암호프로그래밍
CRYPTOGRAPHY PROGRAMMING

7. 대칭키 암호

정보보호학과
이병천 교수

차례

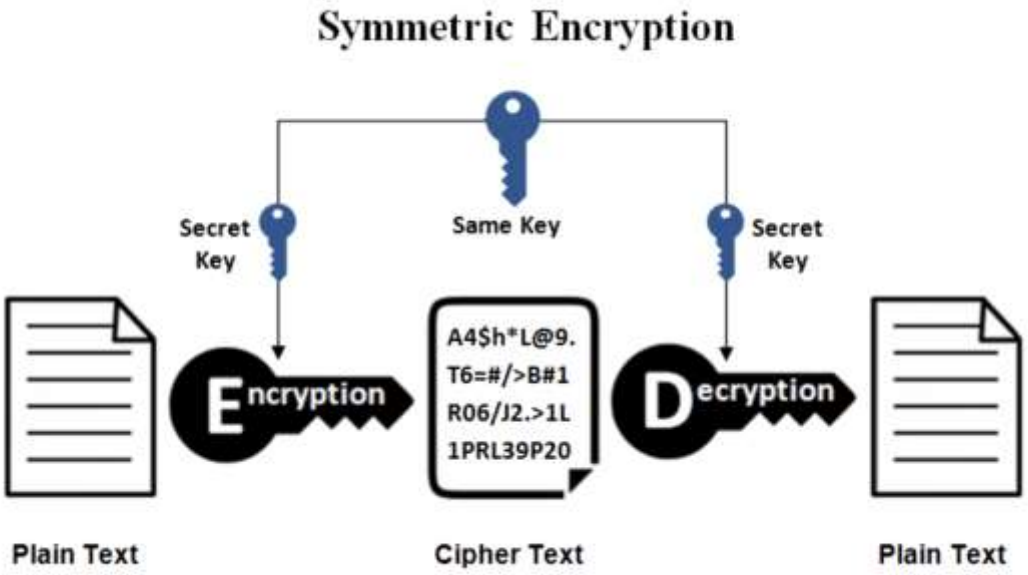
- 1. 강의 개요
- 2. 암호와 정보보호
- 3. 프로그래밍 환경 구축 - 웹, 파이썬
- 4. 해시함수
- 5. 메시지인증코드
- 6. 패스워드기반 키생성
- **7. 대칭키 암호**
- 8. 공개키 암호
- 9. 전자서명
- 10. 인증서와 공개키기반구조(PKI)

7. 대칭키 암호

1. 대칭키 암호
2. 웹, 자바스크립트
3. 파이썬

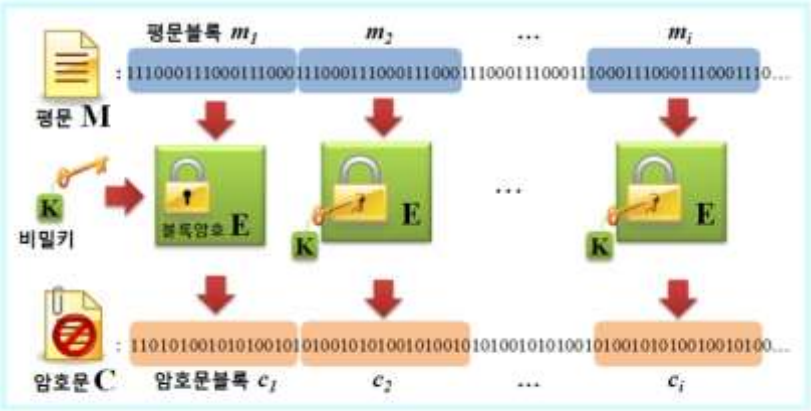
1. 대칭키 암호

- 대칭키 암호화 (Symmetric encryption)
  - 비밀키를 이용한 메시지 암호화, 암호문 복호화
  - 송신자와 수신자가 동일한 비밀키를 공유
  - 비밀키는 난수로 생성하여 사용해야 안전



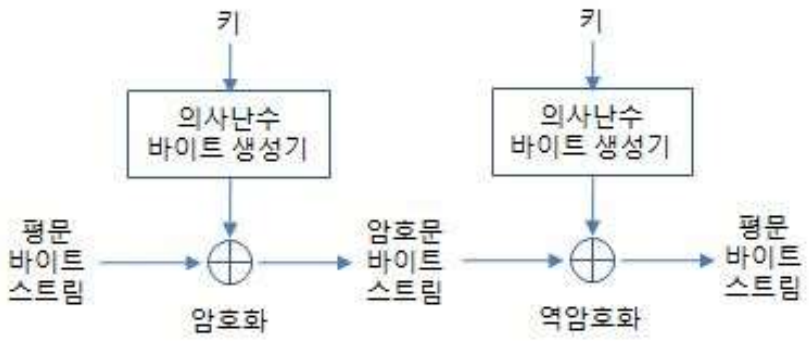
대칭키 암호 방식

- 데이터를 변환하는 단위에 따라 블록 암호와 스트림 암호로 분류
  - 블록암호: 고정된 크기의 블록 단위로 암호화/복호화
  - 스트림암호: 난수열을 생성하여 비트단위, 글자단위로 암호화/복호화



[그림 2-2] 블록암호 암호화(ECB모드) 과정

블록 암호



스트림 암호

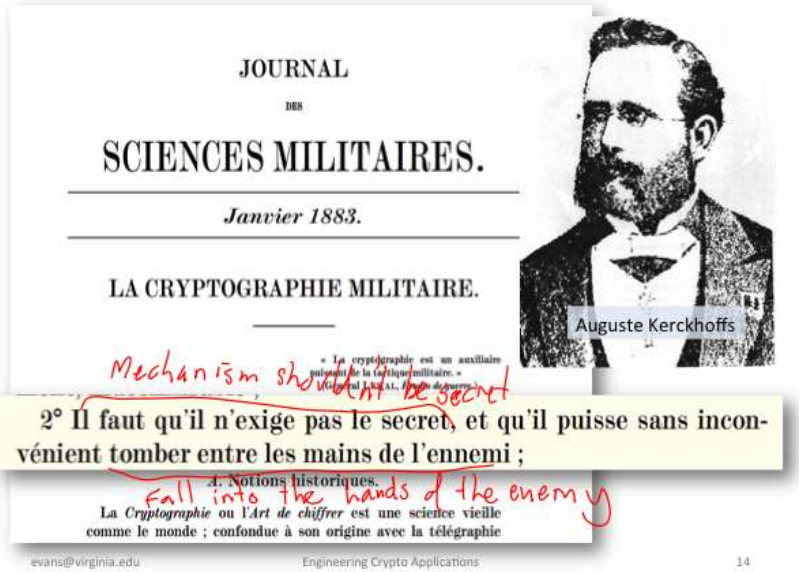
대칭키 암호 방식

- 블록 암호 vs. 스트림 암호

구분	스트림 암호	블록 암호
구현성/효율성	용이	복잡하나, 효율적일 수 있음
컴퓨팅 요구 자원	적음	많음
암호화 단위	비트별 암호화	블록별 암호화
사용 예	GSM 휴대폰 A5/1, RC4 등	DES,3DES,AES,SEED, RC5 등
응용분야	음성/영상 스트리밍, 작은 컴퓨팅 장비 (임베디드시스템 등)	일반 데이터 저장,전 송 등

커크호프의 원리(Kerckhoffs' Principle)

- Kerckhoffs' Principle
  - A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.
  - 키를 제외한 시스템의 다른 모든 내용이 알려지더라도 암호체계는 안전해야 한다. 암호체계를 공개하지 않는 비밀 암호시스템은 오히려 안전하지 않을 수 있음.
  - 많은 전문가들의 공격시도를 견뎌야 안전성을 인정받음



블록 암호

- 블록 암호 알고리즘

Algorithms	Blow Fish	AES	3DES	DES
Key size (bits)	32-448	128, 192, 256	112 or 118	64
Block size (bits)	64	128	64	64
Round	16	10, 12, 14	84	16
Structure	Feistel	Substitution Permutation	Feistel	Feistel
Flexible	Yes	Yes	Yes	No
Features	Secure enough	Excellent Security	Adequate security Replacement for DES,	Not structure, Enough
Speed	fast	fast	Very slow	slow

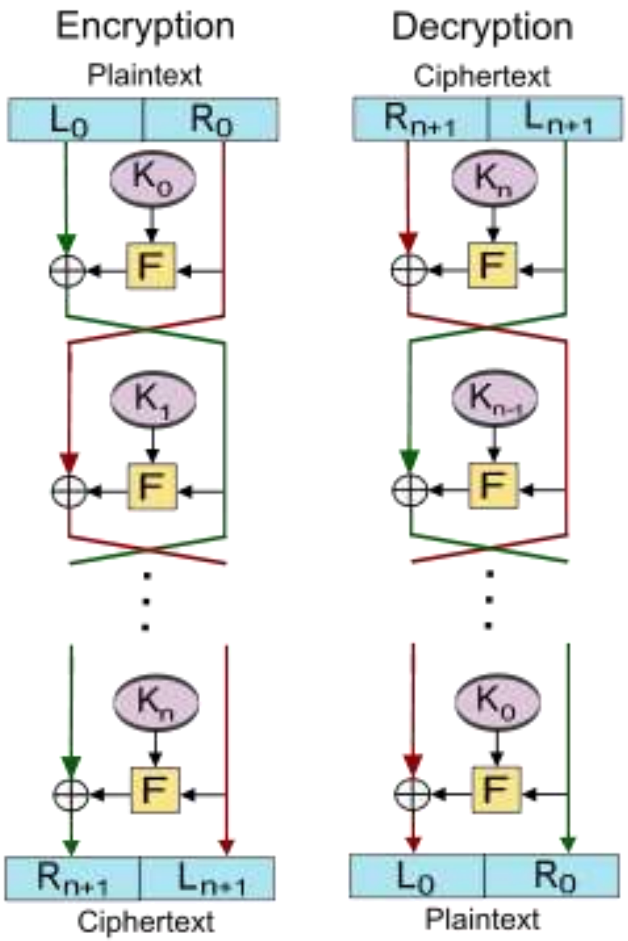
블록 암호 알고리즘

- DES
  - 1977년 NBS(미국 표준국)에서 미국 표준 알고리즘으로 채택
  - 64비트 블록단위 암호화, 64비트(56비트) 키길이
  - 짧은 키길이로 인한 취약성으로 현재는 사용하지 않음
  - Feistel 구조
- 3DES (DESede)
  - DES 암호를 암호화->복호화->암호화 순으로 3번 반복
  - 블록 크기가 64비트, 느린 속도, 3배의 키길이
- AES
  - 2001년 DES를 교체하기 위해 새롭게 제정한 암호 표준
  - 128/192/256 비트의 키길이, 10/12/14 라운드 사용
  - SPN(Substitution Permutation Network) 구조

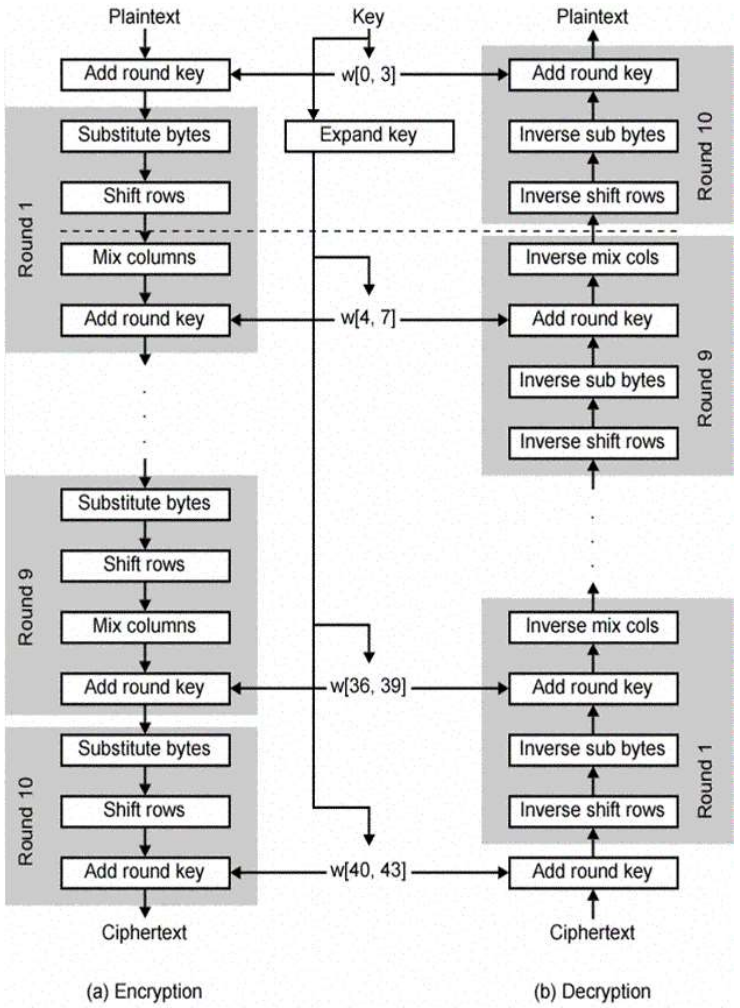
국내 블록 암호 알고리즘

- SEED
  - 1999년 2월 국내에서 개발한 128비트 블록암호알고리즘
  - Feistel 구조
  - 국내 표준, ISO/IEC 국제 표준
  - SEED128, SEED256
- ARIA
  - 국가보안기술연구소에서 개발 (Involutional SPN 구조)
  - 2004년 국가표준, 2010년 웹 표준
  - 경량 환경, 하드웨어 구현 최적화된 128비트 블록암호알고리즘
  - 128/192/256비트의 키길이, 12/14/16 라운드

DES, AES



DES (Feistel 구조)



AES (SPN 구조)

암호 알고리즘의 키길이와 안전성

<NIST의 알고리즘의 안전성 유지 기간 및 최소 키 길이(비트) 권고>

알고리즘의 안전성 보장 기간	보안강도 (비트)	대칭키 알고리즘	비대칭키 알고리즘			ECC (예:ECDSA)	해쉬함수 (기능 A)[1]	해쉬함수 (기능 B)[2]
			인수분해 기반 (예 : RSA)	이산대수 기반 (예 : KCDSA)				
				공개키	개인키			
~ '10년	80	2TDEA	1024	1024	160	160	SHA-1	SHA-1
'11년 ~ '30년	112	3TDEA	2048	2048	224	224	SHA-224	SHA-1
'30년 ~	128	AES-128	3072	3072	256	256	SHA-256	SHA-1
	192	AES-192	7680	7680	384	384	SHA-384	SHA-224
	256	AES-256	15360	15360	512	512	SHA-512	SHA-256

1) 해쉬함수(기능 A) : 해쉬함수가 전자서명 및 해쉬전용 어플리케이션용으로 사용되는 경우
2) 해쉬함수(기능 B) : 해쉬함수가 HMAC, 키 유도 및 난수 생성 기능을 위해 사용되는 경우

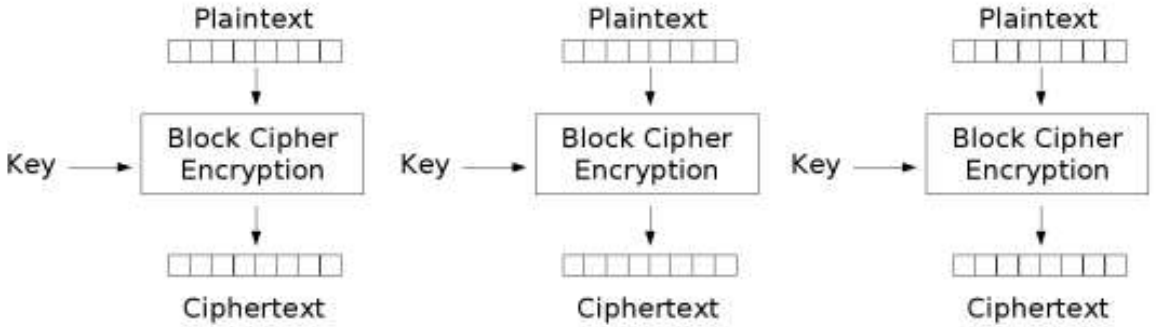
<ECRYPT의 알고리즘의 안전성 유지 기간 및 최소 키 길이(비트) 권고>

보안등급	보안강도 (비트)	알고리즘의 안전성 보장 기간	대칭키 알고리즘	비대칭키 알고리즘			ECC	해쉬함수
				인수분해 기반	이산대수 기반			
					공개키	개인키		
4	80	~ '11년	80	1248	1248	160	160	160
5	96	~ '18년	96	1776	1776	192	192	192
6	112	~ '28년	112	2432	2432	224	224	224
7	128	~ '38년	128	3248	3248	256	256	256

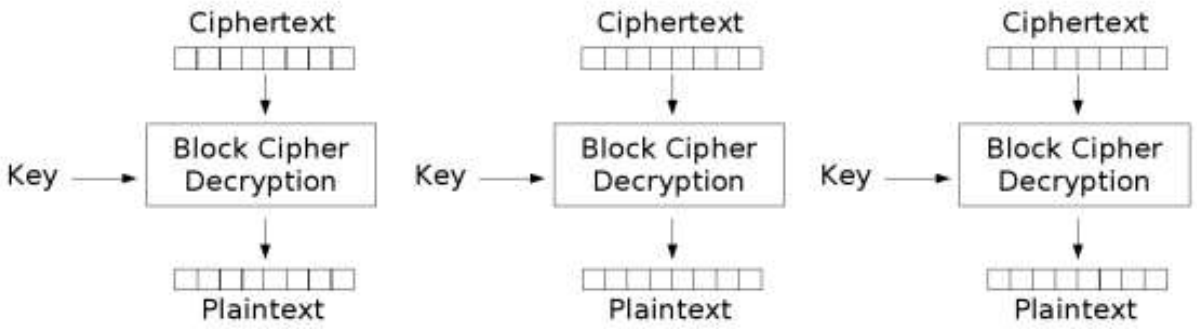
블록암호의 운영모드

- 전자 코드북(electronic codebook, ECB) 모드
- 암호문 블록 체인 (cipher-block chaining, CBC) 모드
- 암호문 피드백(cipher feedback, CFB) 모드
- 출력 피드백(output feedback, OFB) 모드
- 카운터(Counter, CTR) 모드
- 암호화와 인증을 결합한 모드 – CCM, GCM

ECB (Electronic Codebook) 모드



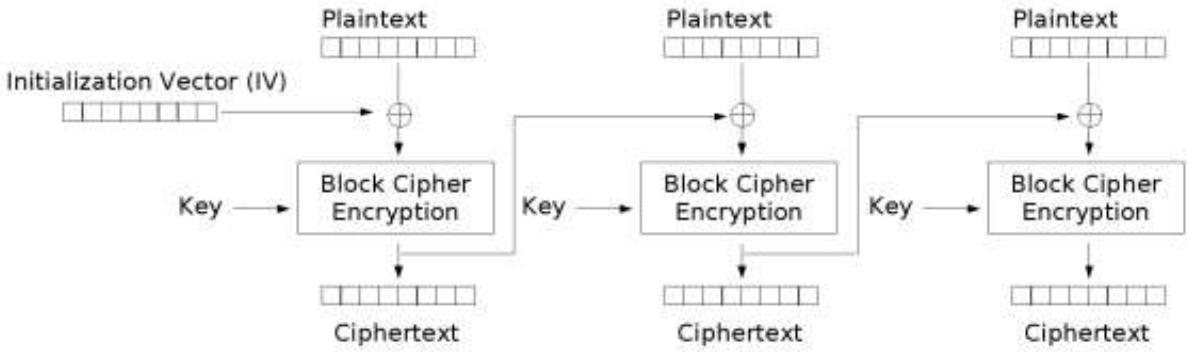
Electronic Codebook (ECB) mode encryption



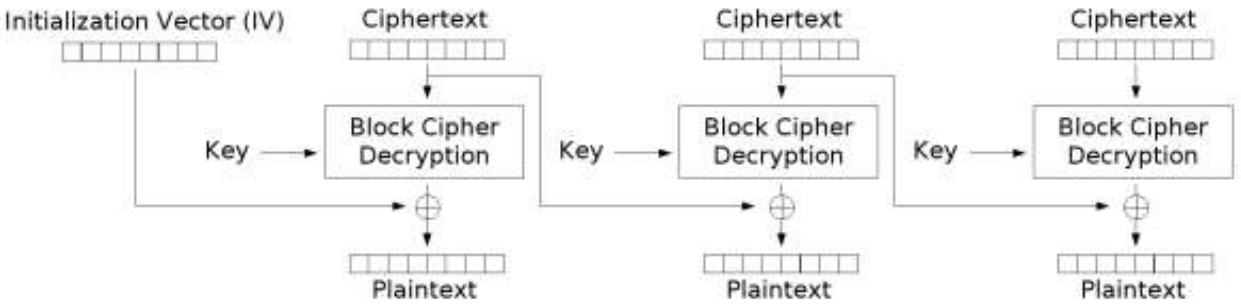
Electronic Codebook (ECB) mode decryption

- 가장 단순한 모드로 블록단위로 순차적으로 암호화 하는 구조이다.
- 한 개의 블록만 해독되면 나머지 블록도 해독이 되는 단점이 있다.
- 각 블록이 독립적으로 동작하므로 한 블록에서 에러가 난다고 해도 다른 블록에 영향을 주지 않는다.
- 블록간의 상관관계가 없기 때문에 평문이 같으면 암호문도 같다.

CBC (Cipher Block Chaining) 모드



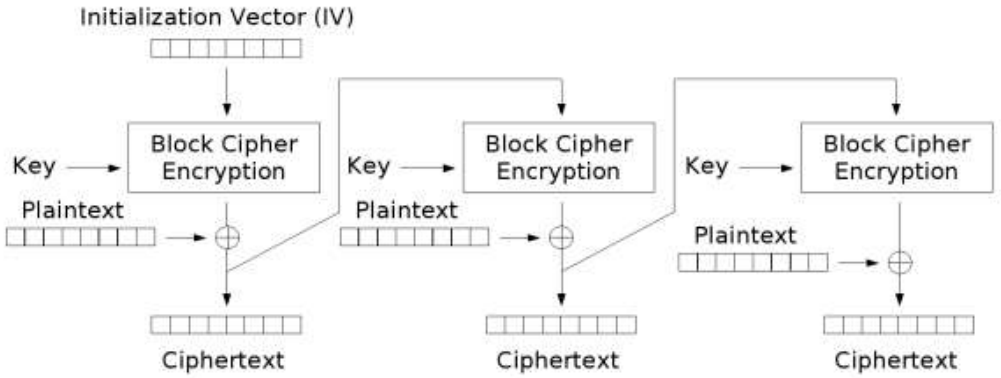
Cipher Block Chaining (CBC) mode encryption



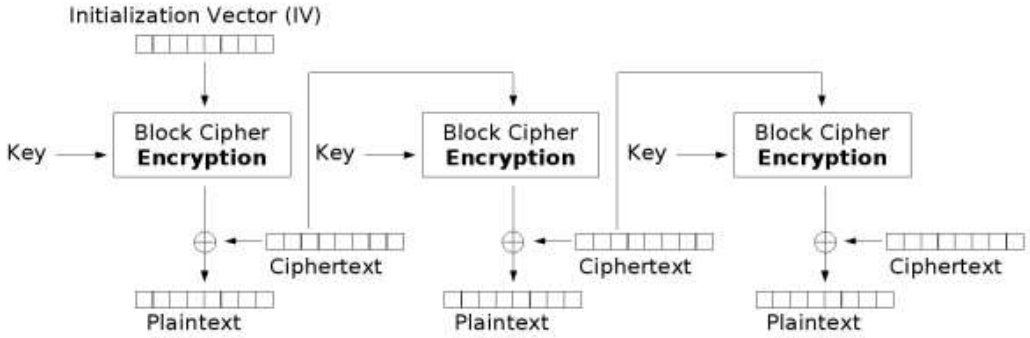
Cipher Block Chaining (CBC) mode decryption

- 블록 암호화 운영 모드 중 보안성이 높은 암호화 방법으로 가장 많이 사용된다.
- 평문의 각 블록은 XOR연산을 통해 이전 암호문과 연산되고 첫번째 암호문에 대해서는 IV(Initial Vector)가 암호문 대신 사용된다. 이 때, IV는 제 2의 키가 될 수 있다.
- 암호화가 병렬처리가 아닌 순차적으로 수행되어야 한다.

CFB (Cipher Feedback) 모드



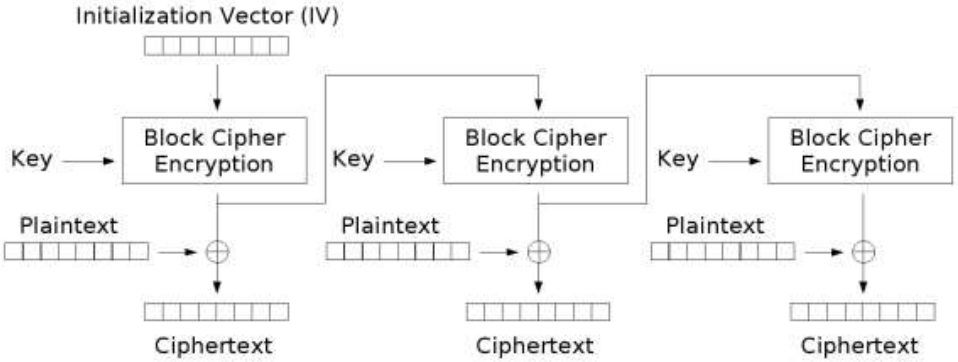
Cipher Feedback (CFB) mode encryption



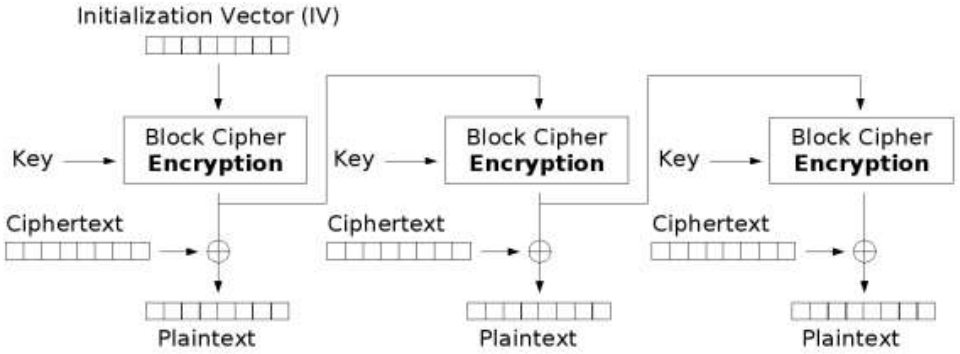
Cipher Feedback (CFB) mode decryption

- 블록 암호화를 스트림 암호화처럼 구성해 평문과 암호문의 길이가 같다(패딩이 필요 없다)
- 최초의 키생성 버퍼로 IV가 사용되며, 이때 IV는 제2의 키가 될수 있다.
- 스트림의 기본단위를 Bit단위로 설정할 수 있으며, Bit단위에 따라 CFB8~CFB128로 쓰인다.
- 암호화, 복호화 모두 암호화로만 처리할 수 있다.
- CBC모드와 마찬가지로 암호화는 순차적이고, 복호화는 병렬적으로 처리할 수 있다.

OFB (Output Feedback) 모드



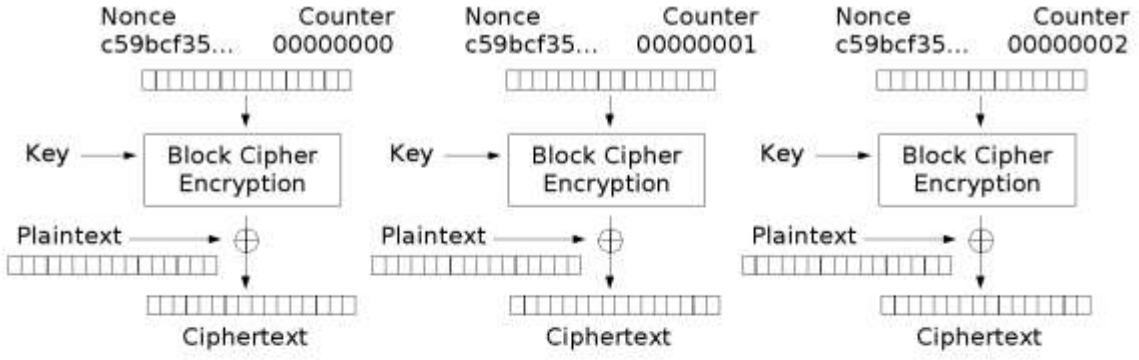
Output Feedback (OFB) mode encryption



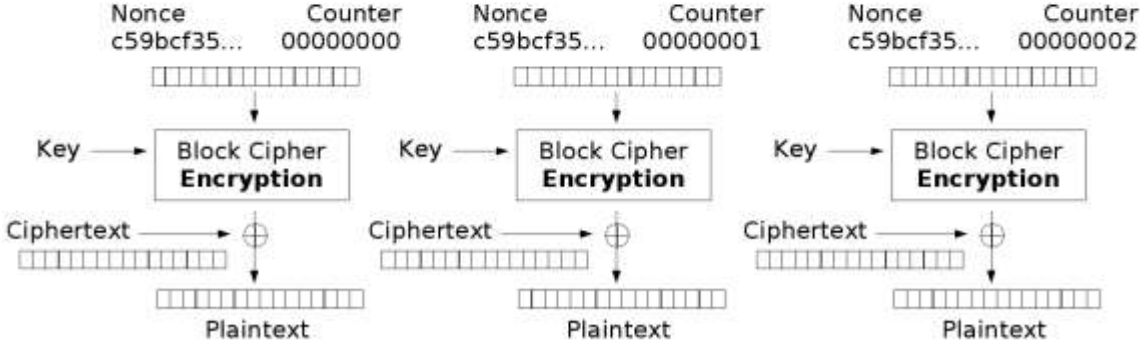
Output Feedback (OFB) mode decryption

- 블록 암호화를 스트림 암호화처럼 구성해 평문과 암호문의 길이가 같다.(패딩이 필요없다)
- 암호화 함수는 키 생성에만 사용되며, 암호화 방법과 복호화 방법이 동일해 암호문을 한번 더 암호화하면 평문이 나온다. (복호화시에 암호화)
- 최초의 키생성 버퍼로 IV가 사용되며, 이 때 IV는 제2의 키가 될수 있다.
- 스트림의 기본 단위를 Bit단위로 설정할 수 있으며, Bit단위에 따라 OFB8~OFB128로 쓰인다.

CTR (Counter) 모드



Counter (CTR) mode encryption

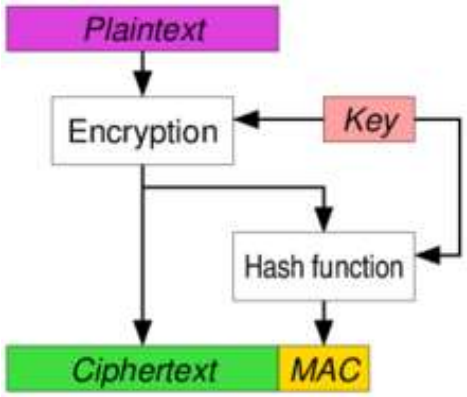


Counter (CTR) mode decryption

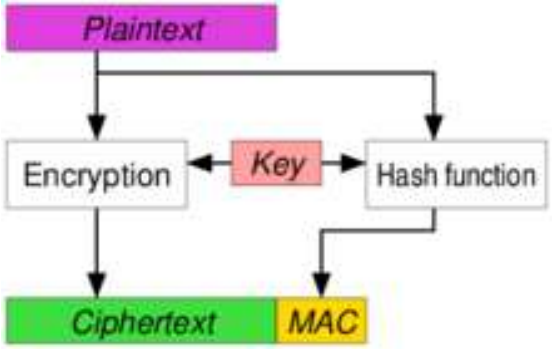
- 블록을 암호화할 때마다 1씩 증가해 가는 카운터를 암호화 해서 키스트림을 만든다. 즉 카운터를 암호화한 비트열과 평문블록과의 XOR를 취한 결과가 암호문 블록이 된다.
- CTR모드는 OFB와 같은 스트림 암호의 일종이다.
- CTR모드의 암/복호화는 완전히 같은 구조가 되므로 구현이 간단하다.(OFB와 같은 스트림 암호의 특징)
- CTR모드에서는 블록의 순서를 임의로 암/복호화 할 수 있다.(블록번호로부터 카운터를 구할 수 있음)
- 블록을 임의의 순서로 처리할 수 있다는 것은 처리를 병행할 수 있다는 것을 의미한다.(병렬처리 가능)

암호화와 인증을 결합한 블록암호 운영모드

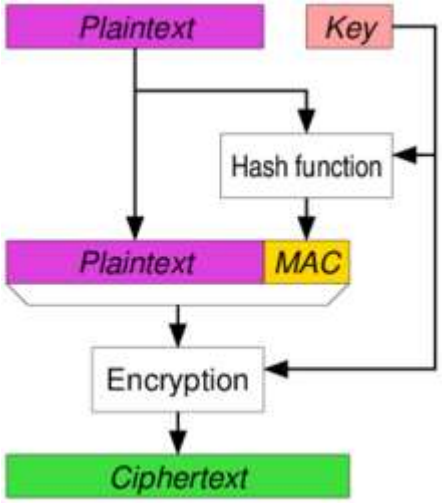
- Authenticated Encryption
  - 암호화와 인증을 함께 제공
  - Encrypt-then-MAC (EtM)
  - Encrypt-and-MAC (E&M)
  - MAC-then-Encrypt (MtE)



Encrypt-then-MAC (EtM)



Encrypt-and-MAC (E&M)

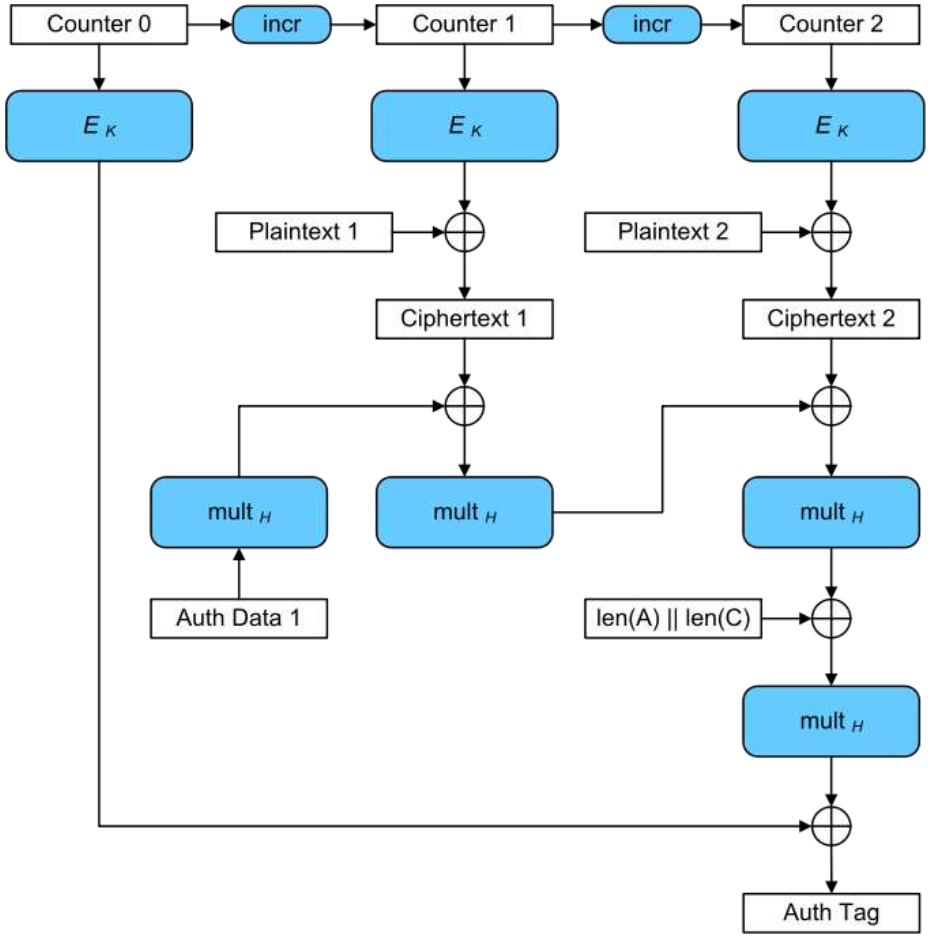


MAC-then-Encrypt (MtE)

암호화와 인증을 결합한 블록암호 운영모드

- CCM (Counter with CBC-MAC)
  - 암호화와 메시지 인증 기능을 동시에 제공하는 운영 모드
  - Authenticated Encryption: authenticate-then-encrypt
  - 메시지 인증을 위해 CBC-MAC 모드를 사용하고, 메시지 암호화를 위하여 CTR 모드를 사용
- GCM (Galois/Counter Mode)
  - CTR 모드와 GHASH 를 함께 사용하는 운영 모드
  - Authenticated Encryption
  - 병렬처리 가능하여 효율성 높음

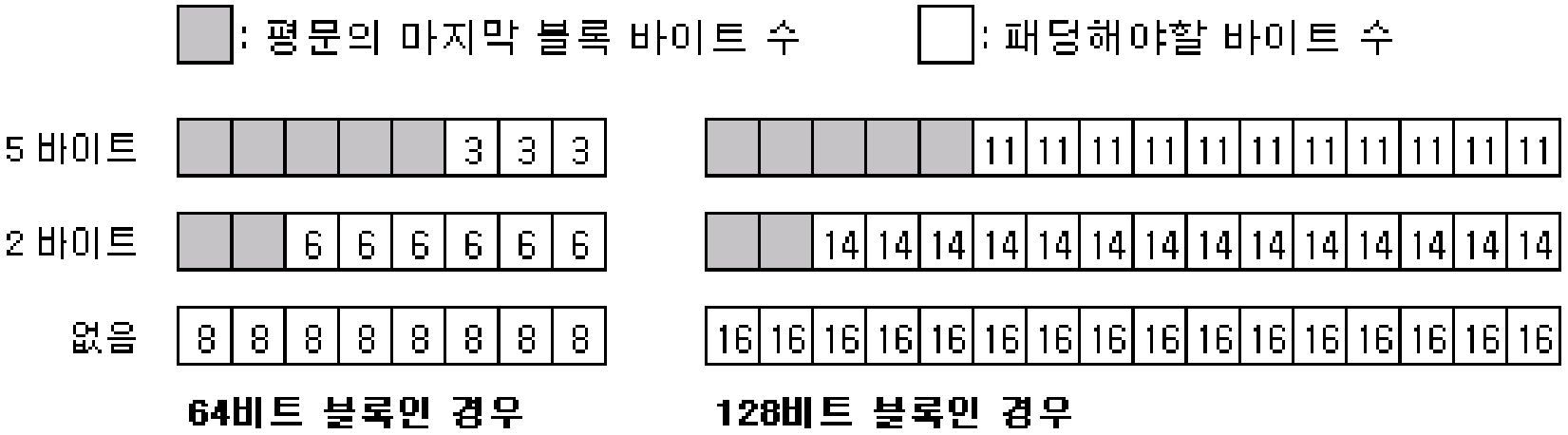
암호화와 인증을 결합한 블록암호 운영모드



GCM (Galois/Counter Mode)

패딩 (padding, 채우기)

- 블록암호는 정해진 길이의 블록단위로 암호화 수행
- 임의의 길이를 가진 데이터를 블록 길이의 배수로 맞추는 것을 패딩(Padding)이라고 함
- 패딩된 데이터에서 원래 데이터를 복구할 수 있어야 함



패딩 (padding, 채우기)

- 패딩의 종류
  - **none:** 사용자가 완전 블록임을 보장해 주어야 한다.
  - **PKCS5Padding**
  - **SSL3Padding (**예약만 되어 있고, 실제 구현되어 있지 않음**)**
  - **OAEPWith<digest>And<mgf>Padding**

2. 웹, 자바스크립트

- Forge.cipher 객체 이용
- 암호 알고리즘
  - **AES** : AES-128, AES-192, or AES-256
  - **3DES** : 192비트
  - **DES** : 64비트
- 운영모드(modes of operation)
  - **ECB**, **CBC**, **CFB**, **OFB**, **CTR**, **GCM**
- 알고리즘과 운영모드를 함께 선택 →

https://github.com/digitalbazaar/forge#cipher

- AES-ECB
- AES-CBC
- AES-CFB
- AES-OFB
- AES-CTR
- AES-GCM
- 3DES-ECB
- 3DES-CBC
- DES-ECB
- DES-CBC

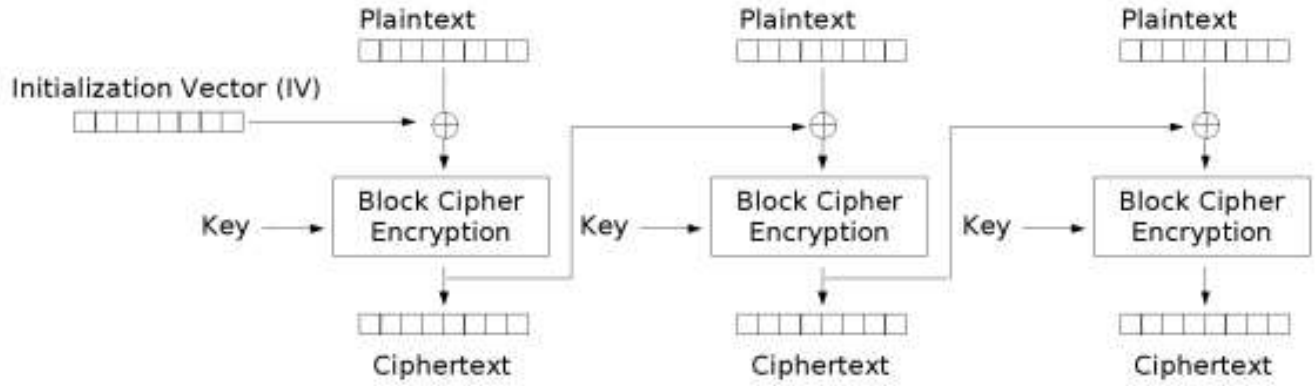
AES-CBC


```
var cipher = forge.cipher.createCipher('AES-CBC', key);
cipher.start({iv: iv});
cipher.update(forge.util.createBuffer(someBytes));
cipher.finish();
var encrypted = cipher.output;
// outputs encrypted hex
console.log(encrypted.toHex());

var decipher = forge.cipher.createDecipher('AES-CBC', key);
decipher.start({iv: iv});
decipher.update(encrypted);
decipher.finish();
// outputs decrypted hex
console.log(decipher.output);
```

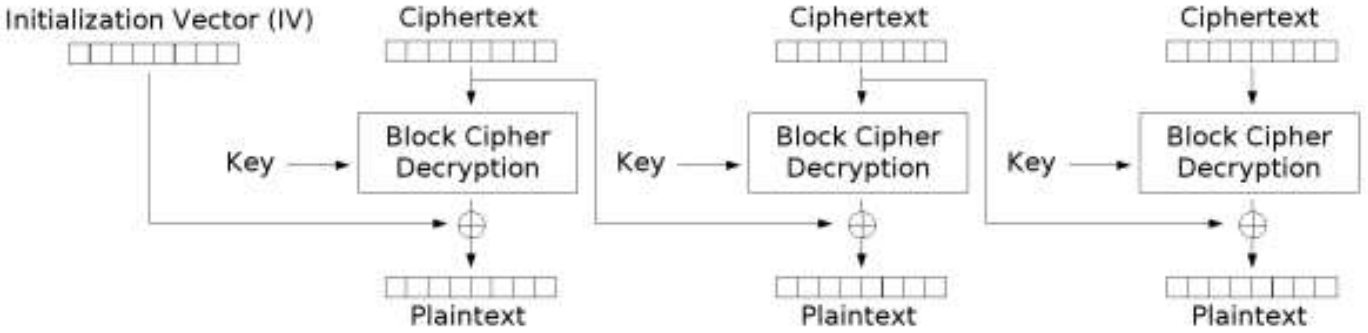
createCipher
Start
Update
finish

createDecipher
Start
Update
finish

CBC 모드



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

AES-CBC


```
var forge = require('node-forge');
var inputText = "Hello world - 헬로월드";
var someBytes = forge.util.encodeUtf8(inputText);

console.log('AES-128-CBC');
var keySize = 16; // 16 => AES-128, 24 => AES-192, 32 => AES-256
var key = forge.random.getBytesSync(keySize);
var iv = forge.random.getBytesSync(keySize);
console.log('- Key: '+forge.util.bytesToHex(key));
console.log('- iv: '+forge.util.bytesToHex(iv));
console.log('- Plaintext: '+forge.util.decodeUtf8(someBytes));
var cipher = forge.cipher.createCipher('AES-CBC', key);
cipher.start({iv: iv});
cipher.update(forge.util.createBuffer(someBytes));
cipher.finish();
var encrypted = cipher.output;
// outputs encrypted hex
console.log('- Encrypted: '+encrypted.toHex());

var decipher = forge.cipher.createDecipher('AES-CBC', key);
decipher.start({iv: iv});
decipher.update(encrypted);
decipher.finish();
console.log('- Decrypted: '+decipher.output);
console.log();
```

한글 처리를 위한 UTF-8 인코딩 적용

난수 키생성

AES-128 ➔ 128/8 = 16
AES-192 ➔ 192/8 = 24
AES-256 ➔ 256/8 = 32
DES ➔ 64/8 = 8
3DES ➔ 192/8 = 24

Key : 공유키
Iv : 초기화 벡터

forge.util.createBuffer 이용

출력 암호문은 byte array
화면으로 표시하려면 toHex() 함수 이용

```
F:₩AppliedCrypto₩mongochat>node cipher.js
AES-128-CBC
- Key: 6eb12763d42b4ee5eaf3a1db9700a15b
- iv: e53fd45638dcc66cbc7a847ee77a3e56
- Plaintext: Hello world - 헬로월드
- Encrypted: 9c718126501be3ba1b992d5a966bdf1c18bf24e7f644f35eedd096c124a1ad41
- Decrypted: Hello world - 헬로월드
```

파일 암호화 : AES-CBC

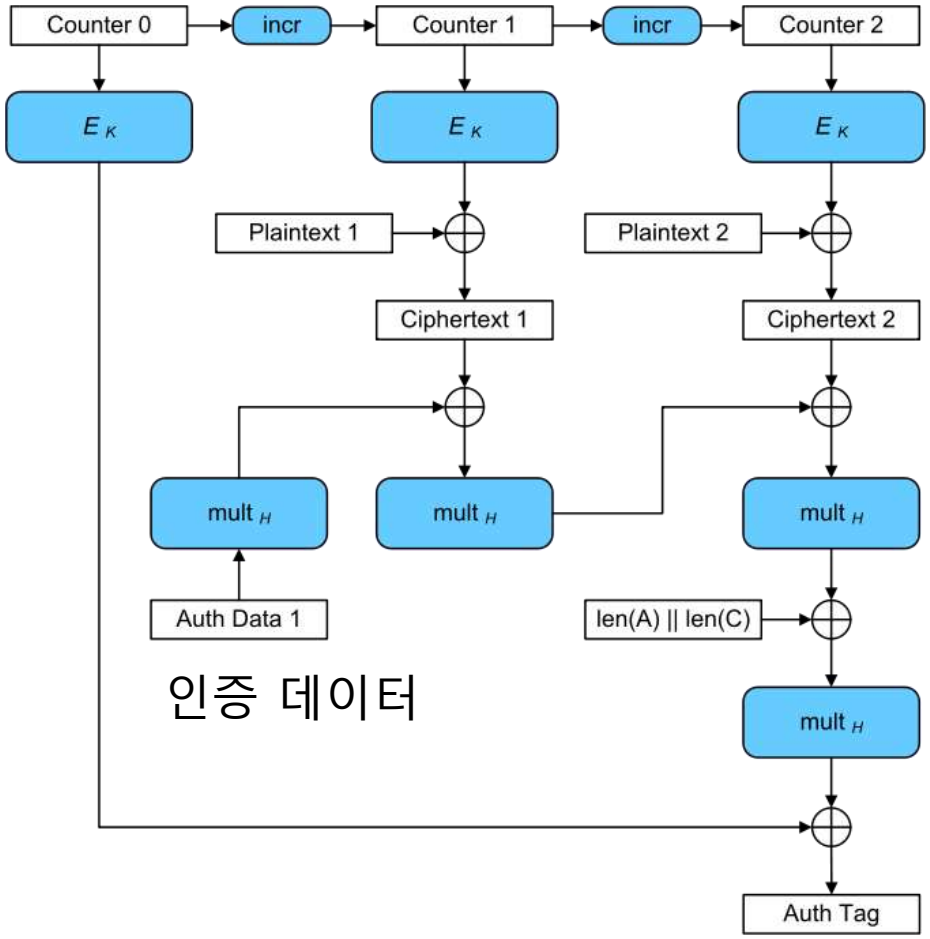
```
const forge = require('node-forge');
const fs = require('fs');

var inputFile = "input.txt";
var encryptedFile = inputFile+'.enc';
var decryptedFile = encryptedFile+'.dec';

// 송신측
var input = fs.readFileSync(inputFile, {encoding: 'binary'});

console.log('AES-128-CBC');
var keySize = 16; // 16 => AES-128, 24 => AES-192, 32 => AES-256
var numIterations = 100;
var salt = forge.random.getBytesSync(8);
// 패스워드 기반 암호화
var key = forge.pkcs5.pbkdf2('password', salt, numIterations, keySize);

var iv = forge.random.getBytesSync(keySize);
console.log('- Key: '+forge.util.bytesToHex(key));
console.log('- iv: '+forge.util.bytesToHex(iv));

var cipher = forge.cipher.createCipher('AES-CBC', key);
cipher.start({iv: iv});
cipher.update(forge.util.createBuffer(input, 'binary'));
cipher.finish();
var encrypted = cipher.output;

var output = forge.util.createBuffer();
// if using a salt, prepend this to the output:
if(salt !== null) {
  output.putBytes('Salted__'); // (add to match openssl tool output)
  output.putBytes(salt);
}
output.putBuffer(cipher.output);
fs.writeFileSync(encryptedFile, output.getBytes(), {encoding: 'binary'});
console.log("Ciphertext is saved in "+encryptedFile);
```

```
// 수신측
var input = fs.readFileSync(encryptedFile, {encoding: 'binary'});
var keySize = 16; // 16 => AES-128, 24 => AES-192, 32 => AES-256
var numIterations = 100;

// parse salt from input
input = forge.util.createBuffer(input, 'binary');
// skip "Salted__" (if known to be present)
input.getBytes('Salted__'.length);
// read 8-byte salt
var salt = input.getBytes(8);
var key = forge.pkcs5.pbkdf2('password', salt, numIterations, keySize);

var decipher = forge.cipher.createDecipher('AES-CBC', key);
decipher.start({iv: iv});
decipher.update(input);
var result = decipher.finish(); // check 'result' for true/false

if(result) {
  fs.writeFileSync(
    decryptedFile, decipher.output.getBytes(), {encoding: 'binary'});
    console.log("Decrypted text is saved in "+decryptedFile);
} else {
  console.log('file decryption error!!!');
}
```

운영모드별 특징

- ECB
  - Iv가 필요 없음
- CBC, CFB, OFB, CTR
  - Iv가 필요함
  - 모드 이름만 바꾸면 동일하게 실행됨
- GCM
  - 인증을 위해 추가적인 인증데이터 (authData) 필요 – 옵션
  - 인증결과로 tag를 생성
  - 복호화시 tag가 일치하는지 확인하고 결과를 출력

GCM 모드



평문

암호문

인증 데이터

인증 tag

GCM (Galois/Counter Mode)

GCM 모드


```
D:\WebDev\2022\crypto>node cipher-gcm.js
Algorithm-mode: AES-GCM
Plaintext: Hello world Hello world 한글메시지 要推出大做出了回应 ポリ
コントント 新品未開封品 مرحبا بك
Key: 5cddd8076c91f7020a7c2172fa136aa7
IV: 2ae785720c895c0f135fb3f8675480d8
Encrypted: 3a4e5937752ebfcc9a59aee81225b3445cbf0e26bdc4031ece8e066846
917320e2097477e896b571c0251c2f89c5436f41370d709bf0e2d2cc88caa9d7d8602
60b3a52ec881fb7ff6b8d1994fb5bf98a2c311f891edff6b4cd9899ac8b98e48acebc
ea9538085127823 9c1923fc2215f3b57793116e46408bfb99f1e801380c1
Tag: 0cd2bd551dcab337239d203b889bbe55
Deciphered: 48656c6c6f20776f726c642048656c6c6f20776f726c6420ed959caab
880eba994ec8b9ceca7b020e8a681e68ea8e587bae5a4a7e5819ae587bae4ba86e59b
9ee5ba9420e3839de383aae382b3e38383e38388e383b3e3838820e696b0e59381e69
caae9968be5b881e5938120d985d8a7d8b0d8a720d8aad8b1d98ad8afd89f
```

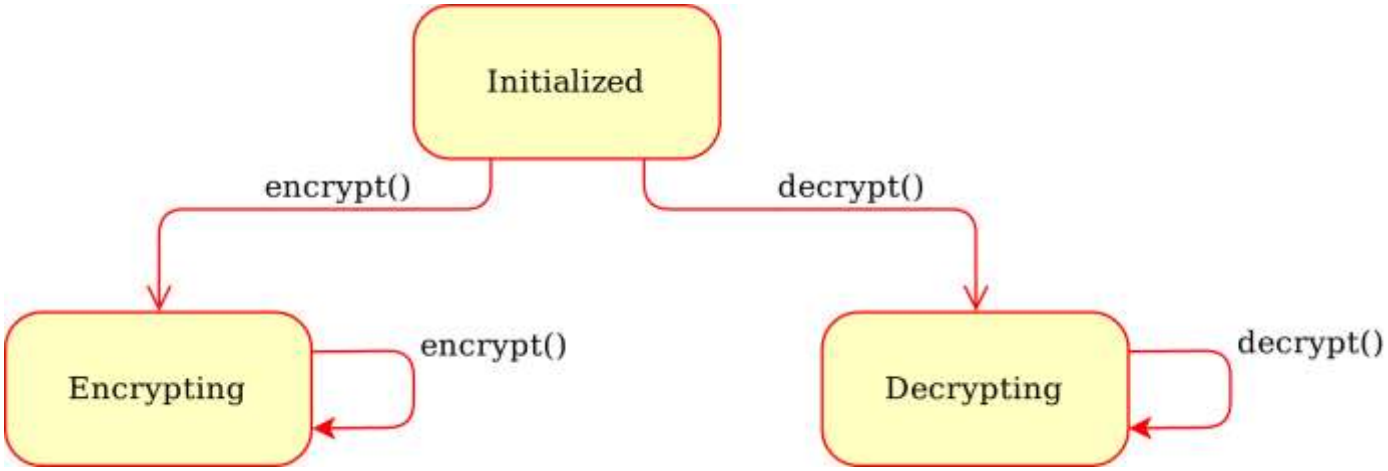
```
const forge = require("node-forge");

const algo = "AES-GCM";
// Modes: ECB, CBC, CFB, OFB, CTR, and GCM)
console.log("Algorithm-mode: " + algo);

const plaintext =
  "Hello world Hello world 한글메시지 要推出大
const authData = "binary-encoded string";

var someBytes = forge.util.encodeUtf8(plaintext

console.log("Plaintext: " + plaintext);

// generate a random key and IV
// Note: a key size of 16 bytes will use AES-128
var key = forge.random.getBytesSync(16); // 16
var iv = forge.random.getBytesSync(16); // 16

console.log("Key: " + forge.util.bytesToHex(key));
console.log("IV: " + forge.util.bytesToHex(iv));
```

```
// encrypt some bytes using GCM mode
var cipher = forge.cipher.createCipher(algo, key);
cipher.start({
  iv: iv, // should be a 12-byte binary-encoded string or byte buffer
  additionalData: "binary-encoded string", // optional
  tagLength: 128, // optional, defaults to 128 bits
});
cipher.update(forge.util.createBuffer(someBytes));
cipher.finish();
var encrypted = cipher.output;
var tag = cipher.mode.tag;
// outputs encrypted hex
console.log("Encrypted: " + encrypted.toHex());
// outputs authentication tag
console.log("Tag: " + tag.toHex());

// decrypt some bytes using GCM mode
var decipher = forge.cipher.createDecipher(algo, key);
decipher.start({
  iv: iv,
  additionalData: "binary-encoded string", // optional
  tagLength: 128, // optional, defaults to 128 bits
  tag: tag, // authentication tag from encryption
});
decipher.update(encrypted);
var pass = decipher.finish();
// pass is false if there was a failure (eg: authentication tag didn't match)
if (pass) {
  // outputs decrypted hex
  console.log("Deciphered: " + decipher.output.toHex());
}
```

3. 파이썬 – 대칭키 암호

https://pycryptodome.readthedocs.io/en/latest/src/cipher/cipher.html
https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html

cipher=Crypto.Cipher.<algorithm>.new(key, mode)

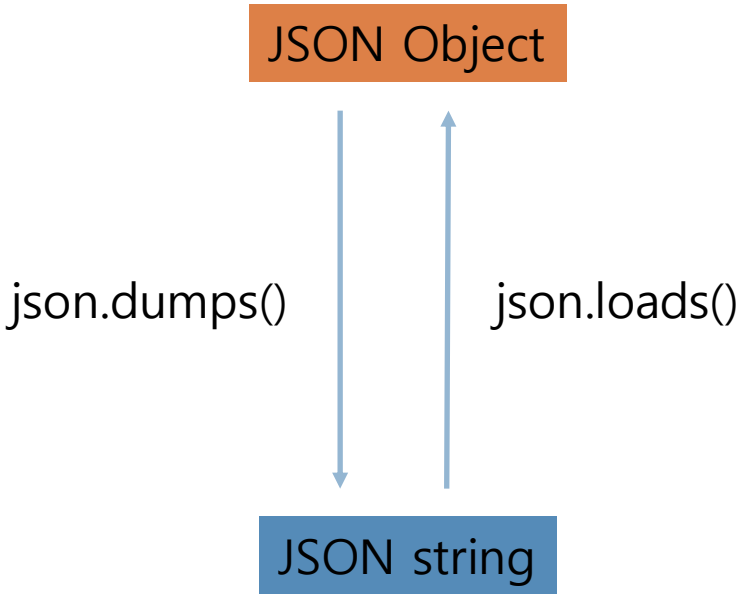


cipher.encrypt()

cipher.decrypt()

JSON

- JSON (JavaScript Object Notation)




```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

Base64 인코딩

- Base64 인코딩
  - 8비트 이진 데이터(예를 들어 실행 파일이나, ZIP 파일 등)를 문자 코드에 영향을 받지 않는 공통 ASCII 영역의 문자들로만 이루어진 일련의 문자열로 바꾸는 인코딩 방식
  - 64 = 2^6은 화면에 표시되는 ASCII 문자들을 써서 표현할 수 있는 가장 큰 진법
  - 알파벳 대소문자(A-Z, a-z)와 숫자(0-9) 그리고 두 개의 기호("+", "/")로 이루어져 있으며, 패딩 문자로 "="를 사용
  - 바이너리 데이터 3바이트를 4개의 Base64 문자로 변환 (8*3=6*4)
  - 입력되는 데이터가 3바이트로 나누어 떨어지지 않는 경우에는 "="를 패딩 문자로 사용
  - 저장, 전송, 화면 표시 등에 편리. 그러나 데이터가 확대되는 단점

ASCII 와 Base64

ASCII (8비트)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

128	Ç	144	É	160	á	176	░	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	|	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	n
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Source: www.LookupTables.com

Base64 (6비트)

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Base64 인코딩

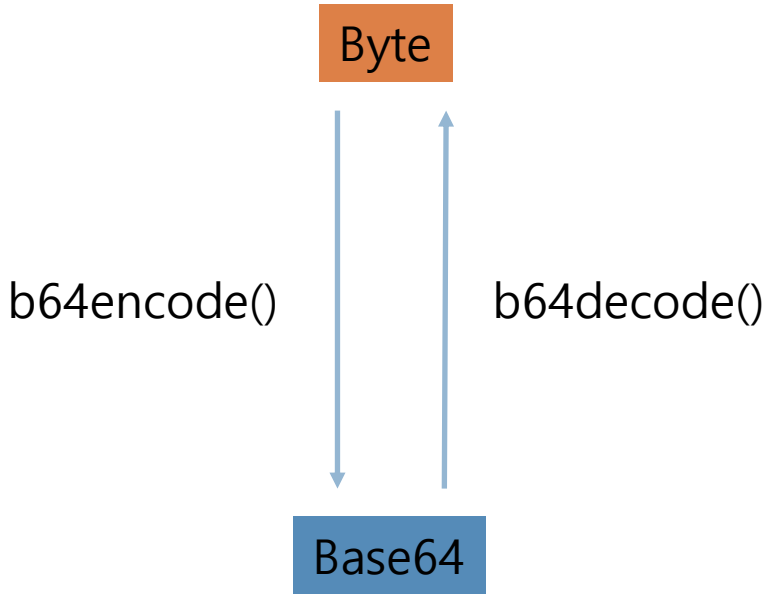
- ASCII 데이터 “Man”을 Base64 인코딩하면 “TWFu”로 변환

Text content	M								a								n							
ASCII	77								97								110							
Bit pattern	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
Index	19								22								5							
Base64-Encoded	T								W								F							
	u																							

- 온라인 Base64 인코딩 테스트 사이트
  - https://www.base64encode.org/

Python에서의 Base64 encoding


```
TWFuIGIzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGIz
IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udG1u
dWVkIGFuZCBpbmR1ZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=
```



인코딩 변환

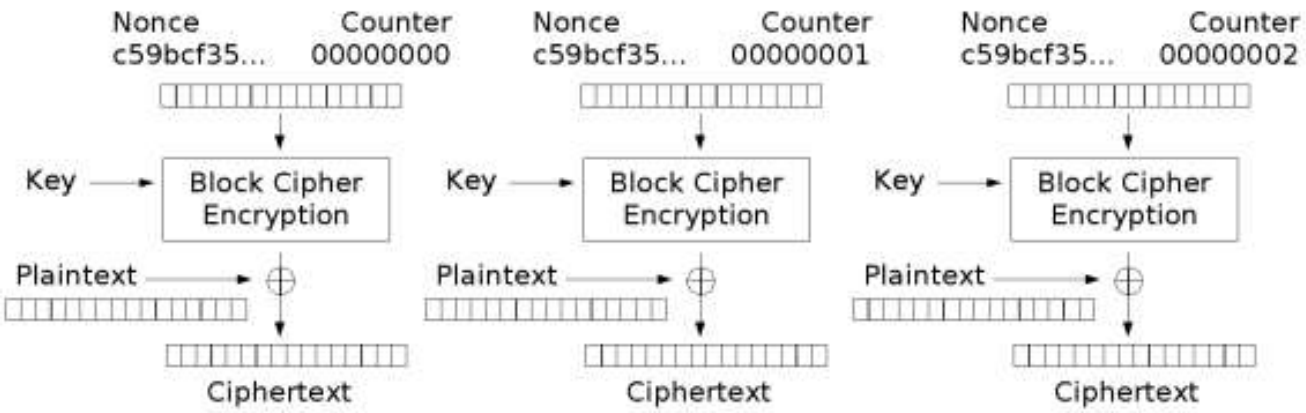
- String <--> byte
- Byte <→ string
- String ←> base64

```
from base64 import b64encode, b64decode

p = "Hello 한글"

# string <--> byte
print('string <--> byte')
print('string: ', p)
utf8 = p.encode('utf8')
print('bytes: ', utf8)
p1 = utf8.decode('utf8')
print('string: ', p1)
print()

# byte <--> base64
print('byte <--> base64')
print('byte: ', utf8)
b64 = b64encode(utf8)
print('base64: ', b64)
b64d = b64decode(b64)
print('byte: ', b64d)
print()

# string <--> base64
print('string <--> base64')
print('string: ', p)
b = b64encode(p.encode('utf8'))
print('base64: ', b)
p1 = b64decode(b).decode('utf8')
print('string: ', p1)
print()
```

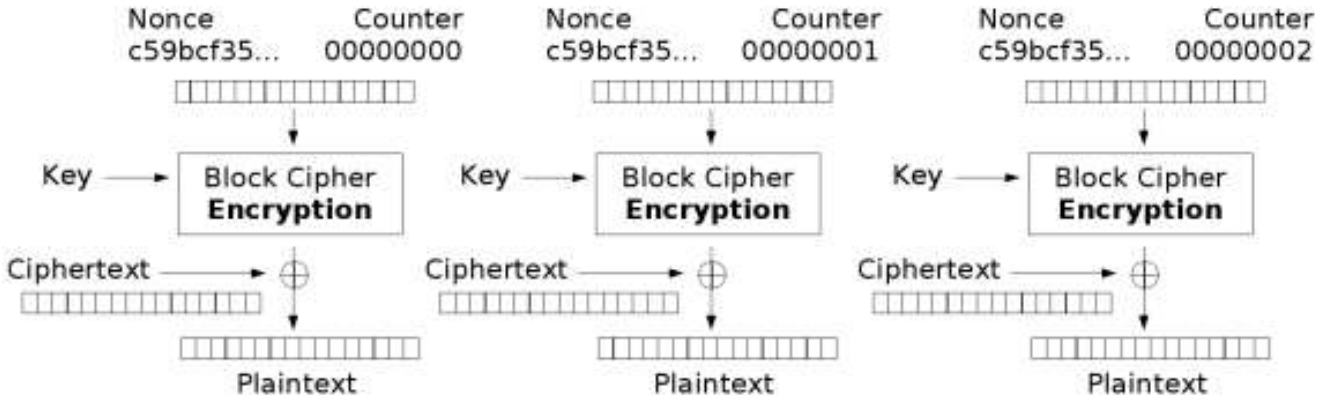
AES-CBC


```
import json
from base64 import b64encode, b64decode
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes

# 암호화
data = "secret한글".encode("utf-8")
key = get_random_bytes(16)
cipher = AES.new(key, AES.MODE_CBC)
ct_bytes = cipher.encrypt(pad(data, AES.block_size))
iv = b64encode(cipher.iv).decode('utf-8')
ct = b64encode(ct_bytes).decode('utf-8')
result = json.dumps({'iv': iv, 'ciphertext': ct})
print("암호문: ", result)

# 복호화
try:
    b64 = json.loads(result)
    iv = b64decode(b64['iv'])
    ct = b64decode(b64['ciphertext'])
    cipher = AES.new(key, AES.MODE_CBC, iv)
    pt = unpad(cipher.decrypt(ct), AES.block_size).decode("utf8")
    print("The message was: ", pt)
except ValueError as KeyError:
    print("Incorrect decryption")
```

CTR (Counter) 모드



Counter (CTR) mode encryption



Counter (CTR) mode decryption

AES-CTR


```
import json
from base64 import b64encode, b64decode
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

# 암호화
data = b"secret"
key = get_random_bytes(16)
cipher = AES.new(key, AES.MODE_CTR)
ct_bytes = cipher.encrypt(data)
nonce = b64encode(cipher.nonce).decode('utf-8')
ct = b64encode(ct_bytes).decode('utf-8')
result = json.dumps({'nonce': nonce, 'ciphertext': ct})
print(result)

# 복호화
try:
    b64 = json.loads(result)
    nonce = b64decode(b64['nonce'])
    ct = b64decode(b64['ciphertext'])
    cipher = AES.new(key, AES.MODE_CTR, nonce=nonce)
    pt = cipher.decrypt(ct)
    print("The message was: ", pt)
except ValueError as KeyError:
    print("Incorrect decryption")
```

AES-CBC 파일 암호화


```
import json
from base64 import b64encode, b64decode
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes

# 송신측, 암호화
file_to_encrypt = 'mp4.mp4'
input_file = open(file_to_encrypt, 'rb')
output_file = open(file_to_encrypt + '.enc', 'wb')
key = get_random_bytes(16)
buffer_size = 65536  # 64kb

cipher = AES.new(key, AES.MODE_CBC)
output_file.write(cipher.iv)

buffer = input_file.read(buffer_size)
while len(buffer) > 0:
    ciphered_bytes = cipher.encrypt(pad(buffer, AES.block_size))
    output_file.write(ciphered_bytes)
    buffer = input_file.read(buffer_size)

input_file.close()
output_file.close()
```

```
# 수신측, 복호화
input_file = open(file_to_encrypt + '.enc', 'rb')
output_file = open(file_to_encrypt + '.dec', 'wb')

iv = input_file.read(16)

cipher = AES.new(key, AES.MODE_CBC, iv=iv)

buffer = input_file.read(buffer_size)
while len(buffer) > 0:
    decrypted_bytes = unpad(cipher.decrypt(buffer), AES.block_size)
    output_file.write(decrypted_bytes)
    buffer = input_file.read(buffer_size)

# Close the input and output files
input_file.close()
output_file.close()
```