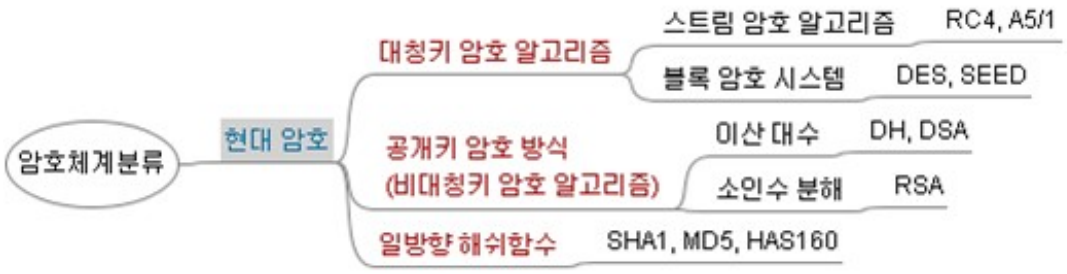
디지털보안학

[12주차. 암호2]

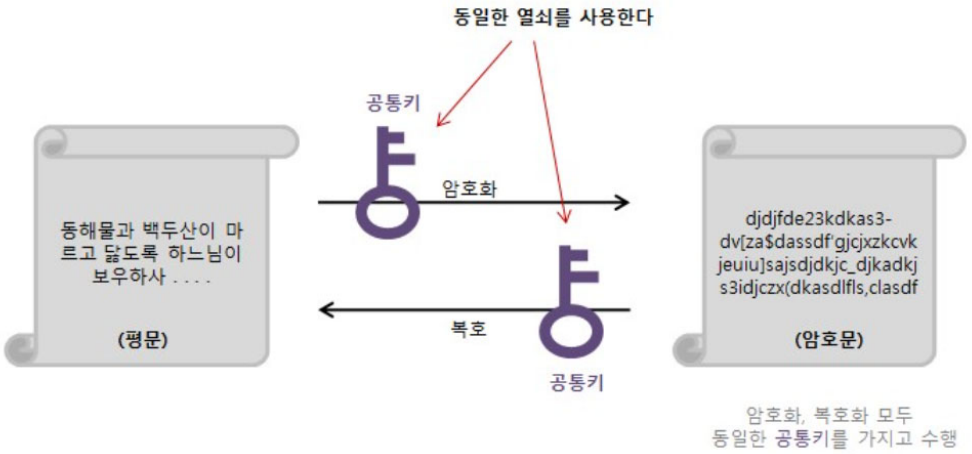
Digital Security

현대 암호 체계



공통키 / 대칭키

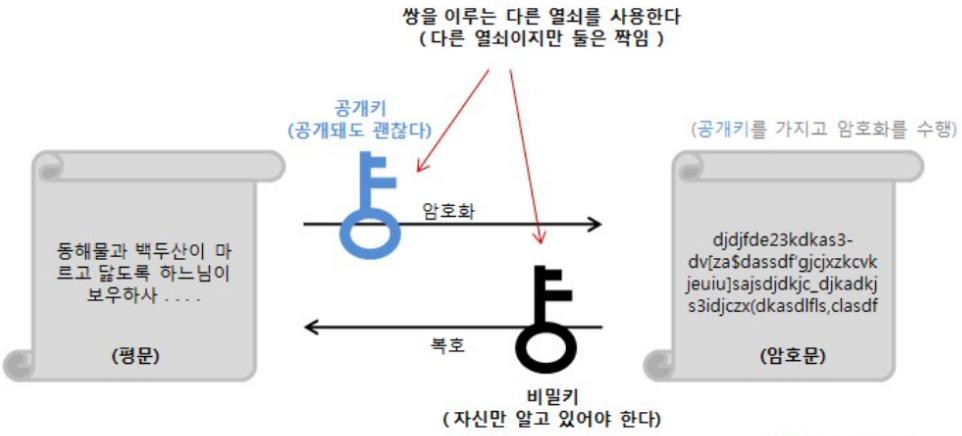
- 암호화 키 = 복호화 키



암호화, 복호화 모두
동일한 공통키를 가지고 수행

공개키 / 비대칭키

- 암호화 키 ≠ 복호화 키



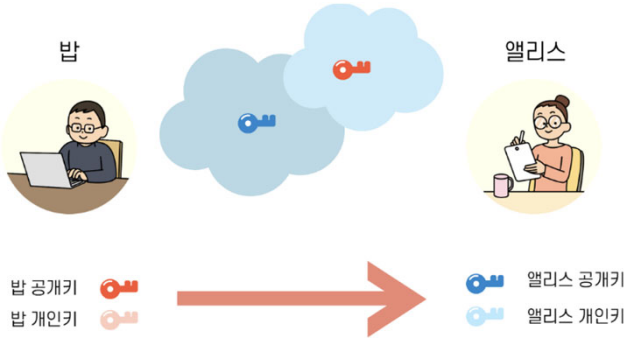
(암호화한 데이터는
암호화에 사용했던 공개키와 쌍을 이루는
비밀키를 가지고 있어야만 복호화할 수 있음)

공개키 / 비대칭키

비교 항목	비밀키 암호화	공개키 암호화
키 개수	한 개의 키를 사용	두 개의 키를 사용
키 보관 형태	비밀리에 보관	개인키는 비밀리에 보관, 공개키는 어디든지 배포
키 교환	키를 교환하는 것이 어려우며 위험하다.	공개키를 교환하는 것은 매우 쉽다.
키 길이	주로 64비트, 128비트 등 작은 길이	주로 512, 1024, 2048비트 등 큰 길이
암호화 속도	빠르다	느리다
암호화할 수 있는 평문의 길이	제한 없음	제한 있음
기밀성	가능함	가능함
인증	부분적 가능함	가능함
무결성	부분적 가능함	가능함
부인 방지	불가능	가능함

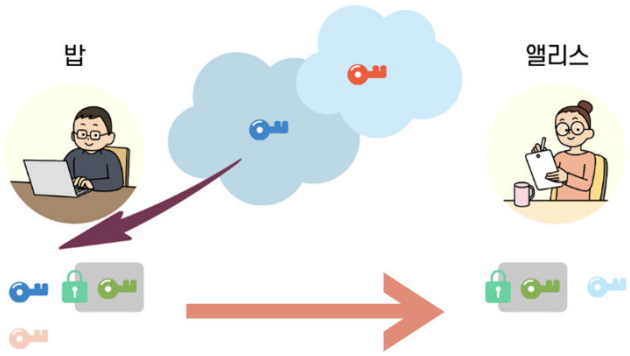
대칭키를 만들기 위한 공개키 암호화

- 대칭키를 사용해서 암호화
  - 밥과 앨리스는 각자 공개키와 개인키를 가지고 있음.
  - 밥과 앨리스는 자신의공개키를 오픈되어 있는 공간에 올려놓음
  - 공개된 공개키는 아무나 가지고 갈 수 있음



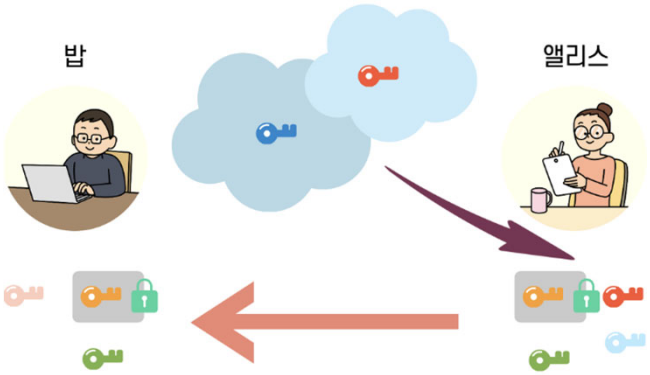
대칭키를 만들기 위한 공개키 암호화

- 필요시에 밥과 앨리스는 오픈된 공간에 있는 상대방의 공개키를 자신이 가져옴
- 밥은 앨리스에게 전달할 새로운 키를 생성하고 이 키를 앨리스 공개키로 암호화(S)해서 보냄

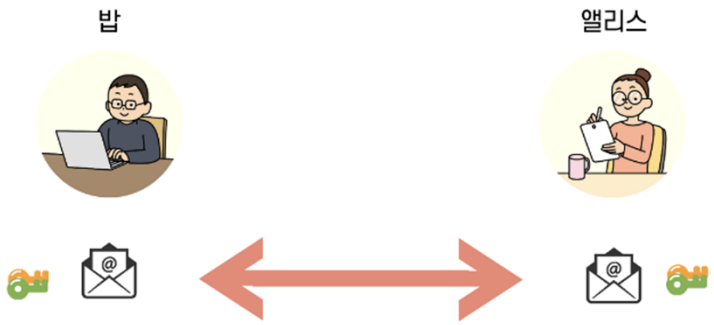


대칭키를 만들기 위한 공개키 암호화

- 이 암호화문(S)은 앨리스 비밀키로만 열수 있음
- 그리고 그 안에 있는 새로운 키는 둘만 알고 사용할 대칭키로 사용될 예정
- 앨리스가 밥이 보낸 키를 그대로 사용할 수 있지만, 밥의 새로운 키를 가지고 앨리스 새로운 키와 결합하여 대칭키를 만들어 밥의 공개키를 이용해서 암호화
- 이를 밥에게 전달하여 사용할 수 있음

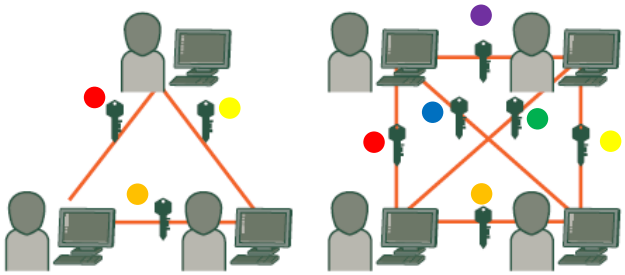


- 밥과 앨리스 사이에 대칭키를 가지고 메시지 교류 가능



공통키 / 대칭 암호 - 키

- 공통 키 암호 / 대칭 암호
  - 대칭 키 암호
  - 암호화와 복호화에 동일한 키 사용
  - 부하가 작아 고속으로 처리할 수 있음
  - 키가 유출되면 누구나 복호할 수 있기에 키 관리의 부담

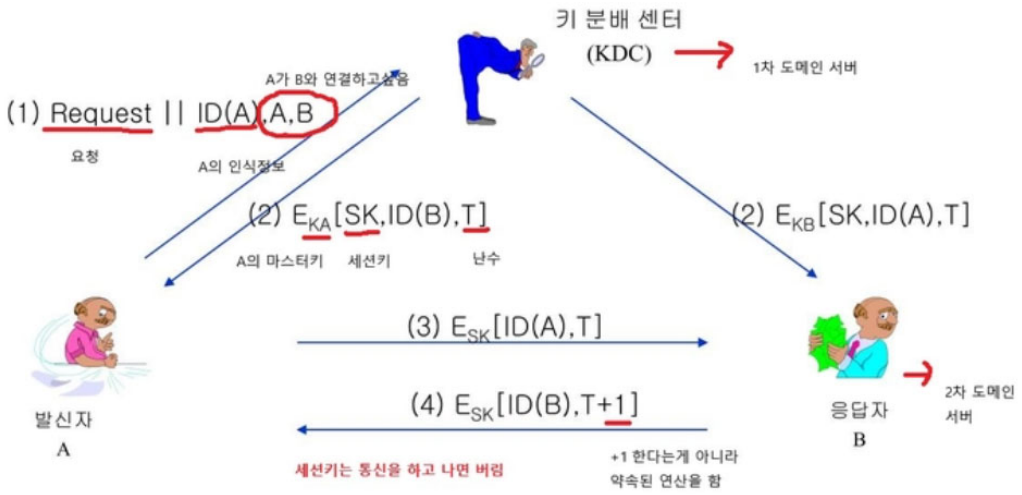


공통키 / 대칭 암호 - 키 분배/공유 문제

- 대칭 암호를 사용하려고 하면 바로 키 배송 문제 (key distribution problem)에 부딪치게 된다.
  - 앨리스가 자신의 메시지를 암호화할 때 사용했던 대칭키를 보내지 않으면 밥은 복호화 할 수 없다.
- 키 사전 공유
  - 키 배송 문제를 해결하기 위한 가장 간단한 방법은 “안전한 방법으로 키를 사전에 건네주는 것” . 이를 키의 사전 공유라 한다
- 사전 공유의 한계
  - 안전한 방법으로 통신 상대에게 키를 건네주지 않으면 안 된다
  - 인원이 많아지면 통신을 위한 키의 수가 방대해지는 것도 문제이다
  - 키를 그대로 보내 버리면 도청자 이브도 복호화 할 수 있다(키 배송 문제)

공통키 / 대칭 암호 - 키 분배/공유 해결 : 키 배포 센터

- 키 배포 센터(key distribution center; KDC)
  - 암호 통신 때마다 통신용의 키를 키 배포 센터에 의뢰해서 개인과 키 배포 센터 사이에서만 키를 사전에 공유
  - 키 배포 센터의 역할을 하는 컴퓨터를 지정
  - 구성원 전원의 키를 보존

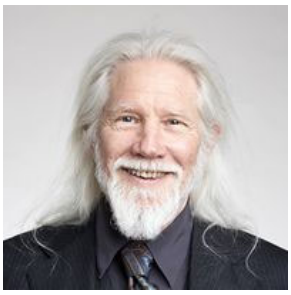


- 비밀키 분배에서 가입자 : 키분배센터와 각자의 마스터키를 나눠 가지고 있음
- Kerboros : 세션키를 이용한 키 분배 시스템

- 각 가입자(A, B)는 KDC와 각자 마스터 키를 가지고 있음
- (1) A가 키 분배 센터에 자신의 인식 정보와 A가 B와 연결하기 위해 세션키를 요구함
- (2) 키 분배 센터는 A의 마스터키로 A와 B에게 세션키와 상대의 신원 정보를 암호화 하여 배포 -> B는 A가 통신을 원한다는 것을 확인
- (3) A는 세션키를 이용하여 자신의 인식 정보와 난수를 암호화 하여 B에게 전송 -> B는 A의 신원을 확인
- (4) B는 세션키를 이용하여 자신의 인식 정보와 난수를 암호화 하여 A에게 전송 -> A는 B의 신원을 확인 => 통신 이후 세션키는 버림

공통키 / 대칭 암호 - 키 분배/공유 해결 : Diffie-Hellman의 키 교환

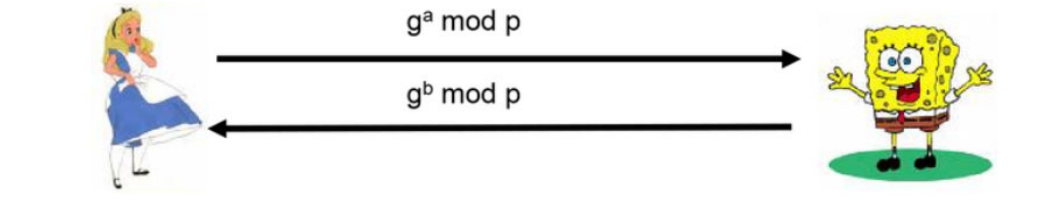
- 공개키를 교환하여 상호간에 사용할 비밀키 생성(1976년 개발)
- 양쪽의 통신 주체가 신뢰할 수 있는 제3자(KDC) 없이 대칭 세션키 교환하는 방식
- 비밀키는 암호문의 생성 및 평문의 복구를 위한 암호 및 복호키로 사용



Whitfield Diffie

Martin Hellman

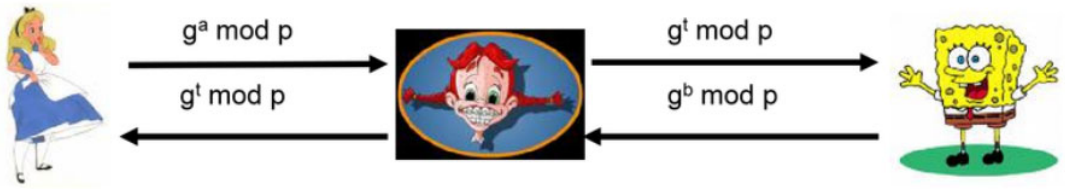
$$y = g^x \bmod p$$



$(g^a)^b \bmod p = g^{ab} \bmod p$

$(g^b)^a \bmod p = g^{ab} \bmod p$

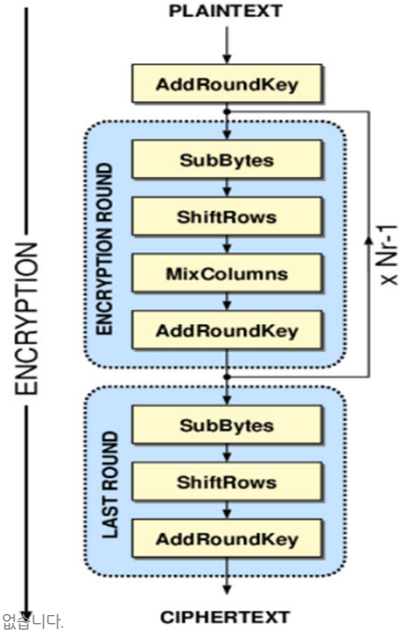
공통키 / 대칭 암호 - Diffie-Hellman의 키 교환 한계 : 중간자 공격 (man-in-the-middle attack)



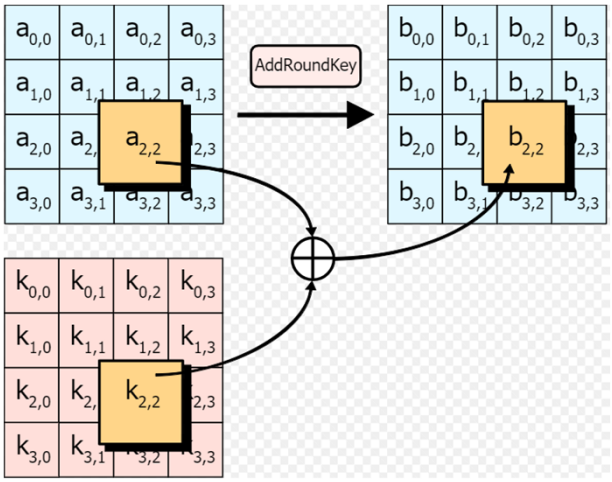
- Trudy shares secret g^{at} mod p with Alice
- Trudy shares secret g^{bt} mod p with Bob
- Alice and Bob don't know Trudy exists!

공통키 / 대칭 암호 - AES (Advanced Encryption Standard)

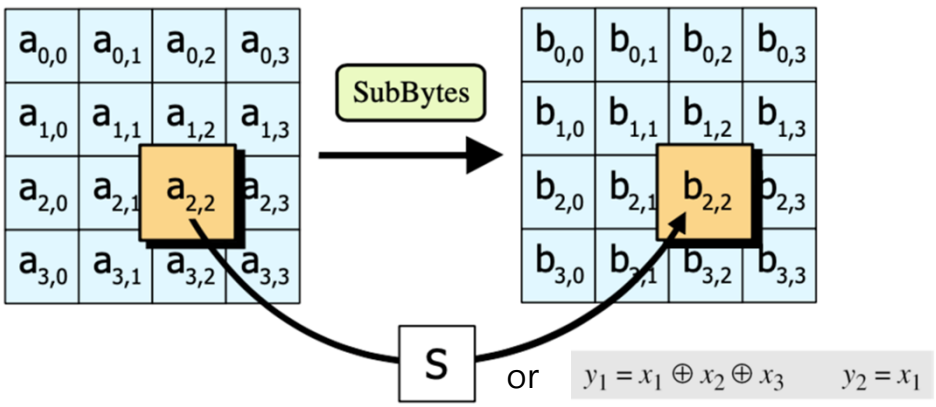
- NIST(표준 기술 연구소)에서 128bit 이상인 새로운 블록 암호 공모(2000)
- 가변 길이의 블록과 가변 길이의 키 사용 가능 (128 bit, 192bit, 256bit)
- 기본 구조
  - AES 는 DES와는 달리 0 ROUND 부터 시작 * 0 라운드는 AddRoundKey 만으로 구성 됨
  - Encryption Round 부분은 총 (Nr-1) 번 반복된다. * Nr = 10 / 12/ 14
  - Last Round 부분은 MixColumns 을 하지 않는다

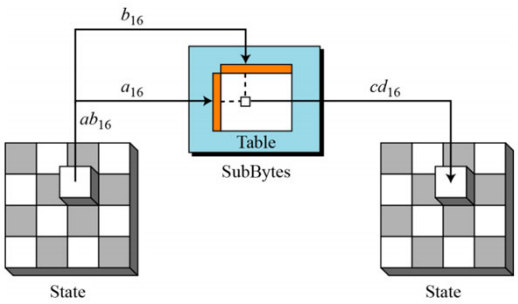


- AddRoundKey : 평문과 Ciper Key를 XOR 하는 과정



- SubBytes : Byte 단위의 비선형 연산으로 치환 테이블인 'S-BOX' 를 이용하거나 '직접 연산'을 수행하는 방식이 존재

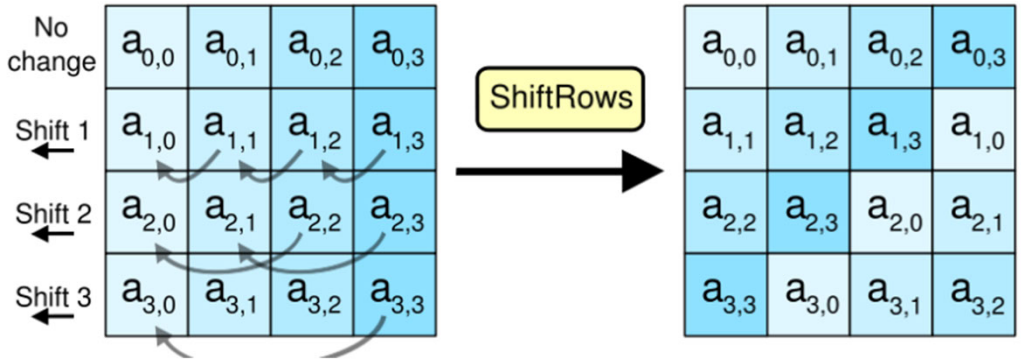


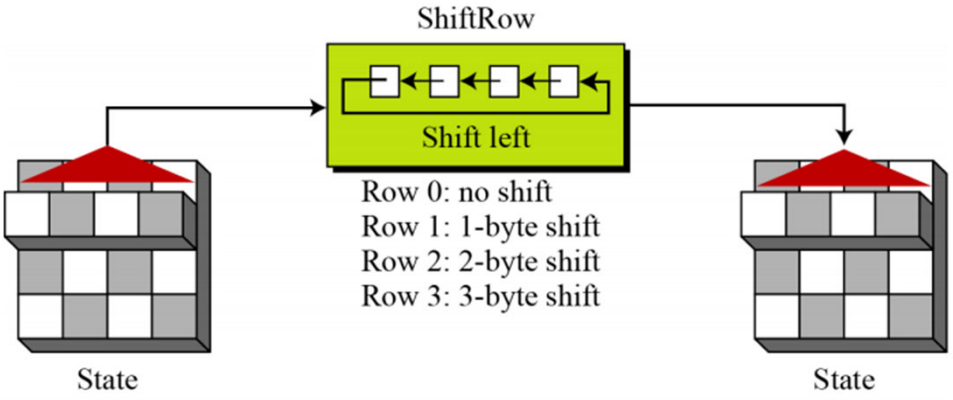


S-BOX

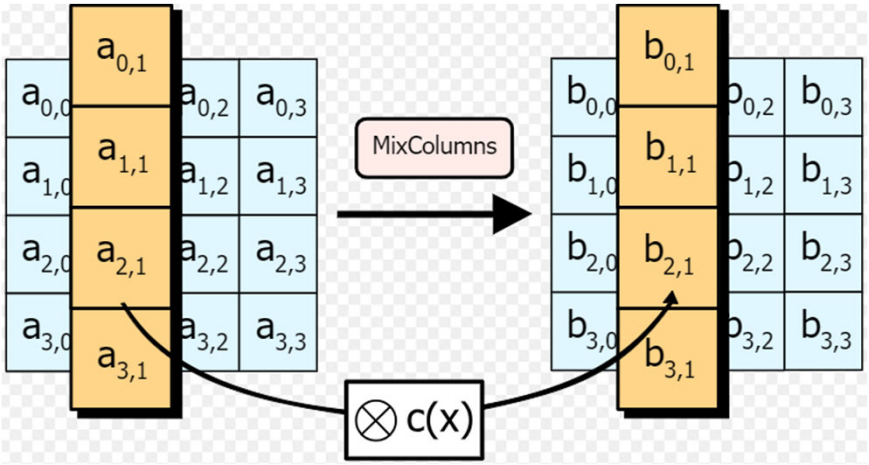
x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

- ShiftRows : Byte 단위의 쉬프트 연산 / 1행 - Leftshift : 0 / 2행 - Leftshift : 1 / 3행 - Leftshift : 2 / 4행 - Leftshift : 3





- MixColumns : GF(2^8) 위의 상수행렬 곱셈 연산



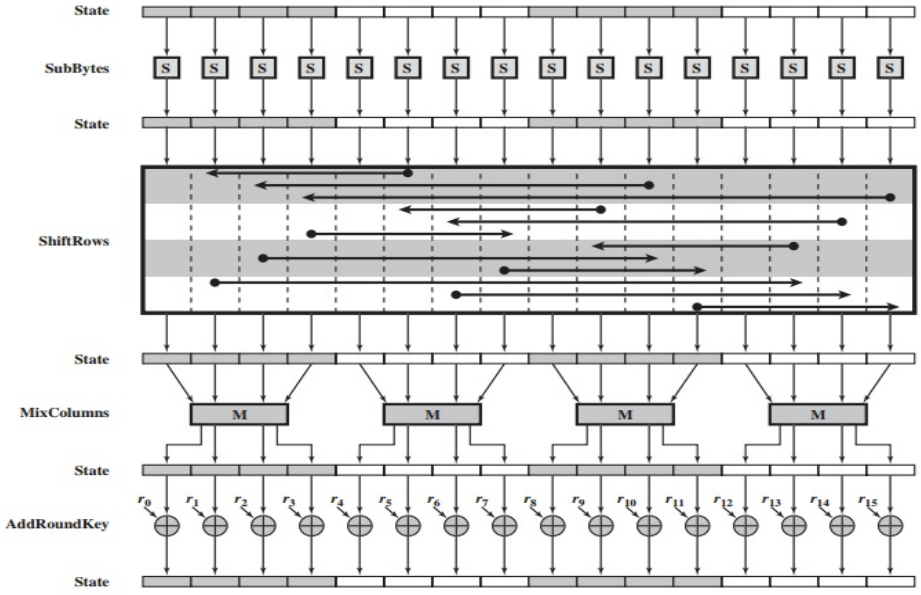
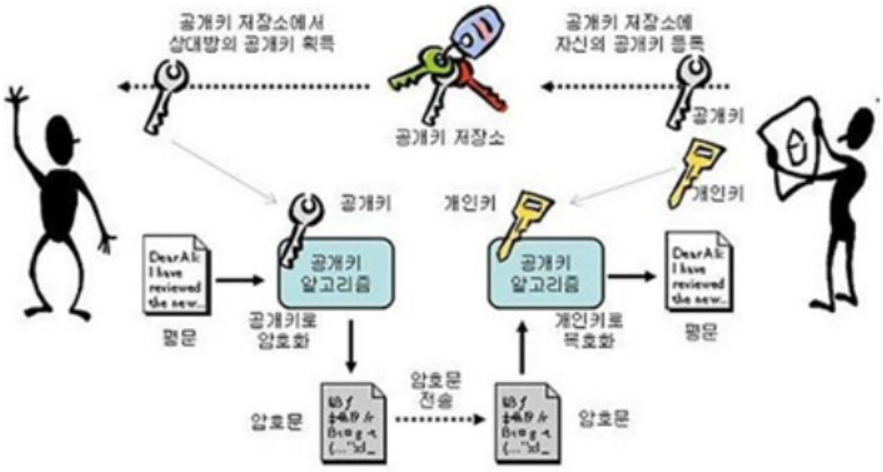


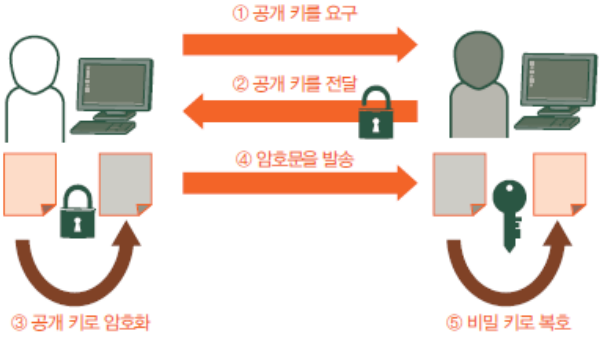
Figure 5.4 AES Encryption Round

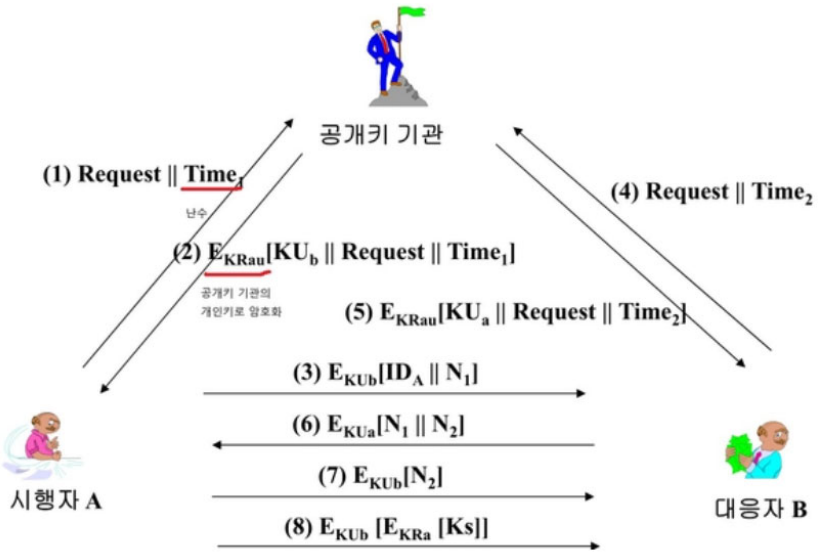
공개키 / 비대칭 암호 방식



공개키 / 비대칭 암호 – 키

- 공개 키 암호 / 비대칭 암호
  - 비대칭 암호
  - 암호화와 복호화에 서로 쌍을 이루는 각기 다른 키 사용
  - 이 중 하나는 공개 키로서 제3자에게 공개하여도 피해 없음
  - 본인임을 증명하는 데 사용





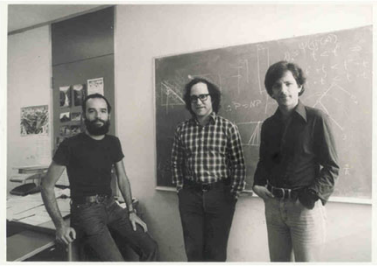
- 비밀키 분배에서 가입자 : 공개키 기관과 각자의 공개키를 나눠 가지고 있음
- 공개키 시스템은 공개키의 근원지를 입증하기 어려움
- 그래서 공개키 기관을 이용한 방법을 사용함

- (1) B의 공개키에 대한 요구를 난수와 함께 전송
- (2) B의 공개키와 1단계 메세지를 공개키 기관의 개인키로 암호화하여 전송
- (3) B의 공개키를 저장하고 A의 인식 정보와 난수를 B의 공개키로 암호화 하여 B에게 전송 -> 나 A인데 너랑 통신하고싶어 너 B 맞지?
- (4), (5) B도 (1), (2) 단계와 같은 방법으로 공개키 기관에 A의 공개키를 획득
- (6) A의 공개키로 A가 보냈던 난수와 B가 만든 난수를 암호화 하여 전송 -> 응 나 B 맞는데 너 진짜 A 맞지?
- (7) A는 B가 보낸 난수를 B의 공개키로 암호화 하여 전송 -> 응 나 진짜 A야
- (8) 서로의 확인이 끝난 후 A는 자신의 개인키로 통신에 필요한 세션키를 암호화한 것을 B의 공유키로 암호화한 후 B에게 전송 -> B는 자신의 개인키로 복호화한 후 A의 공유키로 한 번 더 복호화 세션키를 획득한다. => 통신이 끝난 후 세션키는 버려진다.

- ※공개키 상태 검증
- => 한 번 통신한 후 다시 검증하는 과정을 하기 귀찮기 때문에 공개키 기관에 물어본 후 통신함
- A : 기관아 저번에 썼던 B 공개키 계속 써도 되지? -> au : OK -> A : OK

공개키 / 비대칭 암호 - RSA (Rivest, Shamir and Adleman)

- 소인수분해의 어려움 이용
  - 21 = 3x7 / 143 = 11x13
  - 38724229 = ?



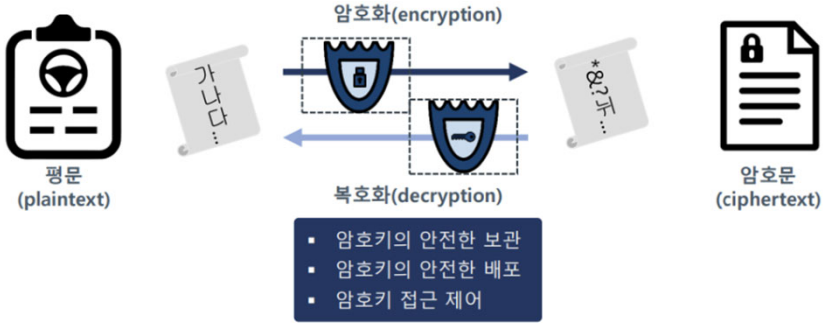
- $C = M^e \bmod N$
- $M = C^d \bmod N$
  - M은 평문, C는 암호문, N과 e는 공개키입니다.
  - 복호화에는 비밀키인 d가 사용됩니다.

- 키 생성 : 공개키와 개인키를 생성
  - 두 개의 큰 소수인 p와 q를 선택합니다.
  - p = 13, q = 17
  - p와 q의 곱인 N을 계산 : *N = p * q = 13 * 17 = 221*
  - 1부터 N-1 까지의 정수 중 N과 서로소인 정수의 개수를 구합니다.
  - 서로소란 두 숫자의 최대 공약수가 1이 되는 수를 말합니다.
  - N이 소수라면 N과 서로소인 정수는 N-1개, N이 소수의 곱이라면 N과 서로소인 정수는 (p-1) * (q-1) 이 됩니다.
  - 이 값을 N에 대한 오일러 파이(φ) 함수 값이라고 합니다.
  - φ(N) = (p-1) * (q-1) = (13-1) * (17-1) = 192
  - 1 < e < φ(N) 이고 φ(N)과 서로소인 공개키 e를 생성합니다.
  - 1 < e < 192,
  - e = 7 로 생성
- (d * e) mod φ(N) = 1 이고 0 <= d <= N 인 d를 구합니다.
  - (d * e) mod φ(N) = 1, (d * 7) mod 192 = 1,
  - d = 55 로 선택
- 생성된 값들 중 N과 e는 공개키로, d는 비밀키로 사용됩니다.

- 암호화
  - M은 평문, C는 암호문, N과 e는 공개키입니다.
    $C = M^e \bmod N$
  - $M = 123$, $N = 221$, $e = 7$
    $C = 123^7 \bmod 221 = 98$
- 복호화
  - 복호화에는 비밀키인 d가 사용됩니다.
    $M = C^d \bmod N$
  - $C = 98$, $N = 221$, $d = 55$
    $M = 98^{55} \bmod 221 = 123$

암호화 키 관리 필요성

- 암호화의 핵심은 “암호키”
  - 암호키 안전 관리(보관, 배포, 접근제어) 매우 중요
  - 가령, 암호화 키가 유실되면 아무리 강력한 알고리즘으로 암호화 하더라도 무용지물

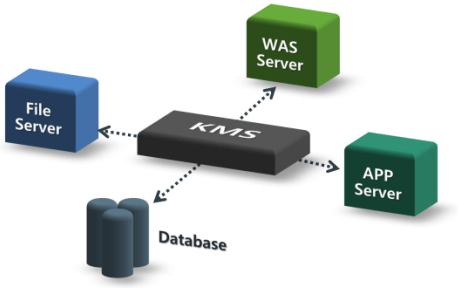


암호키 관리 권고사항

- 금융위원회 전자금융감독규정: 제 31조 (암호프로그램 및 키 관리통제)
- PCI-DSS 암호화 키관리 요구사항
  - (PCI(Payment Card Industry) DSS(Data Security Standard))
- 행정자치부 정보시스템 구축, 운영에 대한 지침: 소스코드내 비밀번호 하드코딩
- KISA 암호키관리 권고사항
- 정보보호관리체계 ISMS 암호키 관리 권고사항

암호화 키 관리 절차

- 암호화키 관리 절차는 암호키의 생명주기에 따라 “생성 > 저장 > 분배 > 백업 > 복구 > 폐기”의 각 단계별로 관리 필요
- KMS (Key Management System) 전용 암호화 키 관리 시스템
  - 안전한 암호화 키 관리를 위한 적용 하드웨어와 소프트웨어로 구성된 시스템
  - 다양한 서버에 필요한 키를 ‘중앙집중식’으로 관리하여 암호화키 사용기록이나 통계를 로그, 모니터링을 통해 감사를 수행



암호화 키 관리 절차

- 생성
  - 암호화 키는 데이터가 저장되는 장소와 물리적으로 분리하여 보관
  - 암호키를 생성하거나 변경하는 경우 기존키에 대한 백업 기능을 구현.
- 사용
  - Password나 암호화 키는 memory 상에 저장하지 않음 (static 변수에 저장금지)
  - 메모리상에 남겨진 암호나 키는 사용후 바로 초기화 (memory dispose)
  - 중요 정보를 암호화하는 암호키를 다시 암호화해서 별도의 디렉터리에 보관

암호화 키 관리 절차

- 폐기
  - 각 암호화 방식에 따라 대칭키, 비대칭키(개인키, 공개키) 등에 대해 각각 사용기간을 명시
  - 만약 기한이 종료된 경우 안전하게 폐기할 수 있는 절차와 방법을 프로세스로 관리
  - 사용기간이 넘었을 경우 사용자 인터페이스를 통해 기한이 지난 것을 알리고 새로운 키를 생성하도록 권장하는 기능을 추가

암호키 사용 유효기간 (NIST 권고)

키 종류		사용 유효기간	
		송신자 사용기간	수신자 사용기간
대칭 키 암호 알고리즘	비밀 키	최대 2 년	최대 5 년
공개키 암호 알고리즘	암호화 공개키	최대 2 년	
	복호화 개인키	최대 2 년	
	검증용 공개키	최소 3 년	
	서명용 개인키	최대 3 년	

암호화 키 관리수준

- 미국국립표준연구소 NIST 암호화 키 관리 레벨 4단계
  - 시스템에서 다루어지는 주요 정보에 따라 암호화 키 관리 레벨 결정

FIPS 140-2 레벨 분류

레벨	내용
Level 1	• 암호모듈에 대한 기본적인 설계보안항목만을 충족하여 최소한의 보안을 제공한다.
Level 2	• 침입자의 불법적인 접근을 방지하고, 침입 이후에 변조를 나타내는 증거를 제공함으로써 물리적인 보안메커니즘을 제공한다.
Level 3	• 강력한 변조 탐지 및 대응의 일환으로 침입을 감지하면 저장된 키를 삭제한다.
Level 4	• 암호모듈 외부의 전압이나 온도 등을 감지하여 수퍼쿨링(Supercooling) 등 환경의 이상변화 시, 암호키를 삭제한다.

* FIPS : Federal Information Processing Standard, 연방 정보 처리 표준