암호인재 인력양성 2차 교육

정보컴퓨터공학과 권혁동

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

암호 라이브러리

- 라이브러리
  - 다른 프로그램과 링크될 수 있는 **오브젝트 코드**
  - **코드 재사용을 위한 기법** 중 하나
  - 소스코드 제공 없이 기술을 제공
  - **개발 시간의 단축**
- 오픈소스 라이브러리
  - 오픈소스 프로젝트에 속한 라이브러리
  - **암호 라이브러리 중에서도 오픈소스 라이브러리가 존재**

암호 라이브러리

- OpenSSL
  - OpenSSL 재단에서 제작
  - 전세계에서 가장 많이 사용되는 오픈소스 암호 라이브러리
- Crypto++
  - C++ 기반의 오픈소스 암호 라이브러리
- Bouncy Castle
  - Java, C#, Kotlin 기반의 오픈소스 암호 라이브러리
  - 다른 오픈소스 암호 라이브러리에 비해 문서가 친절하고 자세함

암호 라이브러리

- 오픈소스 암호 라이브러리는 사용하기에 앞서 개발환경 세팅 필요
- gcc/g++
  - Visual Studio Code
  - Linux(가상환경도 가능)
  - WSL(Windows Subsystem for Linux)
- Java
  - JDK
  - Eclipse IDE

암호 라이브러리

- 자료형
  - 시스템마다 동일한 자료형 이름이지만, **크기가 다를 수 있음**
  - 모든 시스템에서 균일한 동작을 보장하는 **스탠다드 자료형** 사용
  - stdint.h
  - int8_t, int16_t, int32_t, int64_t, uint8_t, uint16_t, uint32_t, uint64_t

패딩과 초기화 벡터

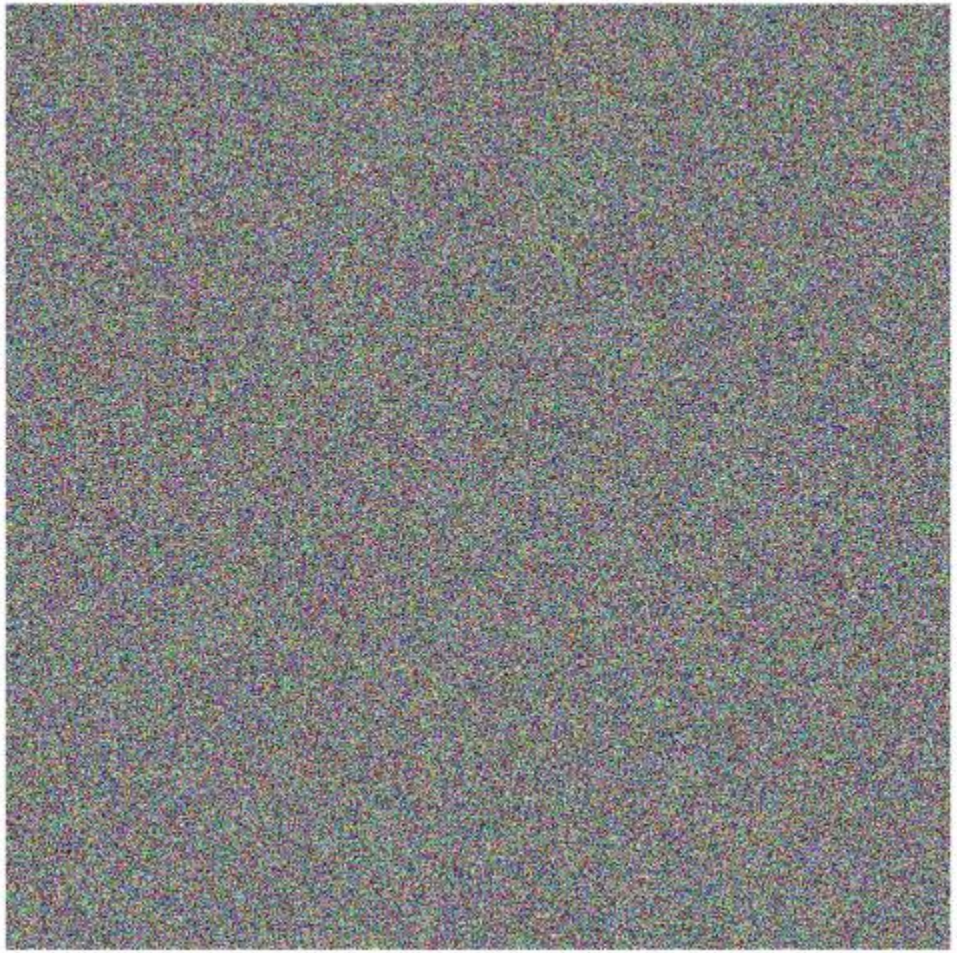
- 패딩
  - **평문 길이가 블록 크기와 맞지 않을 때 사용**
  - 마지막 블록에 추가적으로 붙는 **임의의 값**
  - 임의 값이지만 평문 블록인지 구분해야 하기 때문에 **특별한 규칙을 사용**

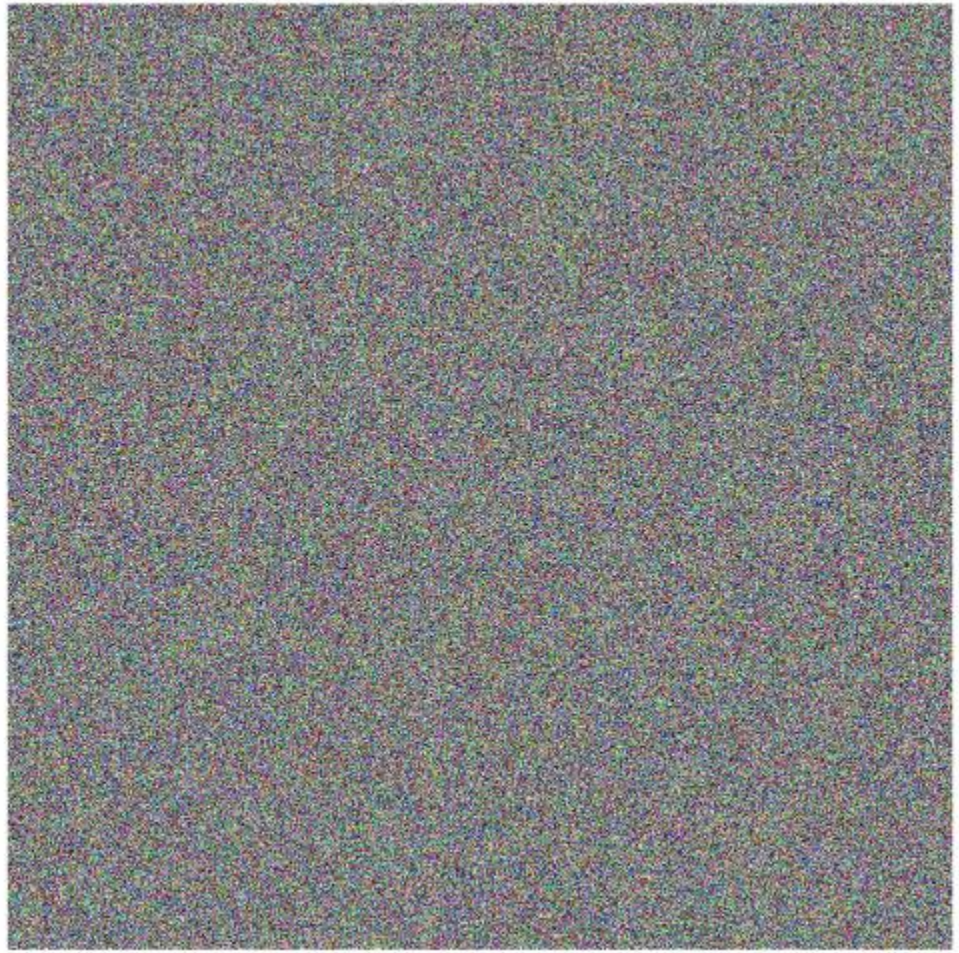
규격	방식
ANSI X.923	dd dd dd dd **00 00 00 04**
ISO 10126	dd dd dd dd **81 a9 d3 04**
PKCS#7	dd dd dd dd **04 04 04 04**
ISO/IEC 7816-4	dd dd dd dd **80 00 00 00**

패딩과 초기화 벡터

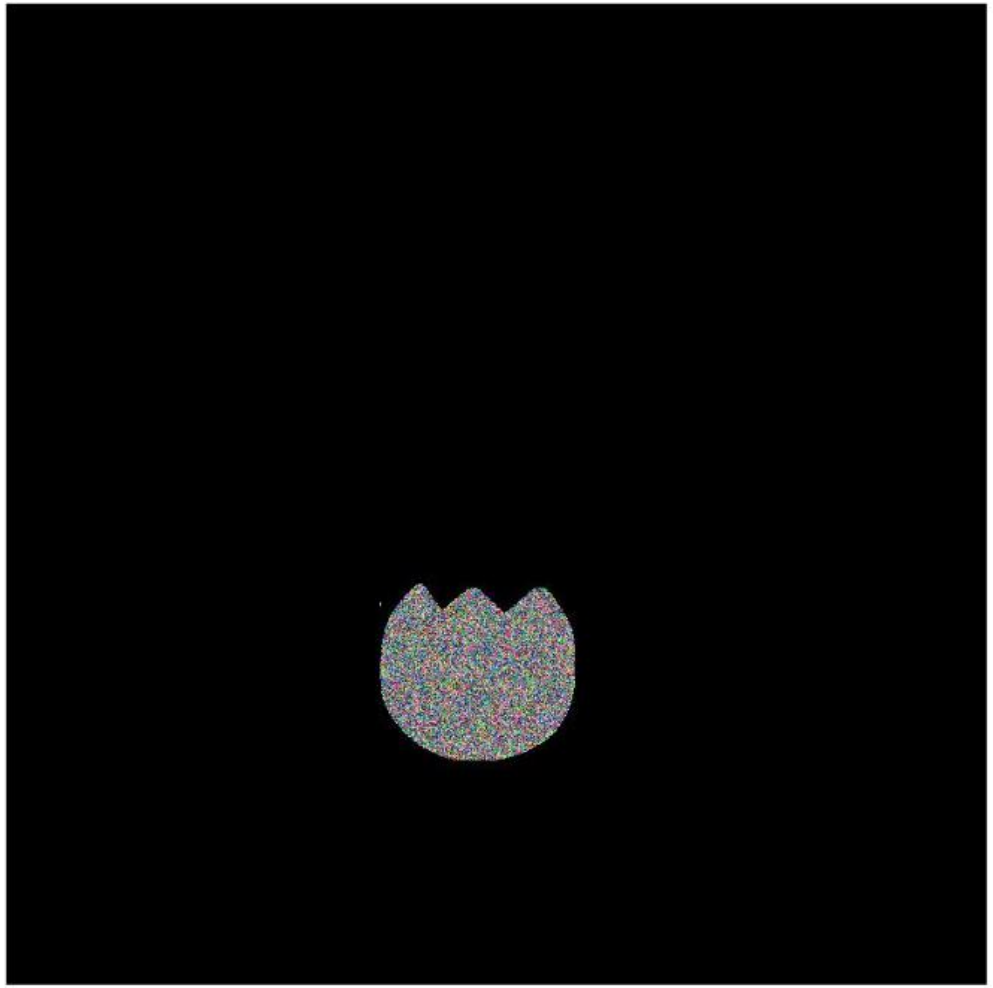
- 초기화 벡터 (Initialization Vector; IV)
  - 첫 블록을 암호화 할 때 사용하는 값
  - 블록 크기와 같은 크기
  - **공개되어도 무관**
  - 반복 사용 지양

패딩과 초기화 벡터

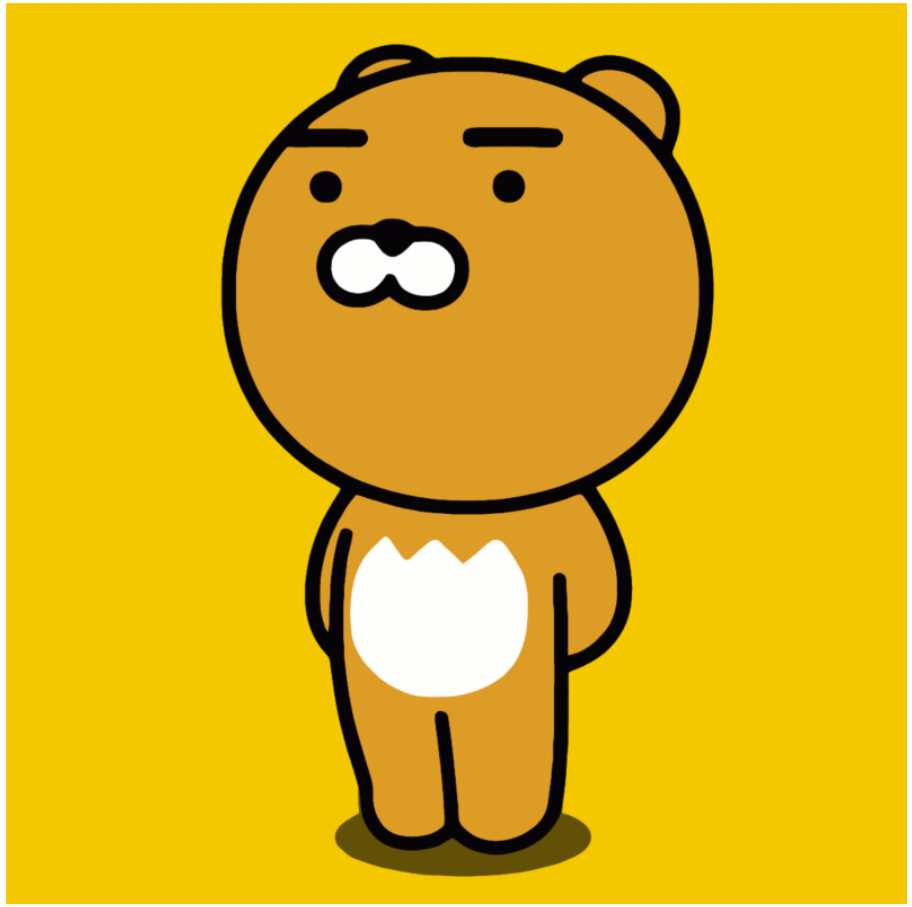




패딩과 초기화 벡터



패딩과 초기화 벡터



OpenSSL 라이브러리 소개

```
#include <stdint.h>
#include <openssl/aes.h>
AES_KEY aes_key;
uint8_t key[32] = ...;
uint8_t pt[16] = ...;
uint8_t ct[16] = ...;
AES_set_encrypt_key(key, 256, &aes_key);
AES_ecb_encrypt(pt, ct, &aes_key, AES_ENCRYPT);
AES_set_decrypt_key(key, 256, &aes_key);
AES_ecb_encrypt(ct, pt, &aes_key, AES_DECRYPT);
```

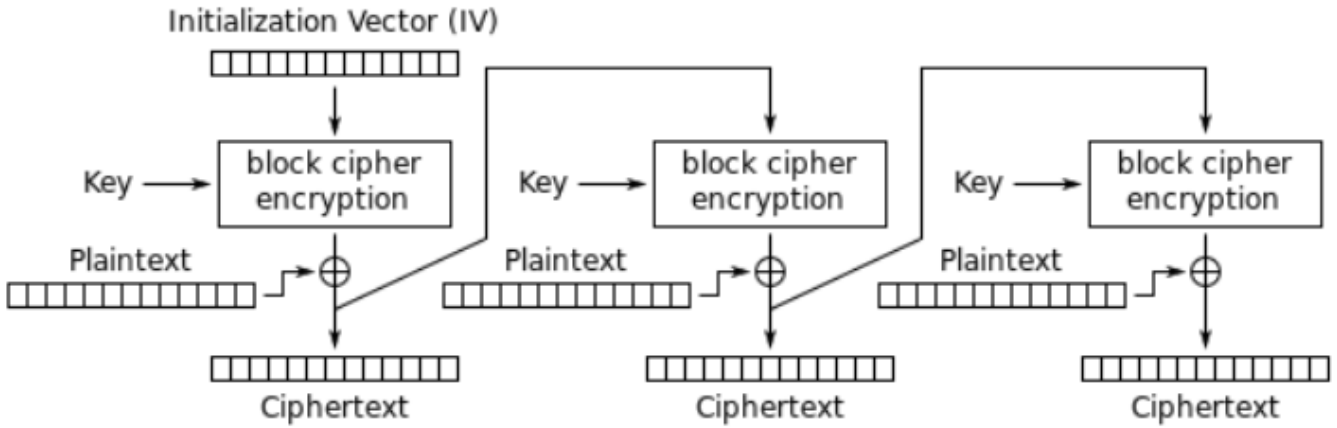
OpenSSL 라이브러리 소개

```
#include <stdint.h>
#include <openssl/aes.h>
AES_KEY aes_key;
uint8_t key[32] = ...;
uint8_t iv[16] = ...;
uint8_t pt[32] = ...;
uint8_t ct[32] = ...;
uint32_t length = 32;
AES_set_encrypt_key(key, 256, &aes_key);
AES_cbc_encrypt(pt, ct, length, &aes_key, iv, AES_ENCRYPT);
AES_set_decrypt_key(key, 256, &aes_key);
AES_cbc_encrypt(ct, pt, length, &aes_key, iv, AES_DECRYPT);
```

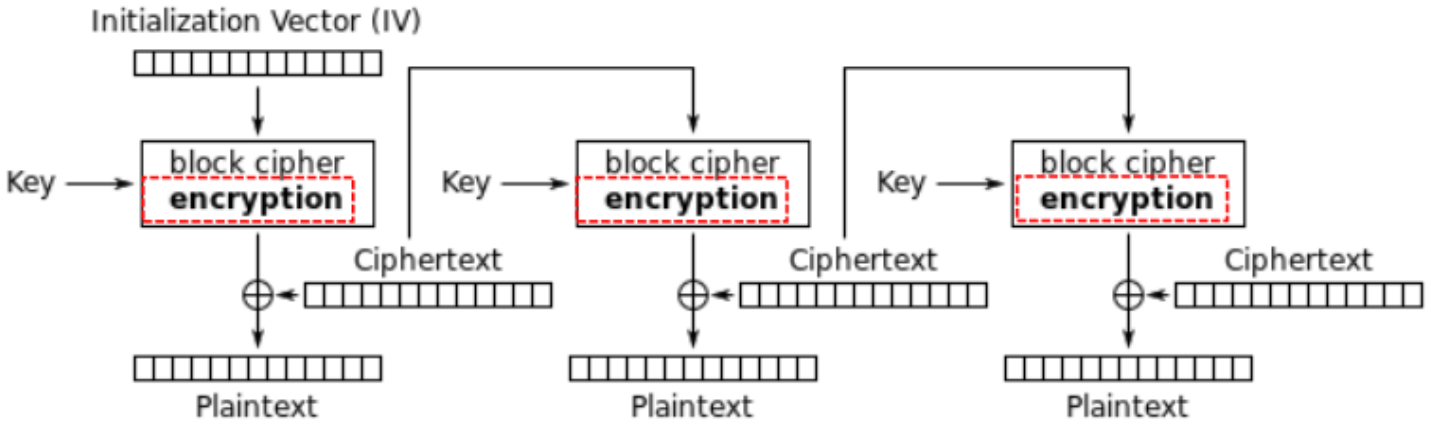
OpenSSL 라이브러리 소개

```
#include <stdint.h>
#include <openssl/aes.h>
AES_KEY aes_key;
uint8_t key[32] = ...;
uint8_t iv[16] = ...;
uint8_t pt[32] = ...;
uint8_t ct[32] = ...;
uint32_t length = 32;
Unsigned int num = 0;
AES_set_encrypt_key(key, 256, &aes_key);
AES_cfb128_encrypt(pt, ct, length, &aes_key, iv, &num, AES_ENCRYPT);
AES_set_encrypt_key(key, 256, &aes_key);
AES_cfb128_encrypt(ct, pt, length, &aes_key, iv, &num, AES_DECRYPT);
```

OpenSSL 라이브러리 소개



Cipher Feedback (CFB) mode encryption

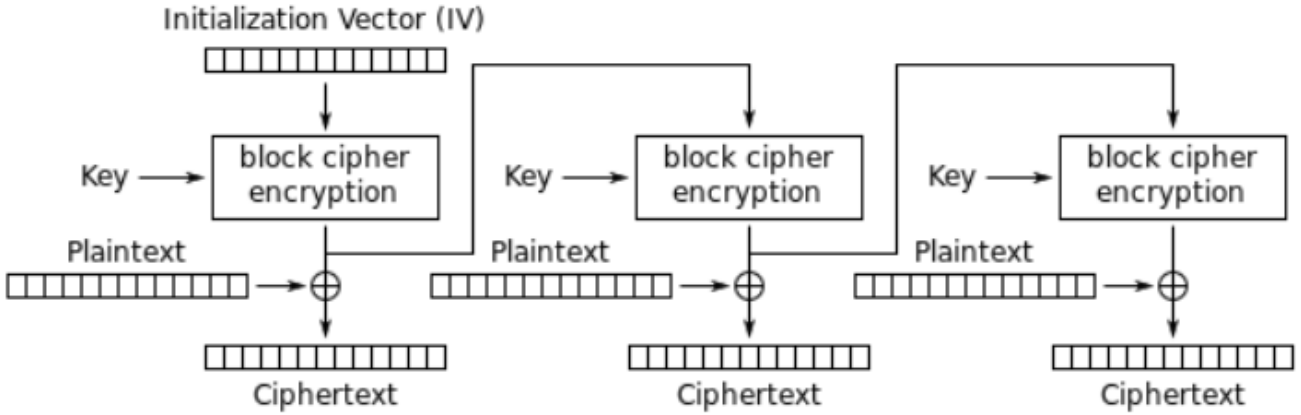


Cipher Feedback (CFB) mode decryption

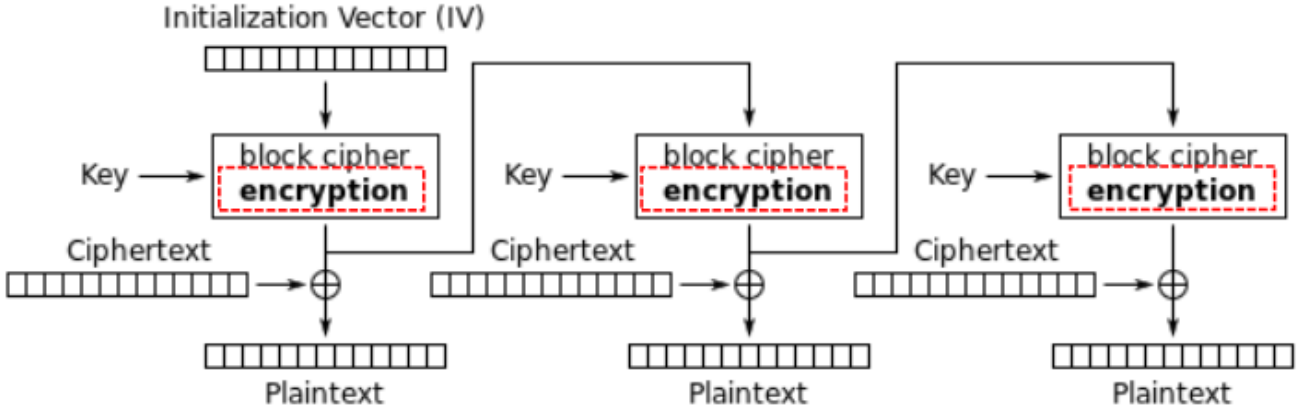
OpenSSL 라이브러리 소개

```
#include <stdint.h>
#include <openssl/aes.h>
AES_KEY aes_key;
uint8_t key[32] = ...;
uint8_t iv[16] = ...;
uint8_t pt[32] = ...;
uint8_t ct[32] = ...;
uint32_t length = 32;
Unsigned int num = 0;
AES_set_encrypt_key(key, 256, &aes_key);
AES_ofb128_encrypt(pt, ct, length, &aes_key, iv, &num);
AES_set_encrypt_key(key, 256, &aes_key);
AES_ofb128_encrypt(ct, pt, length, &aes_key, iv, &num);
```

OpenSSL 라이브러리 소개



Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

OpenSSL 라이브러리 소개

```
#include <stdint.h>
#include <openssl/aes.h>
#include <openssl/modes.h>
AES_KEY aes_key;
uint8_t key[32] = ...;
uint8_t ctr[16] = ...;
uint8_t pt[32] = ...;
uint8_t ct[32] = ...;
uint32_t length = 32;
Unsigned int num = 0;
uint8_t ectr[16] = ...;
AES_set_encrypt_key(key, 256, &aes_key);
CRYPTO_ctr128_encrypt(pt, ct, length, &aes_key, ctr, ectr, &num, (block128_f)AES_encrypt);
AES_set_encrypt_key(key, 256, &aes_key);
CRYPTO_ctr128_encrypt(ct, pt, length, &aes_key, ctr, ectr, &num, (block128_f)AES_encrypt);
```

임베디드 구현

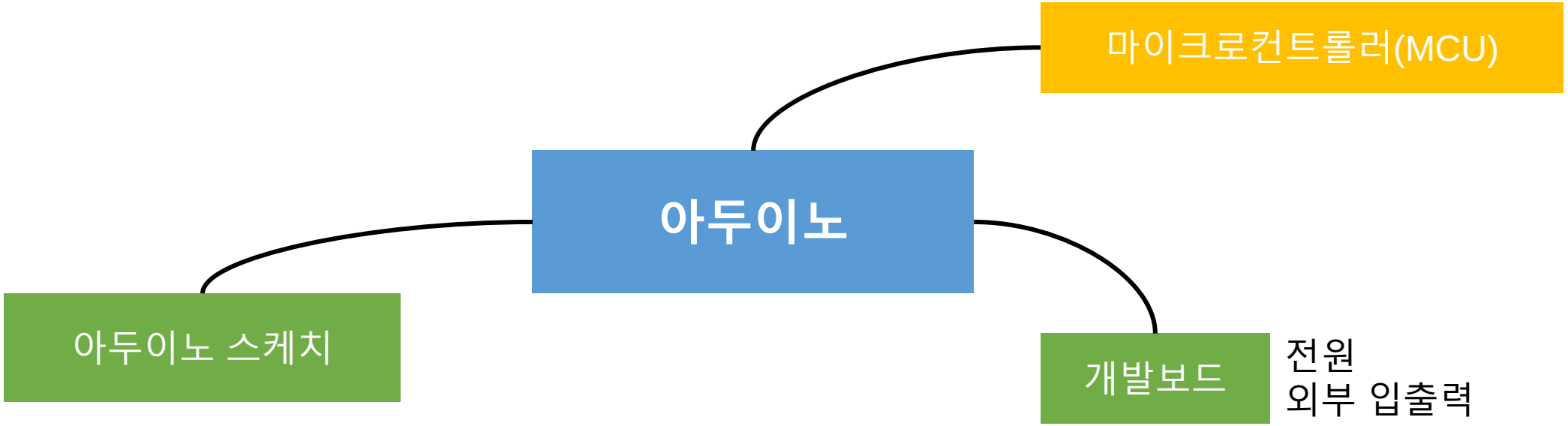
- 마이크로 컨트롤러
  - 집적회로 안에 **최소한의 컴퓨팅 요소**를 내장한 초소형 컨트롤러
  - **저성능, 저전력, 저비용**
  - 전원만 공급된다면, **프로그래밍된 작업을 수행**
- 분야
  - 대부분의 가전제품
  - 산업용 단순제어
  - 프로그래밍 교육
- 제품
  - **8bit Atmel AVR 시리즈**
  - 16bit Texas Instrument MSP 시리즈
  - **32bit ARM Cortex-M 시리즈**

임베디드 구현

- 구현 절차
  - 코드 작성: C코드 작성
  - 컴파일: avr-gcc, gcc-arm-none-eabi
  - 바이너리 업로드: ROM Writer, Platform IO
  - 전원 공급: 코드 실행

임베디드 구현

- 아두이노
  - 오픈소스 하드웨어
  - 스케치 IDE를 사용하여 아두이노 프로그래밍 작성 가능



임베디드 구현

- 마이크로 컨트롤러는 열악한 환경을 지니므로 최적화 전략이 필요

성능 최적화	코드 최적화	균형 최적화
수행 속도가 우선되는 환경 **코드 크기가 커지는 경향** 기법: 함수 사용 지양 인라인 함수, 매크로 사용 루프 해제 인라인 어셈블리	저장공간을 우선시하는 환경 **성능이 다소 떨어짐** 기법: 적극적인 함수 사용	플랫폼에 맞춰 적절한 성능 유지