기초 암호학

https://youtu.be/p5MGmxIuOds

김경호

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

암호화(Encryption)

- 암호화
  - 특정 데이터를 알아볼 수 없도록 만들어서 데이터의 기밀성(Confidentiality)을 보장
  - Ciphertext = E(K, P) -> 암호화(Encryption)
  - Plaintext = D(K, C) -> 복호화(Decryption)

  - 케르크호프스의 원리 -> 암호의 보안성은 키의 비밀성에 의존
    - 키의 비밀성만 지켜지면 암호 알고리즘은 알려져도 상관 X

  - 일회용 패드(One-Time Pad)
  - 대칭키 암호화, 비대칭키 암호화
  - 인증 암호화
  - 형태 보존 암호화
  - 동형 암호화

암호화(Encryption) | 일회용 패드(One-Time Pad)

- 일회용 패드(One-Time Pad)
  - 가장 안전한 암호화 방법
    - 공격자가 무한한 계산 능력을 가져도 평문을 알아낼 수 없음
  - 암호화 시 평문과 동일한 길이의 난수(Key) 생성하여 XOR
  - 난수인 Key값은 일회용 (재사용 시 보안성 X)
    - C1 = (P1 ^ K)
    - C2 = (P2 ^ K)
    - (C1 ^ C2) = (P1 ^ P2)
  - 암호화 할 때마다 평문과 동일한 길이 난수 Key 값을 생성해야 함
    - 평문과 동일한 길이의 Key를 생성하는 것은 매우 비효율적(시간적, 공간적)

암호화(Encryption) | 대칭키 암호 & 비대칭키 암호

- 대칭키 암호
  - 하나의 Key를 Client와 Server가 같이 사용
    - 안전한 Key 교환(Key Exchange) 알고리즘 필요
  - 암호화, 복호화에서 사용되는 Key 값이 같음
  - 비대칭키에 비해 속도가 빠름
  - 블록 암호, 스트림 암호
  - AES, DES, ARIA, LEA …

- 비대칭키 암호
  - 공개 키(Public Key)와 개인 키(Private Key) 2 개의 Key를 가짐
  - 암호화 = 수신자의 공개 키로 암호화 후 전송 -> 수신자는 본인의 개인 키로 복호화
  - 서명(Signature) = 송신자의 개인 키로 암호화 후 전송 -> 수신자는 수신자의 공개 키로 복호화
  - RSA, ECC(Elliptic Curve Cryptography)

암호화(Encryption) | 인증 암호, 형태 보존 암호, 동형 암호

- 인증 암호(Authenticated Encryption)
  - 암호문 + 인증값(Authentication tag)
    - 인증값 = Key를 사용하는 암호를 이용하여 만든 임의 값
  - 유효한 인증값 확인을 통한 메시지 무결성(Integrity) 보장
    - C+T = E(K, P)
    - 암호문 뒤에 오는 T 값이 올바른 값인지 확인하고 복호화

- 형태 보존 암호(Format-Preserving Encryption)
  - 암호화 하는 대상의 형태를 보존하면서 암호화
  - 특정 Format을 유지하면서 암호화 하는 Database에서 사용
    - Ex) 192.168.0.1 -> 775.434.236.157

- 동형 암호(Homomorphic Encryption)
  - 암호화된 평문에 데이터를 추가 및 변경할 경우 복호화 없이 암호문 추가 및 변경 가능
  - 암호화가 상당히 느리다는 단점 -> 활발히 연구 중

난수 발생기 | 무작위성, 엔트로피

- 무작위성(Randomness)
  - 암호의 보안유지를 위해 무작위성을 가진 값을 사용
  - 무작위 = 값의 모든 경우의 수가 같은 균등 분포
    - 128bit의 무작위 값인 경우 가능한 모든 키 값은 $\frac{1}{2^{128}}$

- 엔트로피(Entropy)
  - 난수에 존재하는 무작위를 측정한 값
    - 값이 클수록 난수성이 높음
  - 각 경우의 수의 확률을 이진 로그로 변환하여 더한 값
  - n bit의 난수 값의 최대 엔트로피는 n bit

난수 발생기 | 난수 발생기, 의사 난수 발생기

- 난수 발생기(Random Number Generator)
  - 무작위한 난수를 발생해주는 S/W, H/W
  - 아날로그의 엔트로피를 이용하여 난수를 생성
    - CPU 온도, Time, Noise, 센서, 마우스 움직임,…
  - 아날로그 값이기 때문에 엔트로피 측정 불가 및 환경 조작으로 인한 공격 가능
  - 생성 속도가 느림

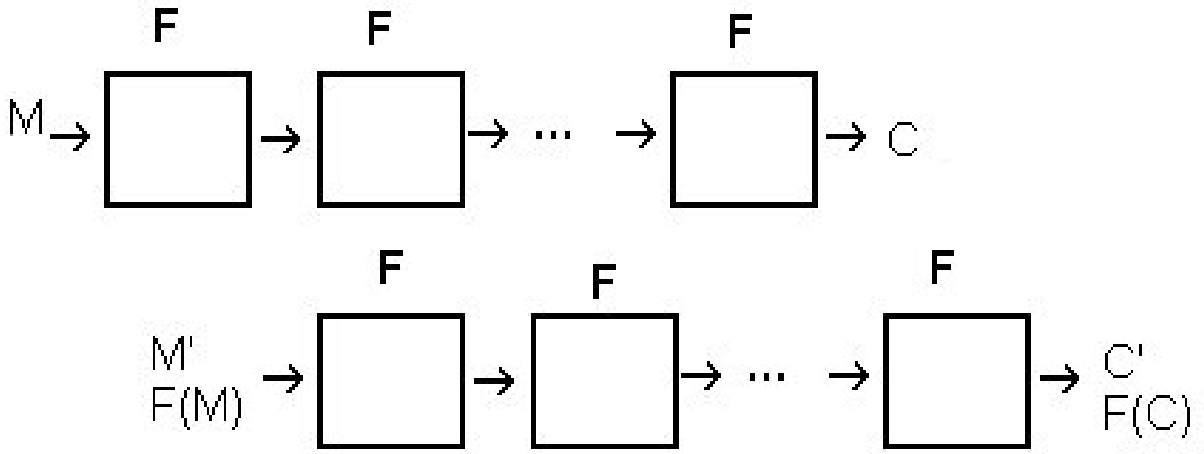
- 의사 난수 발생기(Pseudo-Random Number Generator)
  - RNG로 만들어낸 소수 값의 난수를 이용하여 난수 확장
  - 디지털 Source를 이용하여 난수를 확장 및 높은 엔트로피 유지
  - RNG로 만든 값인 엔트로피 풀을 입력값으로 완벽한 난수 출력
  - 역추적 저항성, 예측 저항성
    - 역추적 저항성(Backtracking Resistance) -> 이전에 생성된 비트 절대 다시 생성 X
    - 예측 저항성(Prediction Resistance) -> 이후 비트 예측 X
  - MS -> Fortuna, Unix -> /dev/urandom, Intel -> RDRAND

블록 암호

- 블록 암호
  - 평문을 특정 블록 단위로 암호화하는 대칭키 암호
  - 안전한 블록 암호 = 암호문이 완벽히 난수처럼 보임
    - 입력, 출력에서 어떠한 패턴도 없어야함
  - 키 없이는 어떠한 방법으로도 평문을 알 수 없음
  - 대부분 암호의 블록 크기는 64 ~ 512bit (128bit가 가장 많이 사용)
    - 너무 크면 연산 효율성 떨어짐
    - 너무 작으면 코드북 공격 가능
      - 발생 가능한 모든 입력 -> 출력 값 Table 생성 후 참조 연산
      - 32bit 블록의 경우 16GB의 메모리로 공격 가능 (64bit 이상은 안전)

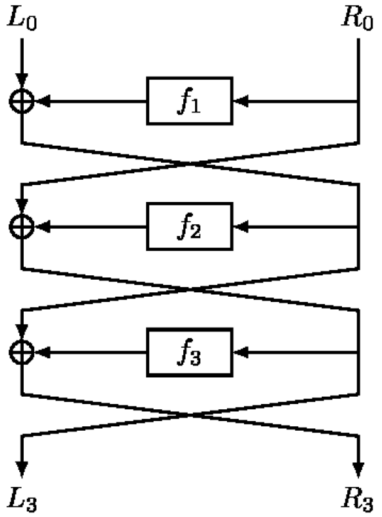
블록 암호 | 블록 암호의 특징

- 블록 암호의 특징
  - 어려운 하나의 연산이 아니라 단순한 연산을 일정 Round 동안 반복
    - 분석 및 구현이 간단
  - 라운드 마다 다른 키를 사용해야 함
    - Key 값을 이용하여 Round Key 생성 (Key schedule)
    - 같은 키 사용할 경우 Slide Attack 가능



블록 암호 | 블록 암호 설계

- SPN(Substitution – Permutation Network)
  - 대입과 치환 연산을 통해 혼돈, 확산
    - 혼돈(Confusion) = 입력에 복잡한 연산
    - 확산(Diffusion) = 변환을 통해 모든 비트에 영향
  - 대입은 S-box 참조 연산을 이용
  - 치환은 Rotation, 행렬 곱셈
  - AES
- Feistel Network
  - 64bit의 블록을 두개로 나눠서 한쪽 부분에 대입-치환 후 교환
  - 암호화에 사용되는 대입-치환 함수가 암호학적으로 강해야 함
  - DES
- ARX(Add, Rotation, XOR)
  - 최근의 경량 암호에서 많이 사용되는 방식
  - Add, Rotation, XOR 연산만을 사용해서 암호화
  - Speck, CHAM, LEA



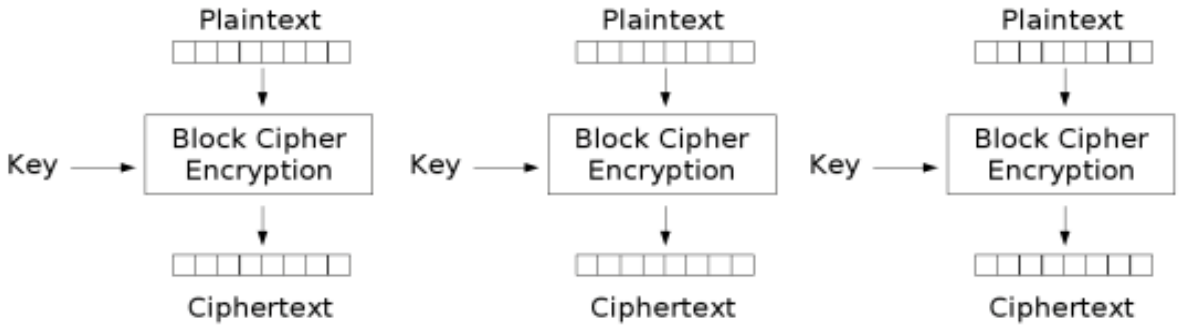
- 블록 암호 구현
  - S/W 암호 구현
    - 유동적인 암호 구현 가능(유지 보수 가능)
    - 속도가 느림
    - 부채널 공격 취약성(Cache Attack, Timing Attack,…)
  - H/W 암호 구현
    - S/W의 단점을 극복
    - 한번 제작 시 변경 불가능
    - AES-NI (Assembly 명령으로 AES 암호화 가능)

Instruction	Description
AESENC	Perform one round of an AES encryption flow
AESENCLAST	Perform the last round of an AES encryption flow
AESDEC	Perform one round of an AES decryption flow
AESDECLAST	Perform the last round of an AES decryption flow
AESKEYGENASSIST	Assist in AES round key generation
AESIMC	Assist in AES Inverse Mix Columns
PCLMULQDQ	Carryless multiply

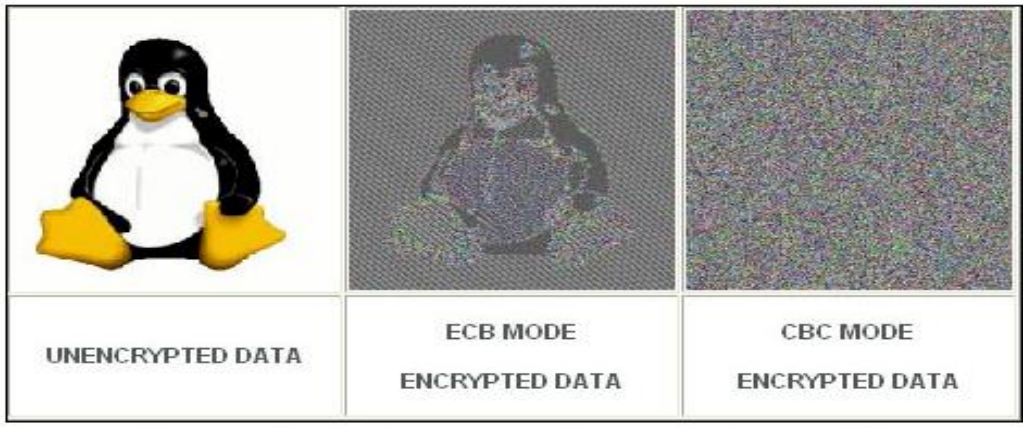
- 블록 암호 운영 모드
  - 암호는 치환 알고리즘 + 운영 모드가 결합
  - 암호의 보안성, 평문의 특징, 상황에 따라 운영 모드 선택
  - 운영 모드에 따라 속도, 안전성 등 결정
  - 대표적인 운영 모드 = ECB, CBC, CTR
  - AEAD(Authenticated Encryption with Associated Data)
    - 암호화 + 인증
    - GCM, CCM

블록 암호 | ECB(Electronic Code Block)

- ECB(Electronic Code Block)
  - 가장 Naïve한 암호 알고리즘
    - 동일한 평문 입력시 동일한 암호문 출력
  - 운영 모드 중 보안성이 가장 떨어짐
    - 평문의 특징이 암호문에서 드러남
    - 무작위 난수인 Nonce, IV 값을 사용하지 않기 때문에 무작위성 X

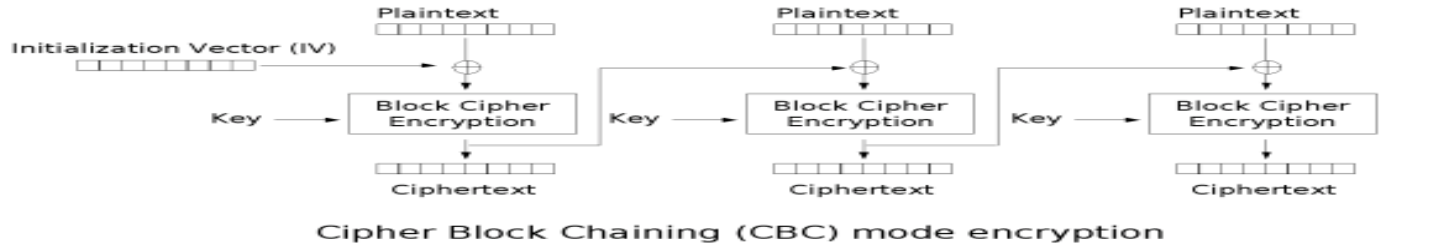


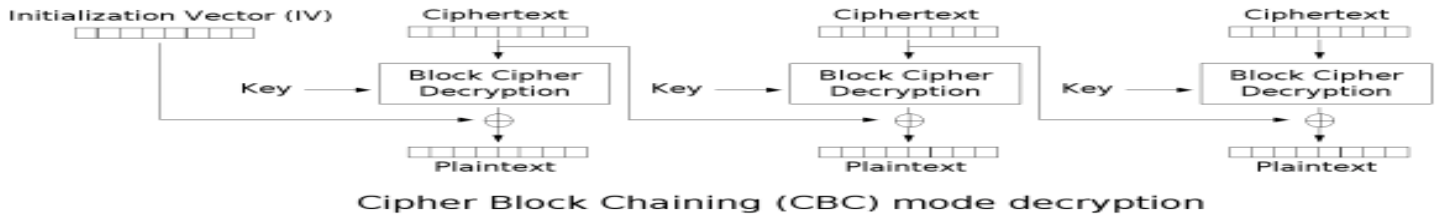
Electronic Codebook (ECB) mode encryption



블록 암호 | CBC(Cipher Block Chaining)

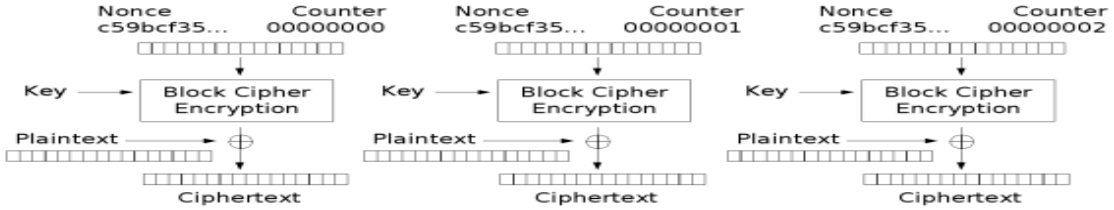
- CBC(Cipher Block Chaining)
  - 보안성이 가장 뛰어난 운영 모드
  - 메시지 인증 코드에서 더 많이 사용 (CCM)
  - IV(Initial Vector) 및 이전 암호문을 평문과 XOR하지 때문에 무작위성 보장
  - 이전 암호문을 이용하기 때문에 병렬처리 불가능



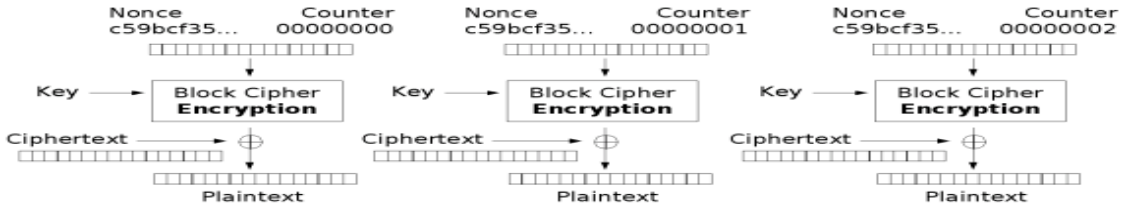


블록 암호 | CTR(Counter)

- CTR(Counter)
  - 96bit의 무작위 값인 Nonce와 32bit의 Counter 값으로 이루어진 IV 사용
  - Nonce 값은 일회용, Counter 값은 블록당 1씩 증가
  - 병렬처리 가능, IV 값이 정해지면 사전 연산 후 스트림 암호처럼 사용 가능
    - 키 스트림 생성 가능
  - 모든 암호화 모드 중 가장 빠른 암호화 시간



Counter (CTR) mode encryption



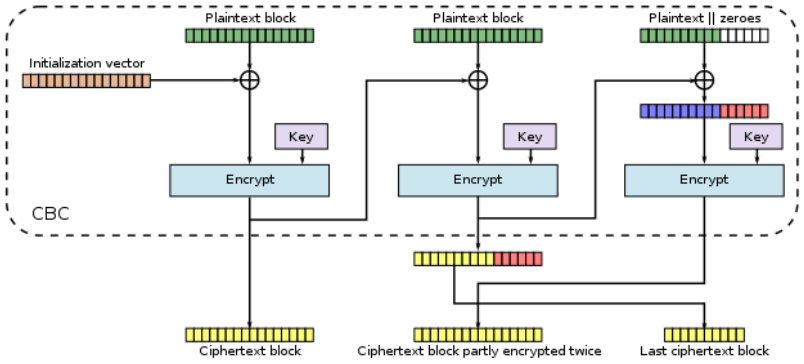
Counter (CTR) mode decryption

블록 암호 | Padding

- Padding
  - 16 Byte의 블록 암호에서 16 Byte 미만의 평문을 암호화 할 때 사용
  - 16 Byte을 채우기 위해 Padding
  - 평문이 1 Byte인 경우 나머지 15개의 Byte를 0x0f로 Padding
  - 평문이 6 Byte인 경우 나머지 10개의 Byte를 0x0a로 Padding

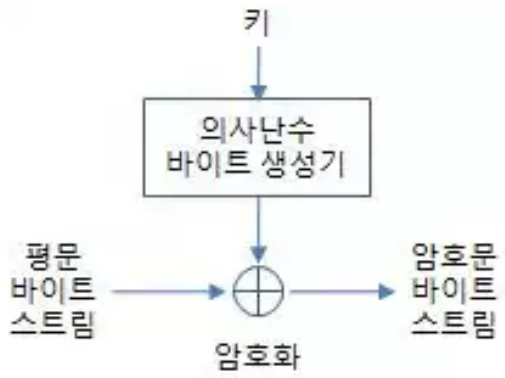
- Ciphertext Stealing
  - Padding에 비해 덜 쓰이고 복잡
  - 평문과 암호문의 길이가 같음, Padding Oracle Attack 방어
  - 구현이 상당히 어려움

- Padding Oracle Attack
  - CBC에서 사용하는 Padding 기법을 악용하여 오류 메시지를 이용한 공격
  - Padding 기법 특징을 이용해 지속적인 Query를 통해 암호문 공격



스트림 암호

- 스트림 암호
  - 비트 단위로 암호화
  - 키 스트림(의사난수) 생성 후 평문과 XOR
  - 블록 암호에 비해 빠른 암호화/복호화
    - But) 경량 암호의 등장으로 스트림 암호와 블록 암호의 연산 시간 비슷해짐
  - Feedback Shift Register를 이용한 H/W 방식
    - FSR의 값을 Shift해서 얻은 값을 이용
    - 선형 FSR, 비선형 FSR을 이용하여 다양한 암호 알고리즘 존재
  - S/W를 이용한 직접 구현 방식
    - RC4, Salsa20
  - CPU의 발전과 하드웨어 비용의 감소로 스트림 암호의 사용이 감소
  - 가장 많이 사용되는 스트림 암호는 블록 암호의 CTR 모드



해시 함수

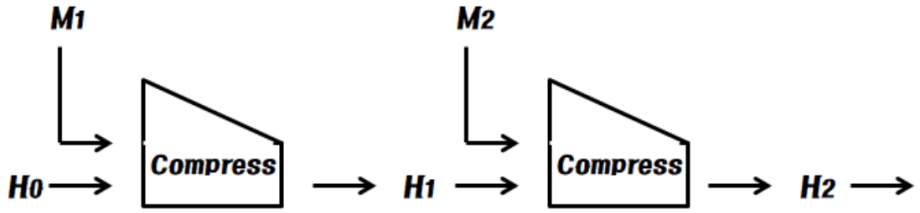
- 해시 함수
  - 임의의 길이의 데이터를 특정 길이의 데이터(Digest, Hash)로 변환
    - 입력 길이에 상관 없이 특정 길이의 Digest 출력
    - 메세지 -> 해시는 가능하지만 해시 -> 메세지 복구 불가능(One way function)
  - 입력 데이터의 1 bit만 바뀌어도 완전 다른 Digest 출력
  - 암호학의 다양한 부분에서 사용
    - 디지털 서명(Digital Signature)
    - 무결성 검증(Integrity Verification)
    - 패스워드 보호
    - 비트코인 작업 증명(Proof of Work)
  - 기밀성을 지키는 암호와 다르게 무결성을 지킴

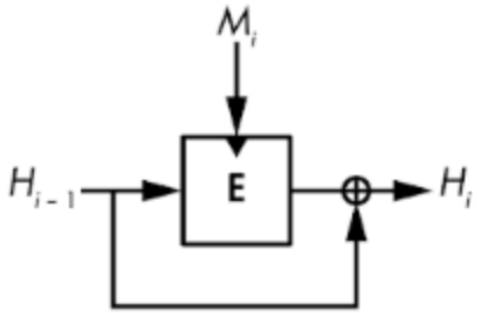
해시 함수 | 해시 함수 보안성

- 해시 함수 보안성
  - 해시 함수 출력의 비예측성
    - 입력이 조금만 달라져도 전혀 다른 값 출력(예측 불가)
    - 출력 값이 진정한 난수처럼 보여야함(패턴, 특징 X)
  - 역상 저항성(Preimage Resistant)
    - Hash(M) = H (H의 역상은 M)
    - 제 1 역상 저항성
      - 공격자가 H에 대한 역상인 M을 절대 찾을 수 없는 것 (찾아도 맞는지 확인 불가)
    - 제 2 역상 저항성
      - 동일한 H 값을 출력하는 M 값을 찾을 수 없는 것
  - 충돌 저항성(Collision Resistant)
    - 해시 충돌 = 다른 메세지가 같은 해시 값을 출력 (비둘기집 원리)
    - 해시 충돌을 일으키는 다른 메세지를 찾을 수 없는 것
  - 생일 공격 & Rho method

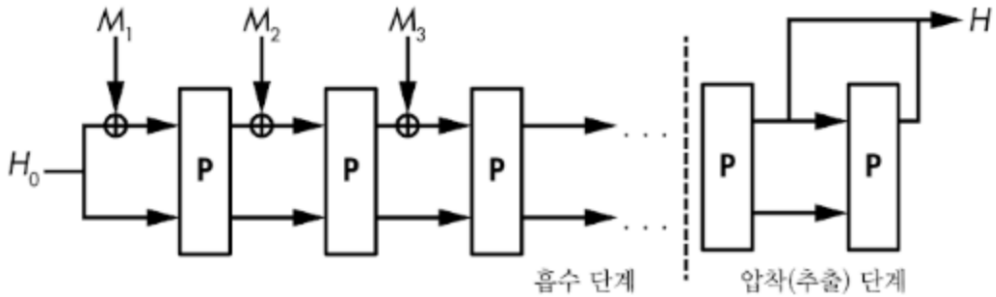
해시 함수 | 해시 함수 설계(머클-담고르 구조)

- M-D Construction(Merkle Damgard)
  - 입력을 더 작은 출력으로 변환하는 압축 함수(Compression function)를 이용
    - 압축 함수 보안성 = 해시 함수 보안성
  - H0에 들어가는 값은 IV 값
  - Padding -> 처음에 1, 이후 0, 마지막에 입력 메세지의 길이
    - Ex) 10101010|10000…0000|1000
  - 데이비스-마이어 Construction
    - 해시 함수에 사용되는 압축 함수를 블록 암호로 사용
  - SHA-1, SHA-2, MD5





- 스펀지 구조
  - 입력과 출력 크기가 같은 함수를 이용 (치환 함수 이용)
  - 해시에서 키를 이용한 블록 암호를 사용하는 기존 MD 방식 대신 치환 함수 이용(XOR 사용)
    - 치환 함수 P는 무작위 치환
  - H0에 들어가는 값은 IV 값
    - Padding -> 처음엔 1, 그 이후 0으로 채움
  - Absorbing phase – Squeezing phase (Keccak)

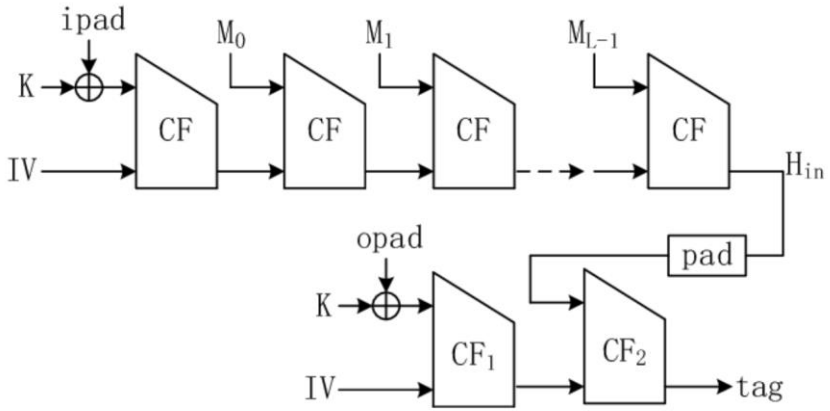


메세지 인증 코드

- 메세지 인증 코드(Message Authentication Code, MAC)
  - 키를 가진 해시 함수
  - T = MAC(K, M) -> T 값을 이용해서 메세지 무결성 및 인증성 보호
    - Ex) A, B가 메세지 교환시 동일한 Key를 이용한 인증값으로 메세지 무결성 인증성 보호
  - TLS, IPSec 등 보안통신에서 MAC 사용
  - MAC에서 사용하는 Key 유출 시 공격자가 위조(Forgery) 공격 가능
  - PRF(의사 난수 함수)
    - 메세지, Key를 이용해서 무작위해보이는 난수 출력해주는 함수
    - 암호 시스템에서 많이 사용 (Key derivation)
  - MAC보다 PRF가 보안성이 더 강하지만 최근에는 거의 비슷
    - HMAC-SHA-256

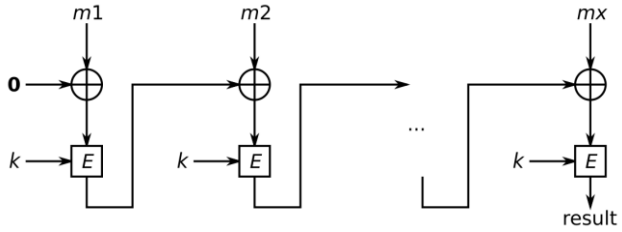
메세지 인증 코드 | HMAC

- HMAC
  - 해시 함수를 이용하여 구축한 MAC
    - 해시 함수의 보안성 = MAC의 보안성
  - 대부분의 보안 통신에서 HMAC을 사용 (SGX도 HMAC 사용)
  - 기존 해시에 Key붙이는 방식에 비해 훨씬 안전하고 빠름
  - Hash((K ⊕ opad) || Hash((K ⊕ ipad) || M))
  - opad = 0x5c, ipad = 0x36



메세지 인증 코드 | CMAC

- 암호 기반 MAC(Cipher-based MAC)
  - 해시 함수 대신 블록 암호를 이용하여 MAC을 생성
  - 암호 모드의 CBC 모드를 사용하여 MAC 생성
    - 메세지 M을 IV가 0인 값으로 CBC 모드로 암호화
    - 암호문의 마지막 블록을 인증값으로 출력
    - 두 개의 메세지-인증값 쌍으로 위조된 세 번째 메세지 생성 가능
  - 변형된 CBC-MAC
    - Key를 이용하여 2개의 Key 추가 생성
      - 임시값 L = E(0, K) 의 MSB가 0이면 K_1 = (L << 1), 1이면 K_1 = (L<<1) ⊕ 87
      - K_1의 MSB가 0이면 K_2 = (K_1 << 1), 1이면 K_2 = (K_1 <<1) ⊕ 87
    - 나머지 과정은 CBC와 동일하지만 마지막 라운드만 차이
      - 블록 크기가 딱 맞으면 K_1 을 XOR
      - 블록 크기가 부족하면 Padding 후 K_2 을 XOR



인증 암호화

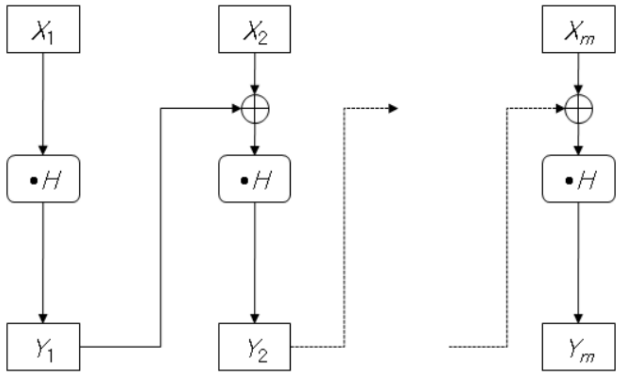
- 인증 암호화 (Authenticated Encryption)
  - 암호화 + MAC (암호화 & MAC, MAC 후 암호화, 암호화 후 MAC)
  - 메시지 기밀성 + 무결성 + 인증성 보호
  - 인증 암호화(AE) -> 암호문 + MAC 한번에 반환
    - 암호화 -> AD(K, P) = (C, T)
    - 복호화 -> AD(K, C, T) = P
  - 암호문을 복호화 하기 전에 MAC 값을 통해 무결성 및 인증성 확인
  - 암호화 과정만큼 MAC 생성하는 과정도 보안성 유지해야 함

인증 암호화 | AEAD(Authenticated Encryption with Associated Data)

- AEAD(Authenticated Encryption with Associated Data)
  - AEAD(K, P, A) = (C, A, T)
    - Associated Data는 평문 그대로 전송
  - 암호화는 하지 않고 인증값만 생성하고 싶은 Associated Data를 포함하는 인증 암호화
    - 네트워크 패킷 헤더
  - 최근 가장 많이 사용되는 암호화 모드
    - AES-GCM
    - NIST 경량 암호 기준
  - A, P 값을 안보내도 암호화 과정에서 문제 없음

인증 암호화 | AES-GCM(Galois/Counter Mode)

- AES-GCM(Galois/Counter Mode)
  - 전세계에서 가장 많이 사용되는 암호화 모드
    - NIST 표준, IPSec, TLS…
  - 암호화 모드 -> CTR Mode
    - IV -> 96 bit Nonce, 32 bit Counter
  - 메시지 인증값 -> GHASH (128bit Binary Multiplication)
    - H -> E(K, 0)
  - 암호화, 복호화 병렬화 가능 (스트림화 가능)
    - CTR Mode의 특성
  - Nonce 값 중복에 취약 (MAC 값 변조 가능)



인증 암호화 | AES-GCM(Galois/Counter Mode)

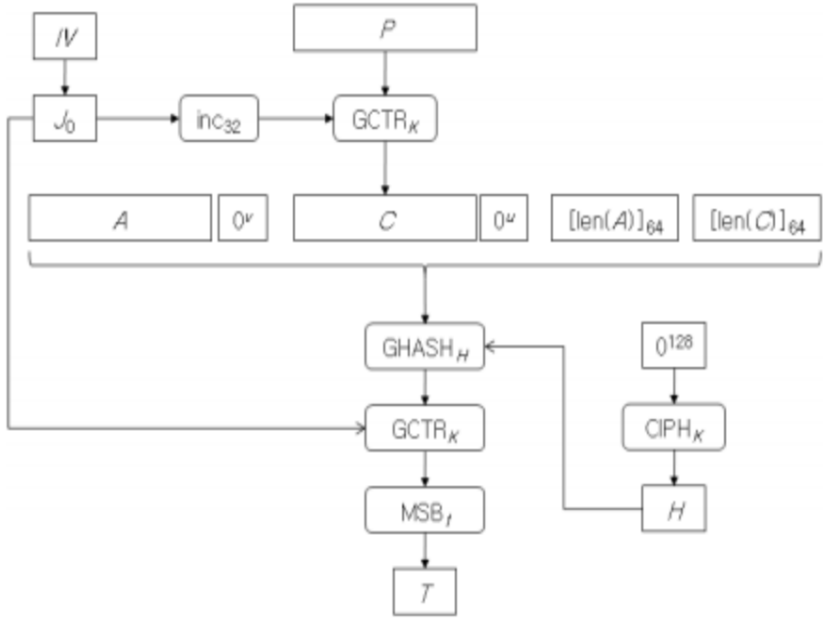
① $H = CIPH_K(0^{128})$

② $C=GCTR_K(inc_{32}(J_0),P)$

③ $S = GHASH_H\,(A \,||\, 0^v \,||\, C \,||\, 0^u \,||\, [len(A)]_{64} \,||\, [len(C)]_{64})$

④ $T=MSB_{Tlen}(GCTR_K(J_0,S))$

⑤ return (C,T)



Next..

- Finite Field, Group
- RSA
- 디피-헬먼
- 타원 곡선
- TLS

Q & A