



Data Encryption on SArduino

조진성

경희대학교 컴퓨터공학과

Mobile & Embedded System Lab.



Computer Engineering in KyungHee University

Mobile & Embedded System Lab.

SArduino: Secure Arduino

❖ RTOS/Firmware 기반 저사양 COTS IoT 디바이스 보안 플랫폼

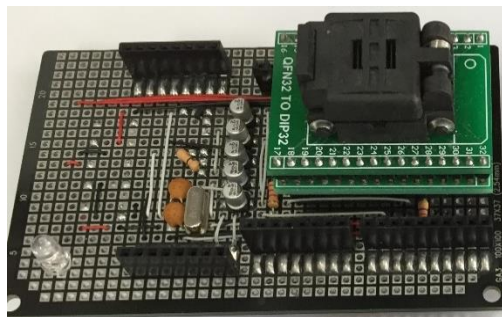
■ Arduino + Infineon OPTIGA Trust P

- Secure Key Storage & Management
- Secure Boot
- Secure Firmware Update
- Remote Attestation
- Secure Communication
- File Encryption

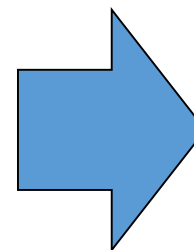


Insecure Arduino

+



SE

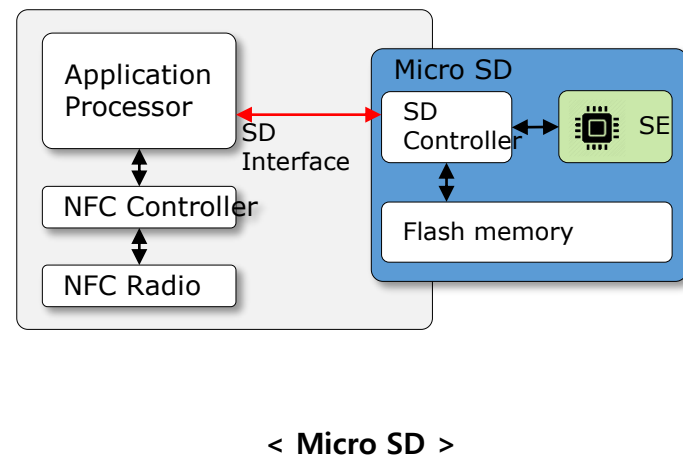
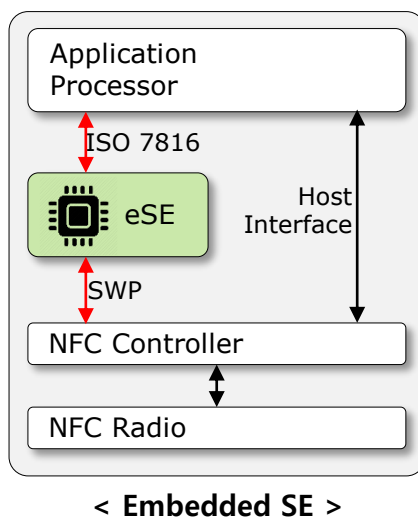
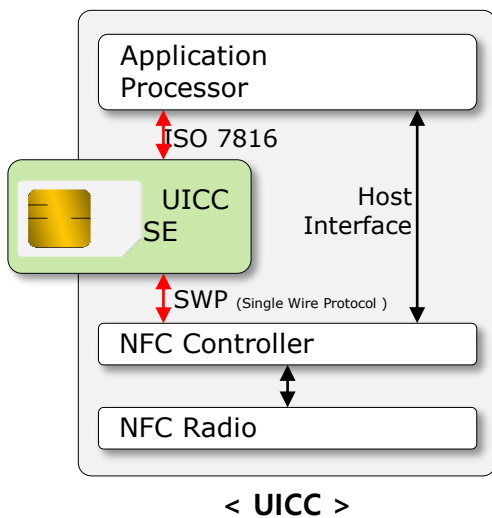


SECURE
플랫폼
(SArduino)



❖ SE (Secure Element)

- 모바일 서비스를 위해 인증 및 서비스 안전성을 지원하는 장치
- GlobalPlatform 표준
- 주요기능
 - 보안 관련 애플릿 호스팅 (e.g. 암호/복호화, 디바이스 인증)
 - 안전한 데이터 저장 (e.g. 암호화 키, 기밀 데이터)
- 다양한 버전의 SE 지원
 - UICC (Universal Integrated Circuit Card), Embedded SE, Micro SD



SArduino: Secure Arduino

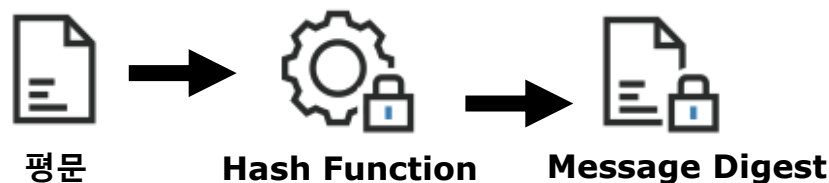
❖ Arduino와 SE의 통신

■ ISO 7816

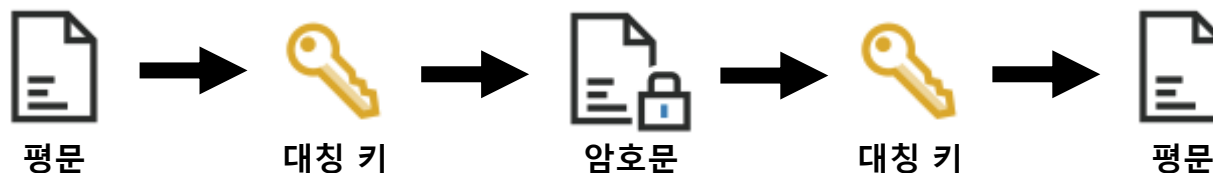
- Command APDU – Arduino에서 SE로 전달되는 명령 수행 메시지
- Response APDU – SE에서 Arduino로 전달되는 명령 응답 메시지

❖ SArduino API

■ SHA (Secure Hash Algorithm)



■ AES (Advanced Encryption Standard)



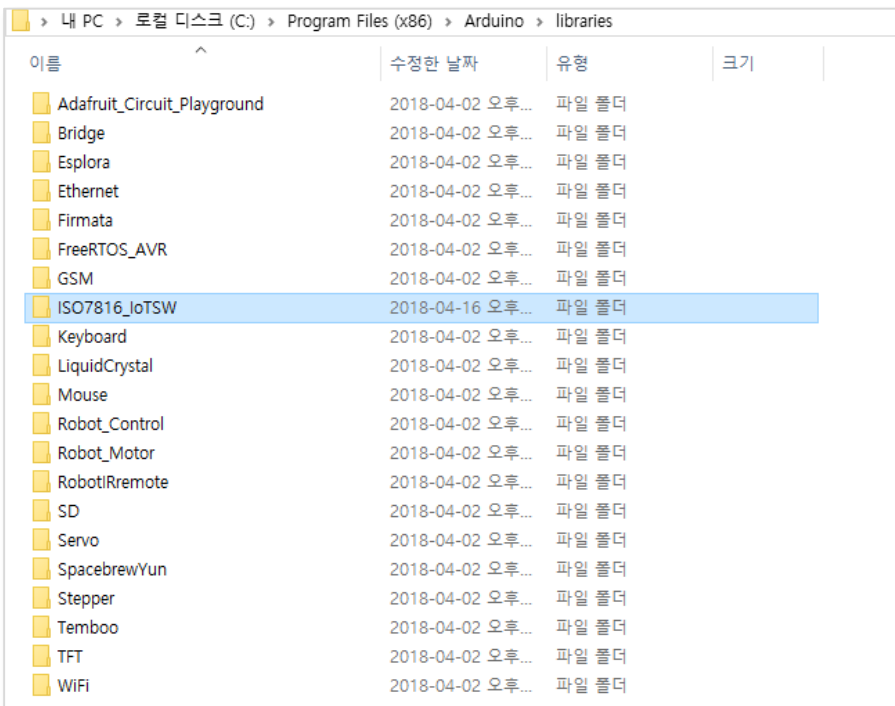
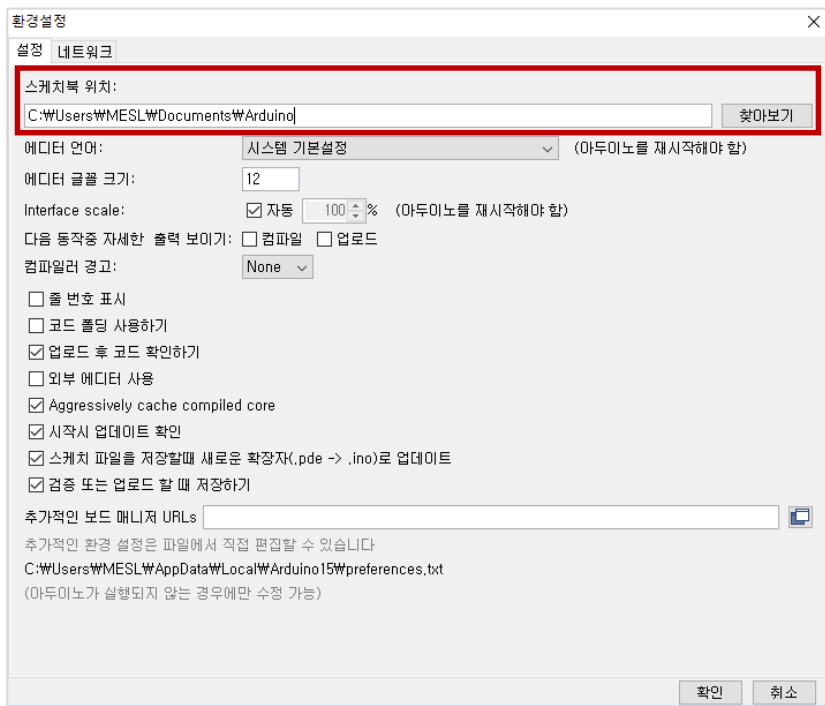
■ RSA (Rivest, Shamir, and Adleman)



개발환경 구축

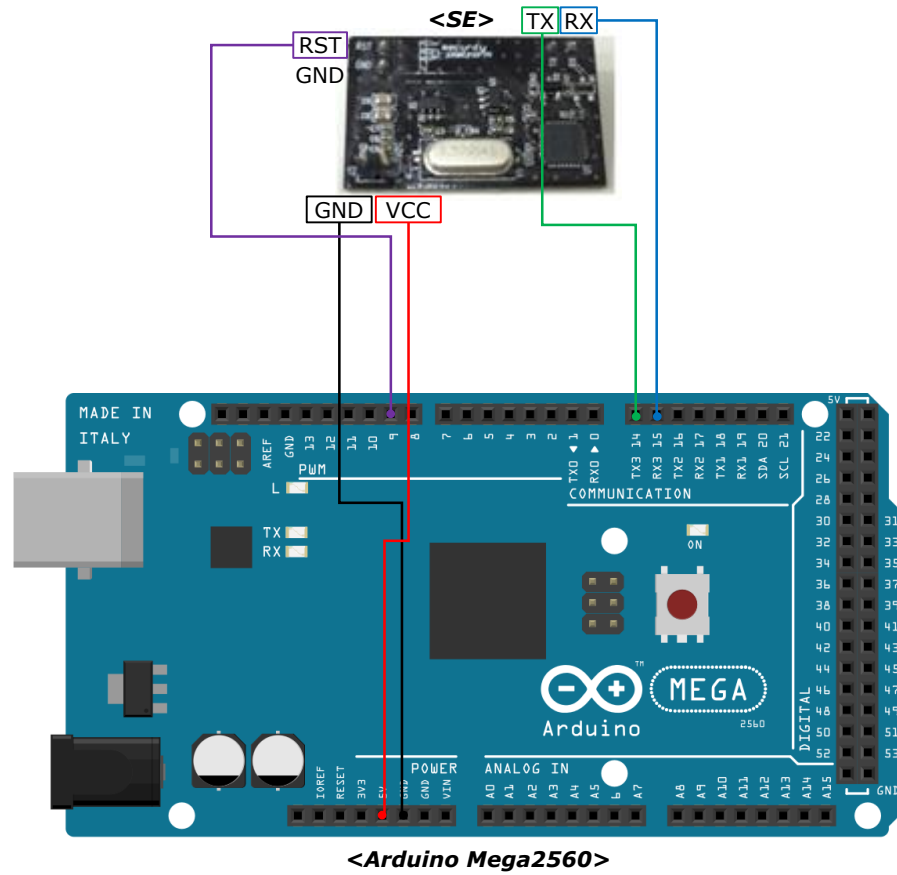
❖ Arduino Mega2560 & SE

- 강의 홈페이지에서 “ISO7816_IoTSW.zip” 파일 다운로드
 - 강의홈페이지 > 설치자료 > “Data Encryption Library on SArduino”
- 다운로드 받은 파일을 압축 해제 후, 폴더 전체를 아두이노 라이브러리에 복사
 - 아두이노 라이브러리: “아두이노 IDE > 파일 > 환경설정” 에서 확인 가능



하드웨어 구성

❖ Arduino Mega2560 & SE





❖ SArduino.h

- `bool Init_SE();`
 - Return
 - true: SE Connection Success
 - false: SE Connection Failure

- `int SHA_256(BYTE* plain_data, int plain_len, byte* digest, int* digest_len);`

- `int Generate_AES128Key(int key_num)`
- `int Encrypt_AES128(int key_num, byte* plain_data, int plain_len, byte* enc_data, int* enc_len);`
- `int Decrypt_AES128(int key_num, byte* enc_data, int enc_len, byte* plain_data, int* plain_len);`

- `int Generate_RSA1024Key(int key_num)`
- `int Encrypt_RSA1024(int key_num, int key_type, byte* plain_data, int plain_len, byte* enc_data, int* enc_len);`
- `int Decrypt_RSA1024(int key_num, int key_type, byte* enc_data, int enc_len, byte* plain_data, int* plain_len);`

SHA-256



❖ API

- `int SHA_256(BYTE* plain_data, int plain_len, byte* digest, int* digest_len);`
 - Return true on success, false on failure
 - `int digest_len`: SHA-256의 결과는 32byte로 나옴

SHA-256

❖ Source code

```
#include <SArduino.h>

// buf에 저장된 데이터를 시리얼 모니터로 출력하는 함수
void dump( byte* buf, int len ) {
    int i;

    for( i = 0; i < len; i++ ) {
        Serial.print( (char)buf[ i ] );
    }
    Serial.println();
}

void setup() {
    // 시리얼 모니터 사용
    Serial.begin( 9600, SERIAL_8E2 );

    // SE 초기화
    if( !Init_SE() ) {
        Serial.println("SE Connection Failure");
    }
}
```

```
byte plain_data[ ] = "Hello IoT Software!";
int plain_len = strlen( plain_data );

byte digest[ 32 ];
int digest_len = 32;

Serial.print("plain data: ");
dump( plain_data, plain_len );

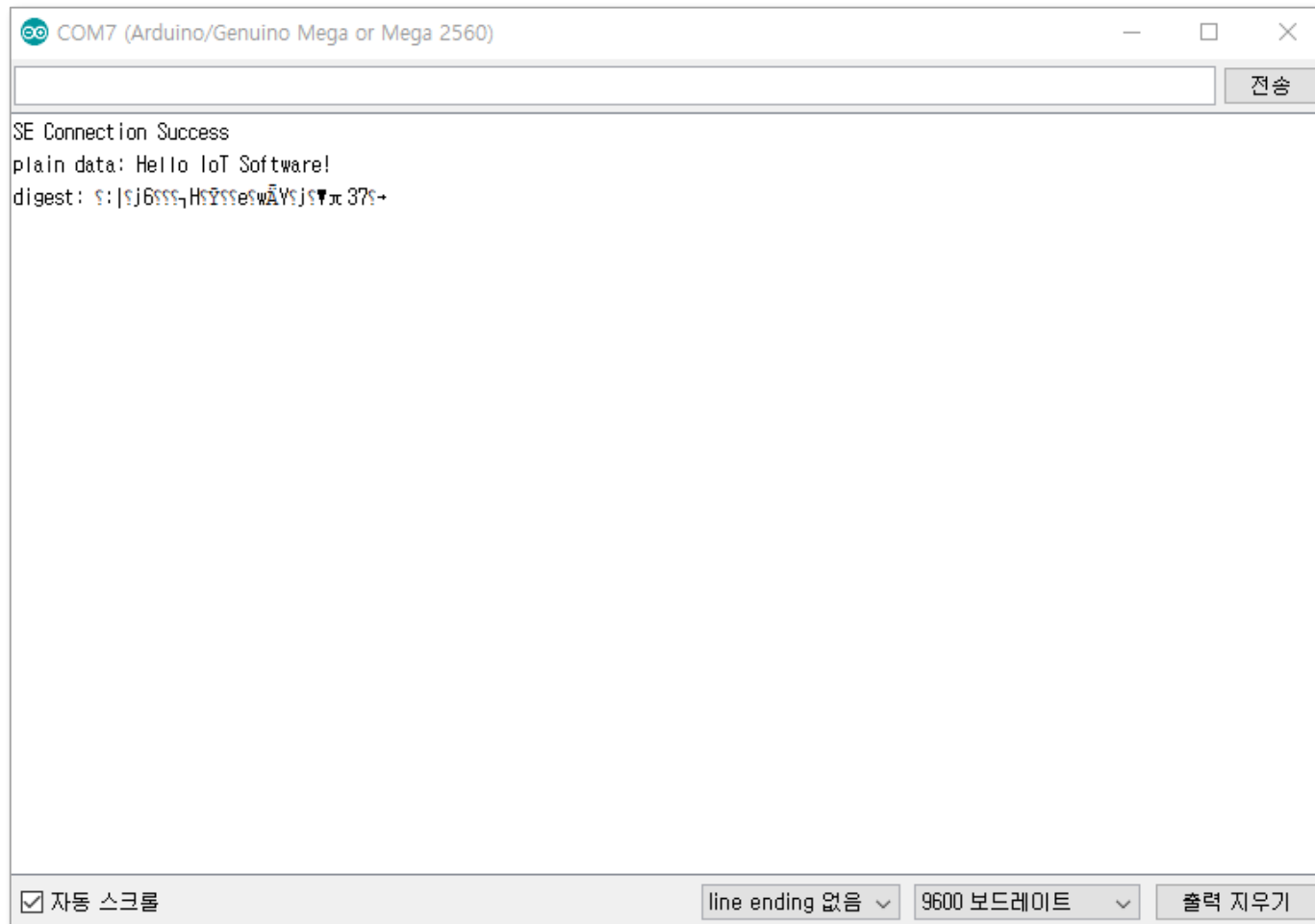
if( SHA_256( plain_data, plain_len, digest, &digest_len ) ) {
    Serial.print("digest: ");
    dump(digest, digest_len);
}
else {
    Serial.println("SHA_256 Failure");
}

void loop() {
}
```

SHA-256

❖ 실행 결과

- 시리얼 모니터로 확인(업로드 완료 후, 7~10초 가량 소요)





❖ API

- `int Generate_AES128Key(int key_num)`
 - Return true on success, false on failure
 - int key_num: 0x0~0x1F

- `int Encrypt_AES128(int key_num, byte* plain_data, int plain_len, byte* enc_data, int* enc_len);`
 - Return true on success, false on failure
 - int key_num: Generate_AES128Key(...)에서 사용한 key_num
 - int enc_len: 16의 배수

- `int Decrypt_AES128(int key_num, byte* enc_data, int enc_len, byte* dec_data, int* dec_len);`
 - Return true on success, false on failure
 - int key_num: Generate_AES128Key(...)에서 사용한 key_num
 - int enc_len: 16의 배수

AES-128

❖ Source code

```
#include <SArduino.h>

// buf에 저장된 데이터를 시리얼 모니터로 출력하는 함수
void dump( byte* buf, int len ) {
    int i;

    for( i = 0; i < len; i++ ) {
        Serial.print( (char)buf[ i ] );
    }
    Serial.println();
}

void setup() {
    // 시리얼 모니터 사용
    Serial.begin( 9600, SERIAL_8E2 );

    // SE 초기화
    if( !Init_SE() ) {
        Serial.println("SE Connection Failure");
    }
}
```

```
int key_num = 0x0;

byte plain_data[ ] = "Hello IoT Software!";
int plain_len = strlen( plain_data );
plain_len = ((plain_len / 16) + 1) * 16;
// 16의 배수이어야 동작

byte enc_data[ 64 ];
int enc_len;

byte dec_data[ 64 ];
int dec_len;

if( !Generate_AES128Key(key_num) )
    Serial.println("Set AES128 Key Generation Failure");

Serial.print("plain data: ");
dump( plain_data, plain_len );
```



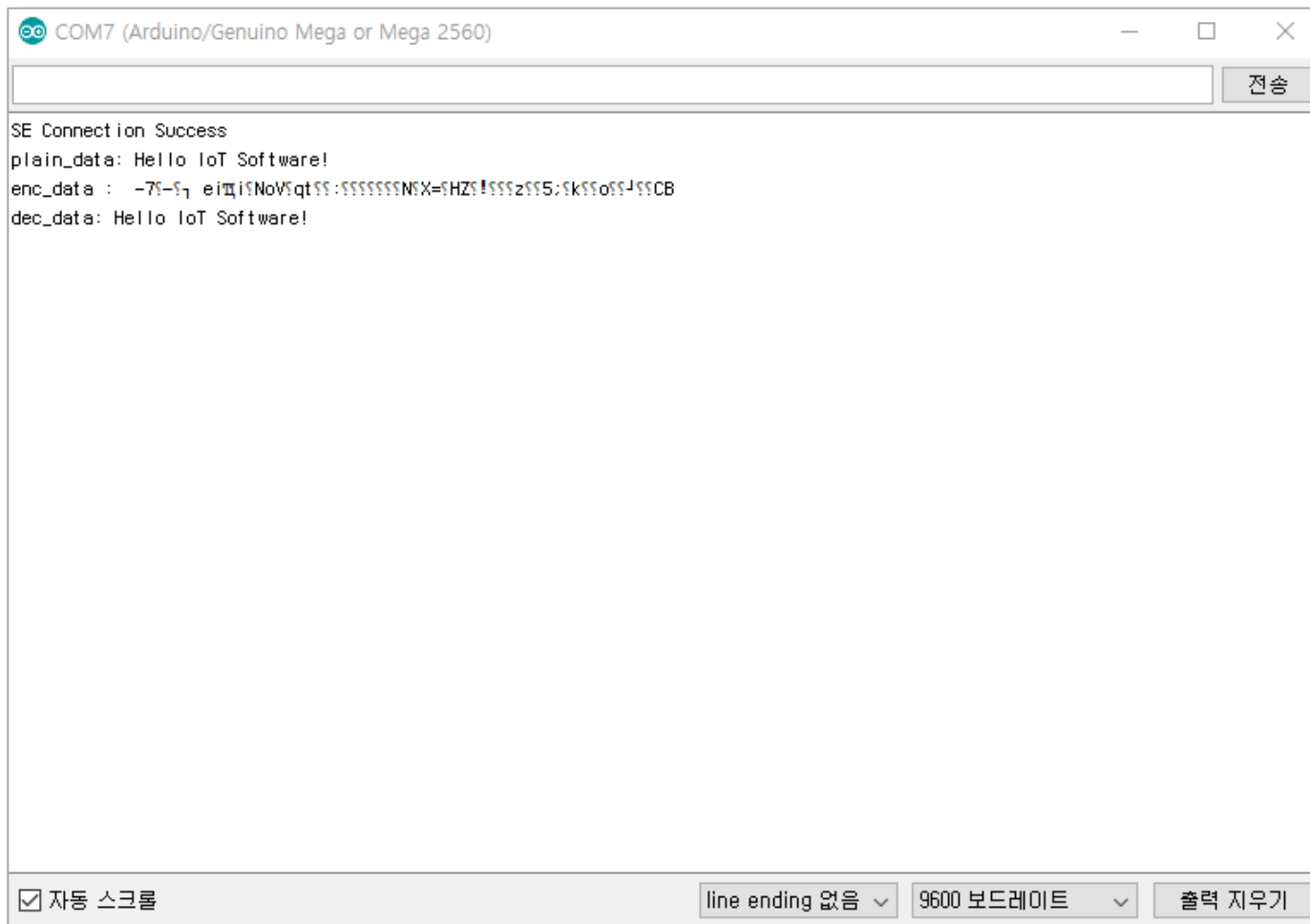
❖ Source code

```
if( Encrypt_AES128( key_num, plain_data, plain_len, enc_data, &enc_len ) ) {  
    Serial.print("enc_data: ");  
    dump( enc_data, enc_len );  
}  
else  
    Serial.println("Encrypt plain_data Failure");  
  
if( Decrypt_AES128( key_num, enc_data, enc_len, dec_data, &dec_len ) ) {  
    Serial.print("dec_data: ");  
    dump( dec_data, dec_len );  
}  
else  
    Serial.println("Decrypt enc_data Failure");  
}  
  
void loop() {  
  
}
```

AES-128

❖ 실행 결과

- 시리얼 모니터로 확인(업로드 완료 후, 7~10초 가량 소요)





❖ API

- `int Generate_RSA1024Key(int key_num)`
 - Return true on success, false on failure
 - int key_num: 0x0~0x1F;

- `int Encrypt_RSA1024(int key_num, byte* plain_data, int plain_len, byte* enc_data, int* enc_len);`
 - Return true on success, false on failure
 - int key_num: int Generate_RSA1024Key(...)에서 사용한 key_num
 - int* enc_len: 128의 배수

- `int Decrypt_RSA1024(int key_num, byte* enc_data, int enc_len, byte* plain_data, int* plain_len);`
 - Return true on success, false on failure
 - int key_num: int Generate_RSA1024Key(...)에서 사용한 key_num
 - int enc_len: 128의 배수

RSA-1024

❖ Source code

```
#include <SArduino.h>

// RSA-1024에서 사용할 key_type 정의
#define PRIVATE    0
#define PUBLIC     1

// buf에 저장된 데이터를 시리얼 모니터로 출력하는 함수
void dump( byte* buf, int len ) {
    int i;

    for( i = 0; i < len; i++ ) {
        Serial.print( (char)buf[ i ] );
    }
    Serial.println();
}

void setup() {
    // 시리얼 모니터 사용
    Serial.begin( 9600, SERIAL_8E2 );

    // SE 초기화
    if( !Init_SE() ) {
        Serial.println("SE Connection Failure");
    }
}
```

```
int key_num = 0x0;

byte plain_data[ 64 ] = "Hello IoT Software!";
int plain_len = strlen( plain_data );

byte enc_data[ 128 ];
int enc_len;

byte dec_data[ 64 ];
int dec_len;

if( !Generate_RSA1024Key(key_num) )
    Serial.println("Set RSA1024 Key Pair Failure");

Serial.print("plain data: ");
dump( plain_data, plain_len );
```



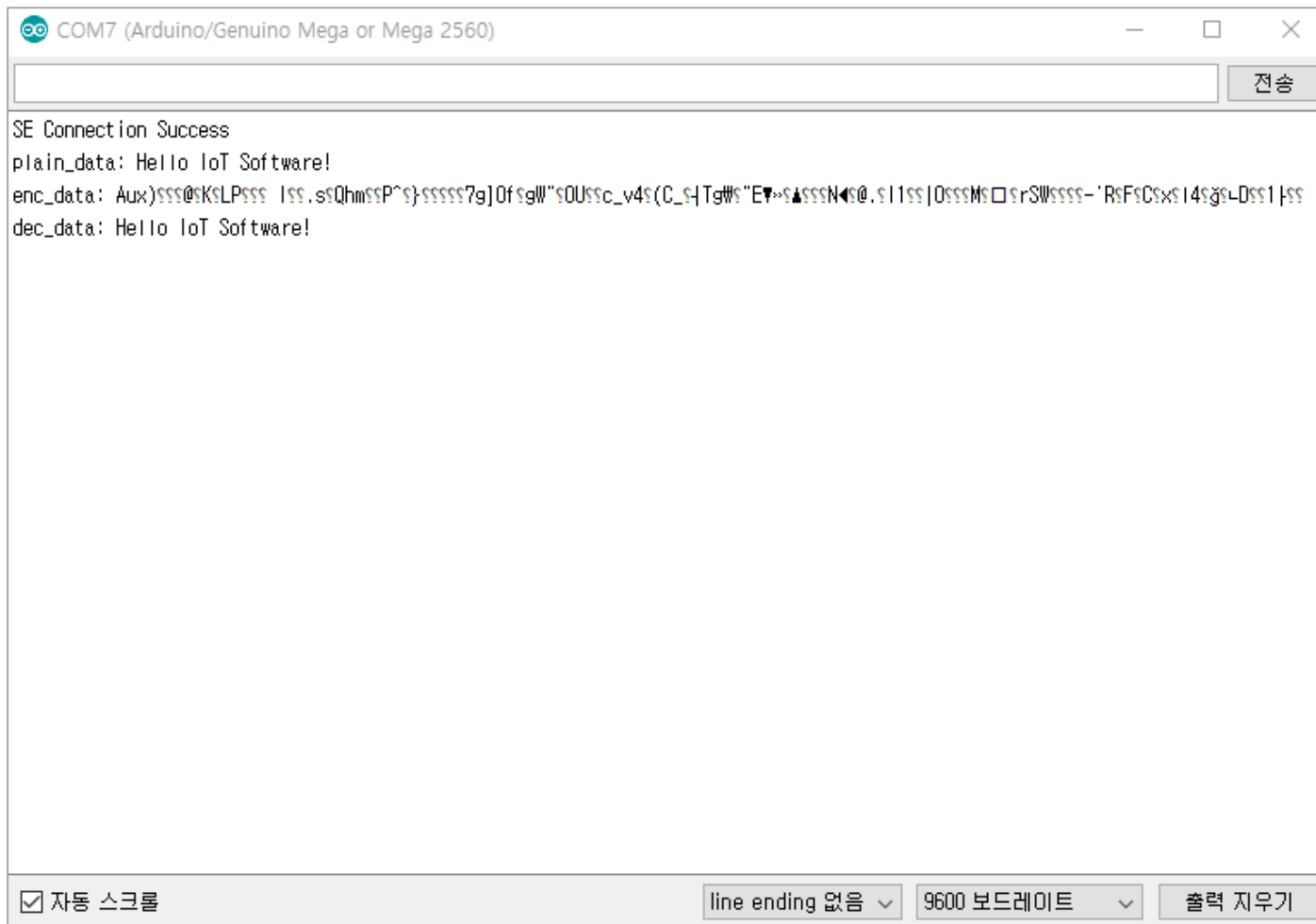

❖ Source code

```
if( Encrypt_RSA1024( key_num, PUBLIC, plain_data, plain_len, enc_data, &enc_len ) ) {  
    Serial.print("enc_data: ");  
    dump( enc_data, enc_len );  
}  
else  
    Serial.println("Encrypt plain_data Failure");  
  
if( Decrypt_RSA1024( key_num, PRIVATE, enc_data, enc_len, dec_data, &dec_len ) ) {  
    Serial.print("dec_data: ");  
    dump( dec_data, dec_len );  
}  
else  
    Serial.println("Decrypt enc_data Failure");  
}  
  
void loop() {  
  
}
```

RSA-1024

❖ 실행 결과

- 시리얼 모니터로 확인(업로드 완료 후, 6~10초 가량 소요)



```
COM7 (Arduino/Genuino Mega or Mega 2560)
SE Connection Success
plain_data: Hello IoT Software!
enc_data: Aux)$$$@K$LP$$$ I$.s$Qhm$$P^$}$$$$7g]Of$gW"$sOU$$c_v4$(C_$|TgW$E$>$$$$N$@.$|1$$|0$$$$M$□$rS$$$- 'R$F$C$x$14$g$-D$1|$$
dec_data: Hello IoT Software!
```

☒ 자동 스크롤 line ending 없음 9600 보드레이트 출력 지우기



❖ 디지털 서명 실습

- 다음 페이지 그림 참조
- SHA-256 및 RSA-1024 실습 활용
- “나는 2018년 6월 1일에 홍길동에게 100만원을 입금하였다” 를 디지털 서명
- 이를 다시 검증



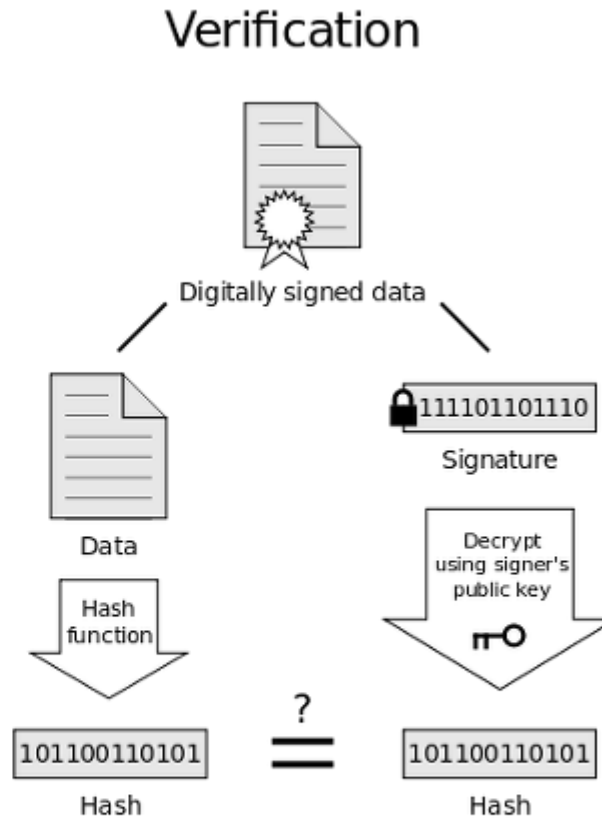
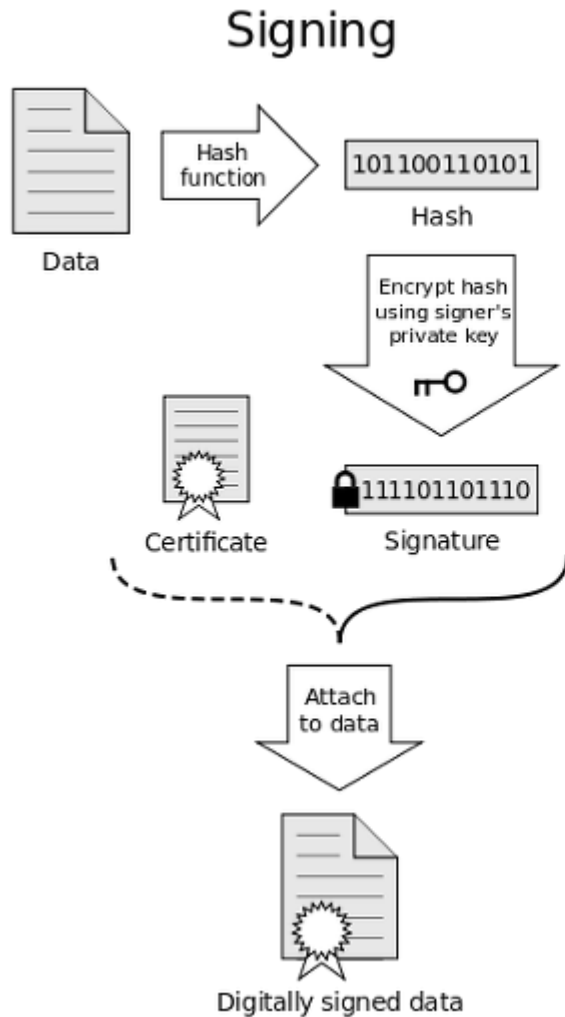
❖ API

- `int Generate_RSA1024Key(int key_num)`
 - Return true on success, false on failure
 - `int key_num`: 0x0~0x1F;

- `int Sign_RSA1024(int key_num, byte* plain_data, int plain_len, byte* sign_data, int* sign_len);`
 - Return true on success, false on failure
 - `int key_num`: `int Generate_RSA1024Key(...)`에서 사용한 `key_num`
 - `int* sign_len`: 128의 배수

- `int Verify_RSA1024(int key_num, byte* sign_data, int sign_len, byte* org_data, int* org_len);`
 - Return true on success, false on failure
 - `int key_num`: `int Generate_RSA1024Key(...)`에서 사용한 `key_num`
 - `int* sign_len`: 128의 배수

Digital Signature



If the hashes are equal, the signature is valid.



Q & A



<http://mesl.khu.ac.kr>