo smartphone (funzione *sendPkt*). Solo a questo punto gli è possibile calcolare la chiave Diffie-Hellman utilizzabile per cifrare i messaggi successivi come $key = Ta^{Sb} \bmod p$.

3.2.3 Invio e ricezione di messaggi cifrati tramite AES

L'iPhone può inviare un messaggio di al più 15 caratteri all'Arduino. Una volta ricevuto, esso viene convertito in un array di byte e poi decifrato tramite la funzione *decrypt* presente nella libreria AES. In seguito l'Arduino invia una risposta cifrata all'iPhone che verrà mostrata sul display dello smartphone. Per semplicità questa è la stringa fissa "risposta ok".

¹⁸ Il buffer di lettura dell'Arduino ha capienza massima di 20 byte

¹⁹ Si veda il capitolo 3.1.2