

Fundamental Crypto Algorithms

Chunghan Yi(chunghan.yi@gmail.com)

Date: 10/01/2020 ~

Doc. Revision: 0.9

Contents

- 1. Public Key Cryptography : 인증 & 키 교환
- 2. Symmetric Cipher Algorithm : 암복호화
- 3. Hash Algorithm : 데이터 무결성

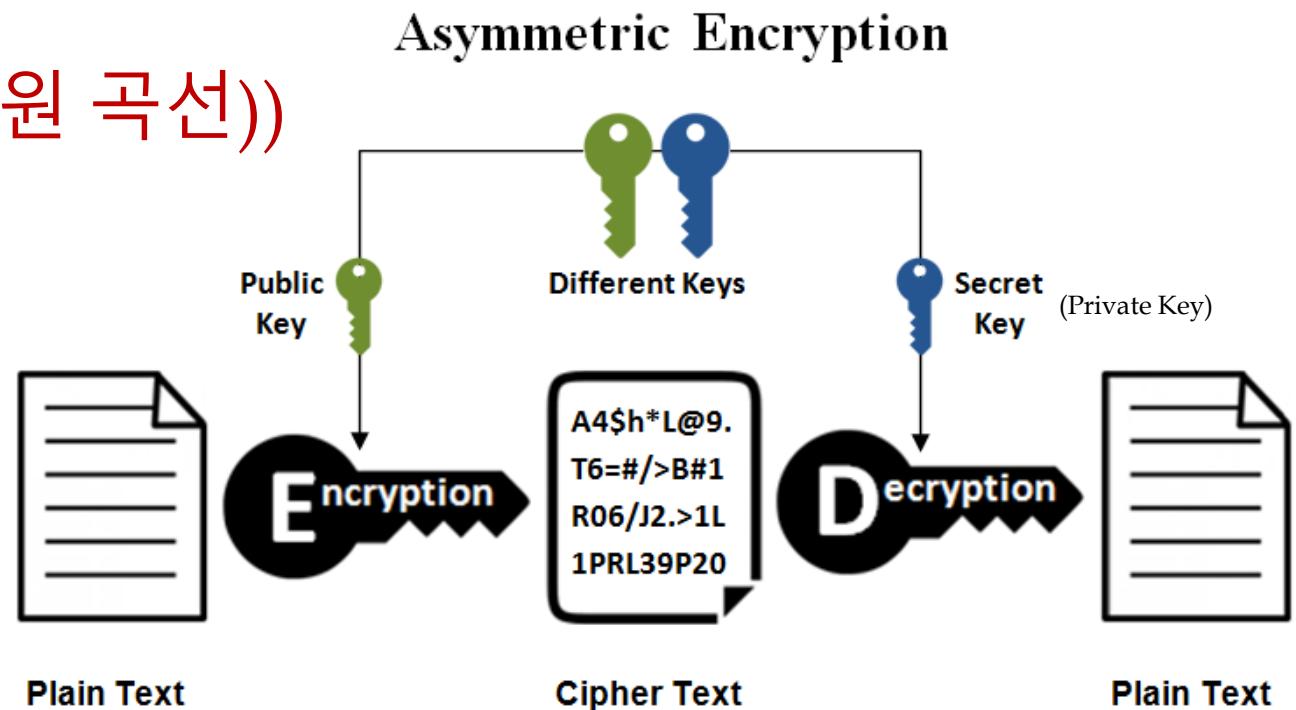
1. Public Key Cryptography(1) - *개요*

- 1) Diffie-Hellman
- 2) RSA
- 3) ElGamal
- 4) ECC(Elliptic Curve(타원 곡선))

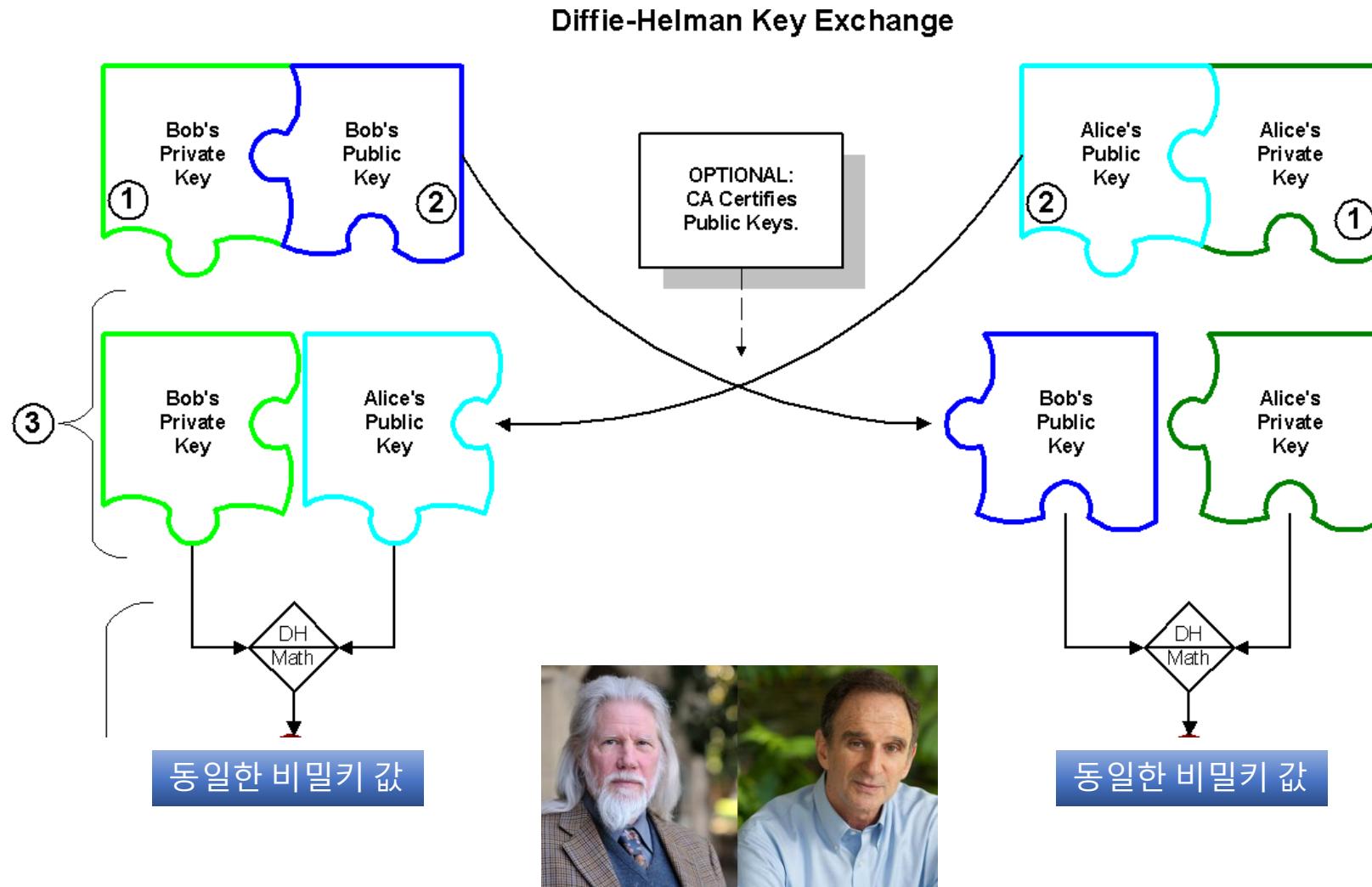
공개키/비대칭키 암호 알고리즘

<공개키 암호 알고리즘 사용 처>

- ✓ 인증
- ✓ 부인 방지
- ✓ 정보 무결성
- ✓ 비밀성



1. Public Key Cryptography(2) - *Diffie-Hellman(1)*



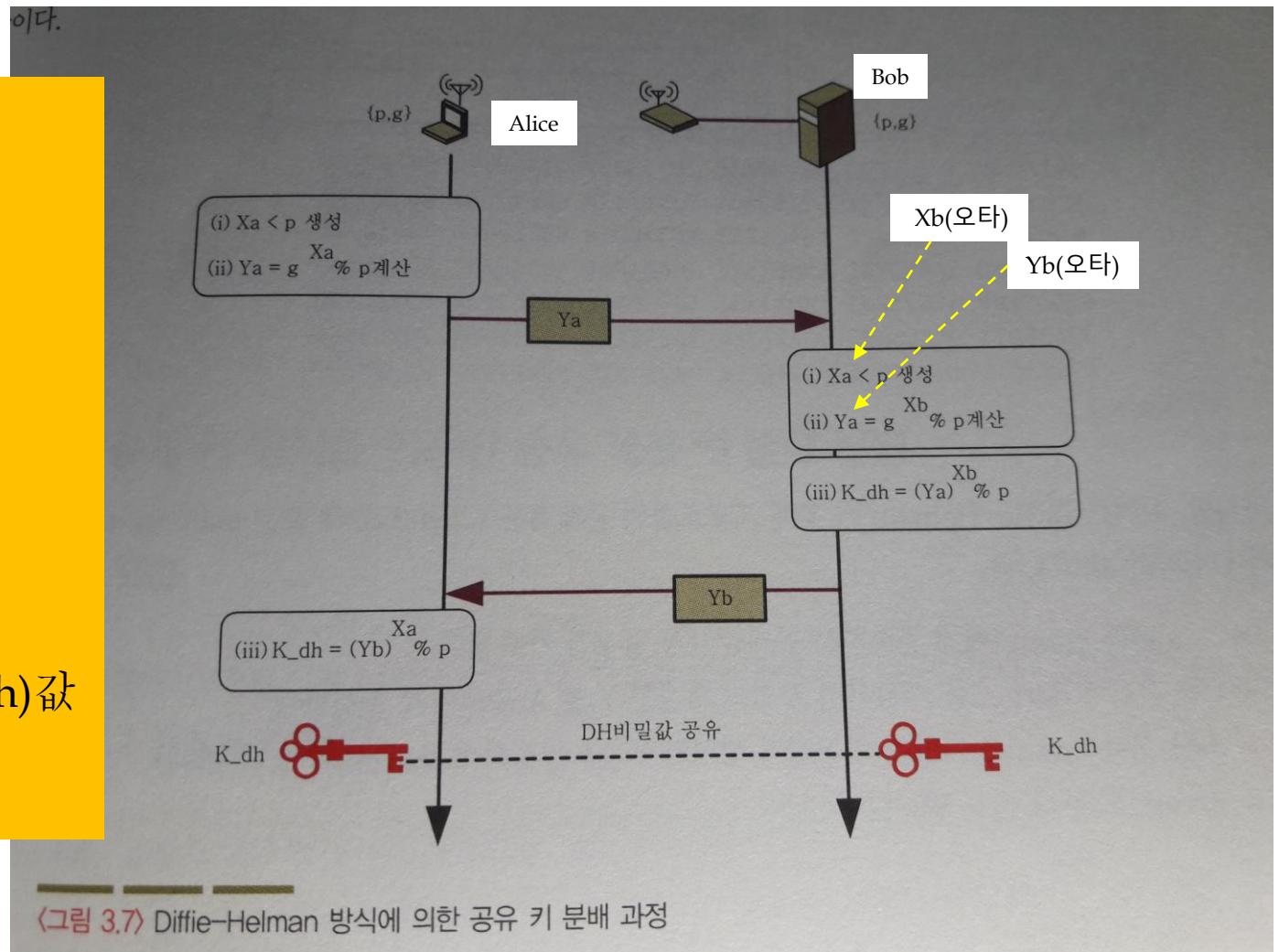
1. Public Key Cryptography(2) - *Diffie-Hellman(2)*

미리 알려진 값: $\{p, g\}$
(p : 소수, g : 원시근(=primitive root of p))

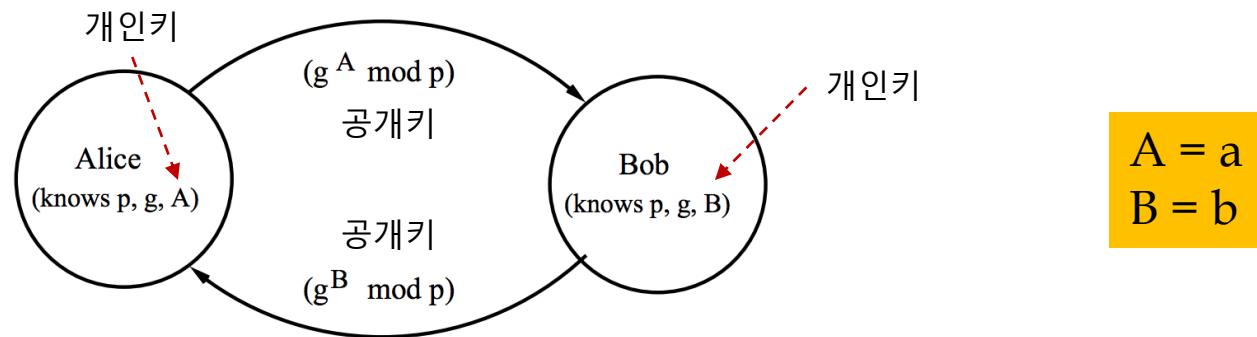
Alice가 생성하는 값
 $X_a < p$: 개인 값(private key)
 $Y_a = g^{X_a} \% p$: 공개 값(public key)

Bob이 생성하는 값
 $X_b < p$: 개인 값
 $Y_b = g^{X_b} \% p$: 공개 값

Alice와 Bob이 공유하는 Session key($= K_{dh}$) 값
Alice: $K_{dh} = (Y_b)^{X_a} \% p$
Bob: $K_{dh} = (Y_a)^{X_b} \% p$



1. Public Key Cryptography(2) - *Diffie-Hellman(3)*



Steps in the algorithm:

- ① Alice and Bob agree on a prime number p and a base g .
- ② Alice chooses a secret number a , and sends Bob $(g^a \text{ mod } p)$.
- ③ Bob chooses a secret number b , and sends Alice $(g^b \text{ mod } p)$.
- ④ Alice computes $((g^b \text{ mod } p)^a \text{ mod } p)$.
- ⑤ Bob computes $((g^a \text{ mod } p)^b \text{ mod } p)$.

Both Alice and Bob can use this number as their key. Notice that p and g need not be protected.

1. Public Key Cryptography(2) - *Diffie-Hellman(4)*

- ➊ Alice and Bob agree on $p = 23$ and $g = 5$.
- ➋ Alice chooses $a = 6$ and sends $5^6 \pmod{23} = 8$.
- ➌ Bob chooses $b = 15$ and sends $5^{15} \pmod{23} = 19$.
- ➍ Alice computes $19^6 \pmod{23} = 2$.
- ➎ Bob computes $8^{15} \pmod{23} = 2$.

Then 2 is the shared secret.

Clearly, much larger values of a , b , and p are required. An eavesdropper cannot discover this value even if she knows p and g and can obtain each of the messages.

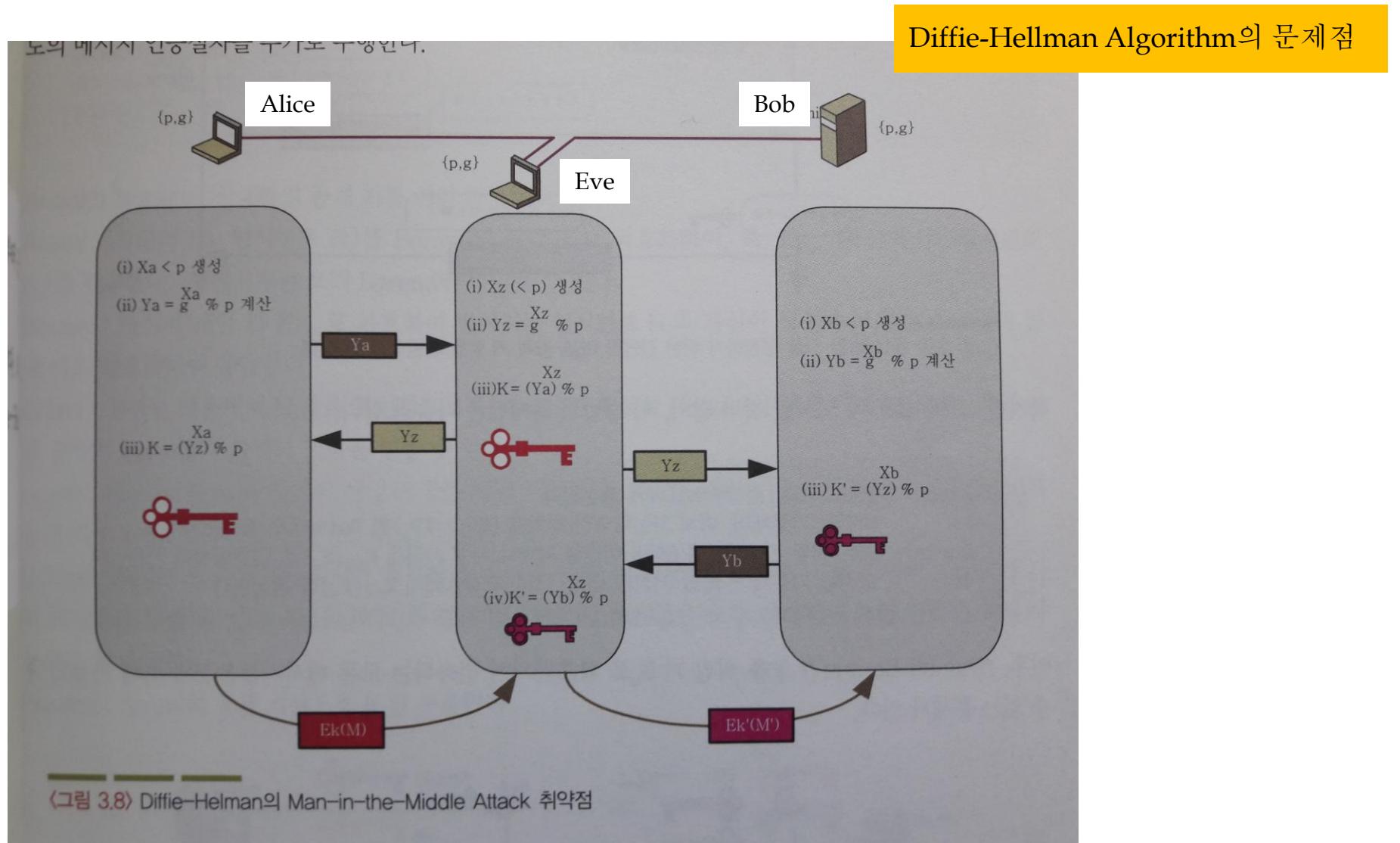
1. Public Key Cryptography(2) - *Diffie-Hellman(5)*

Suppose p is a prime of around 300 digits, and a and b at least 100 digits each.

Discovering the **Private key(a or b)** given g , p , $g^a \bmod p$ and $g^b \bmod p$ would take longer than the lifetime of the universe, using the best known algorithm. This is called the *discrete logarithm problem.*

$$\begin{aligned} Y_a &= g^a \bmod p \\ Y_b &= g^b \bmod p \\ a &= \log_g Y_a \\ b &= \log_g Y_b \end{aligned}$$

1. Public Key Cryptography(2) - Diffie-Hellman(6)



1. Public Key Cryptography(3) - RSA(1)

$$N = p \times q$$

✓ 소인수 분해의 어려움

소인수 분해를 할 수 있는 것
것이 되고 만다. 당장 다음에 제시된 값으로 소인수 분해를 할 수 있겠는가? 이는
이라고 볼 수 있다.

N

748403191420615261237444832310991366573421086805262278994068720043192554
511253327237164425213670384655719426695179058330472822155105711216422613
9504086417

이 값을 소인수 분해하면 다음과 같다. 두 값 중 하나는 p, 하나는 q라고 볼 수 있다.

p

101183725222235794420850217777028583243398012857654035695308875948685532
054091

q

739647793928176799663056465183792004916896176818857281716688613839064966
85587

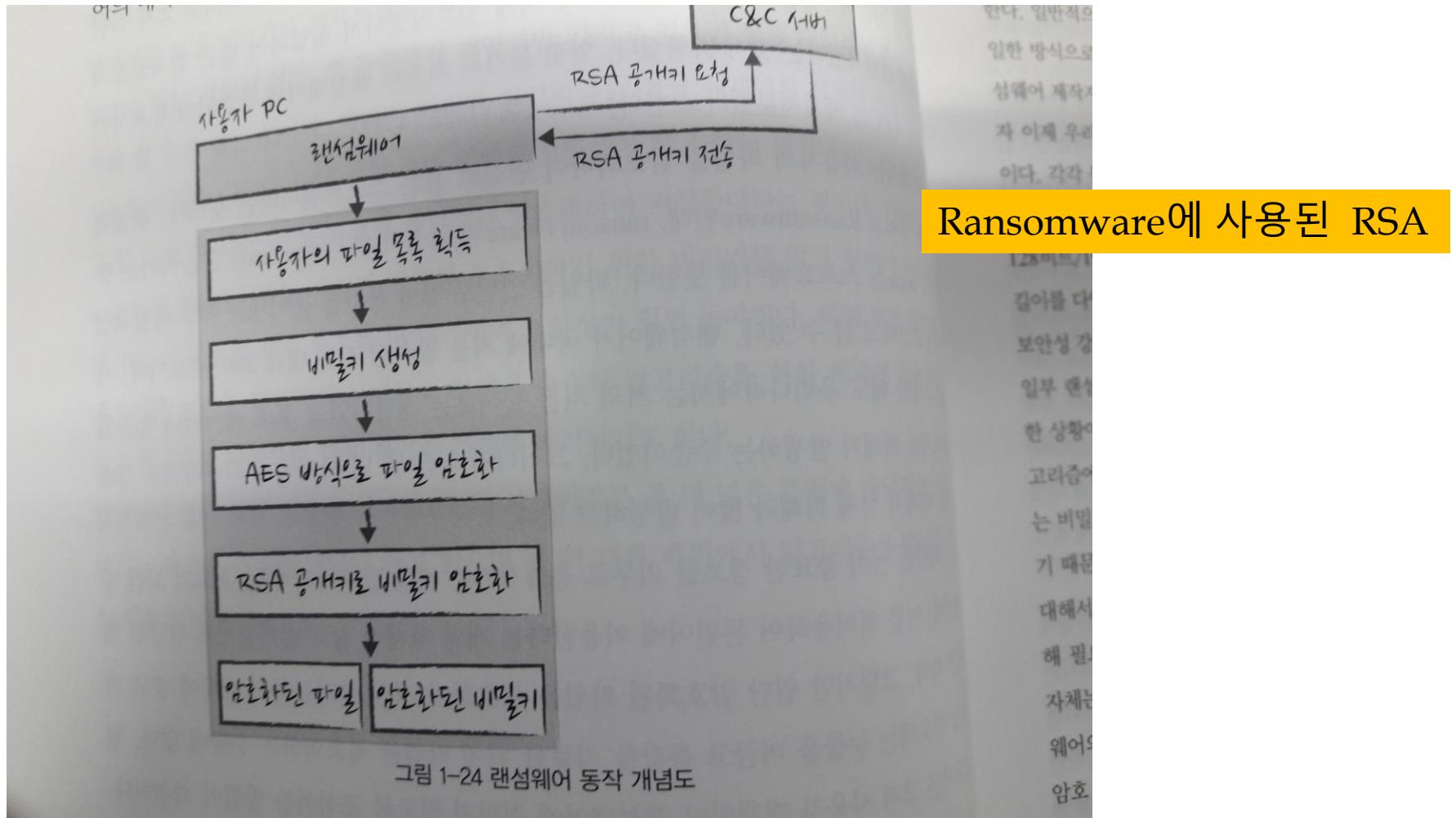
참고로 필자가 직접 소인수 분해를 한 것은 아니다. 두 소수를 찾은 뒤에 곱했기 때문에



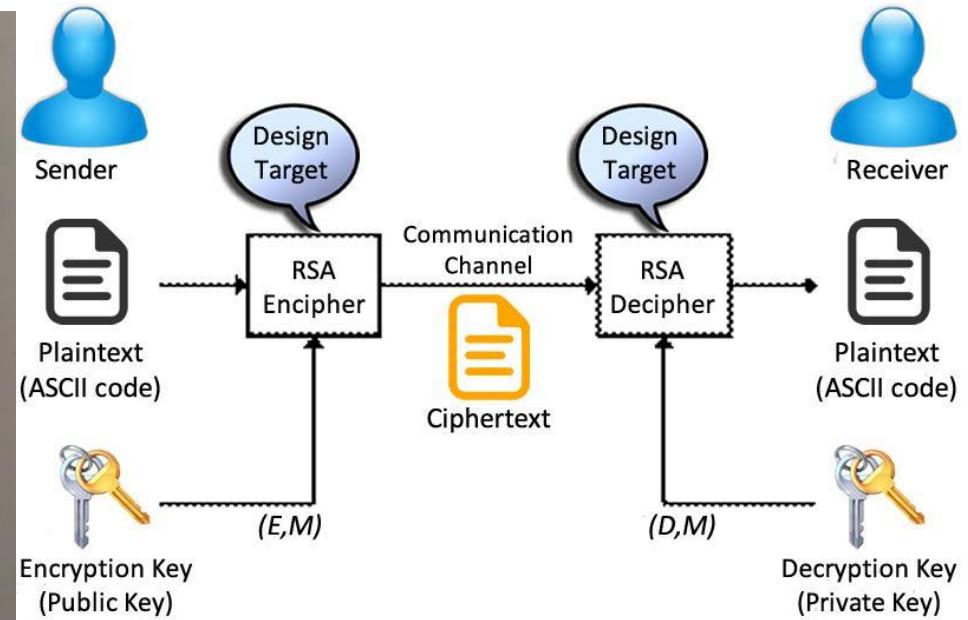
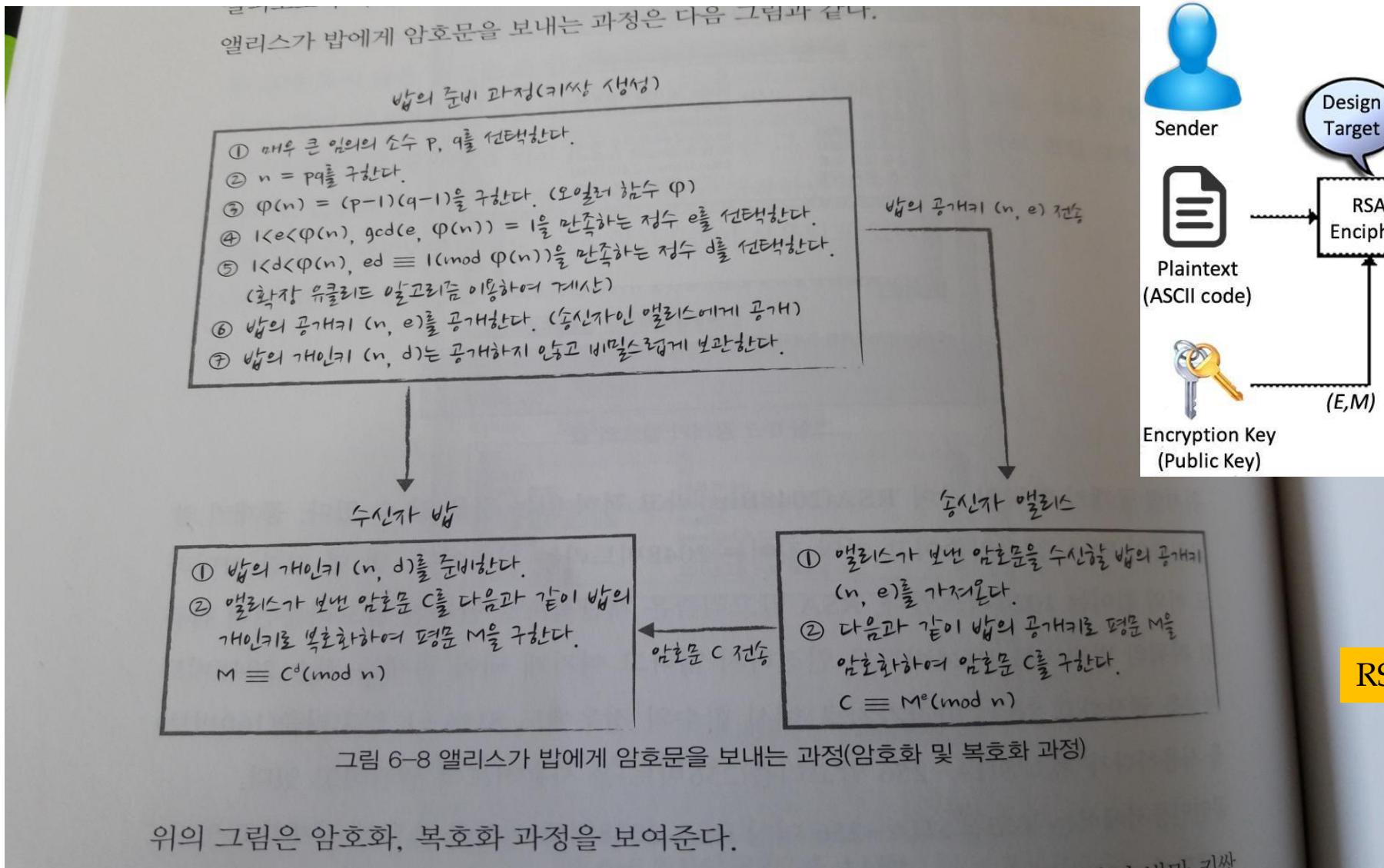
<RSA의 용도>

- 1) 키 교환
- 2) 인증
- 3) 전자 서명
- 4) 암호 통신(X)

1. Public Key Cryptography(3) - RSA(2)



1. Public Key Cryptography(3) - RSA(3)



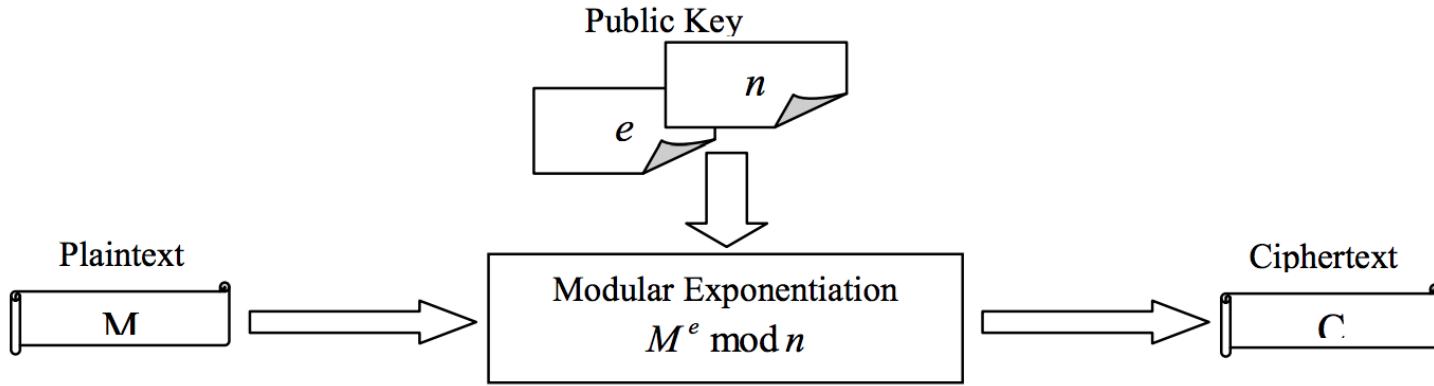
RSA를 사용한 암복호화 과정

위의 그림은 암호화, 복호화 과정을 보여준다.

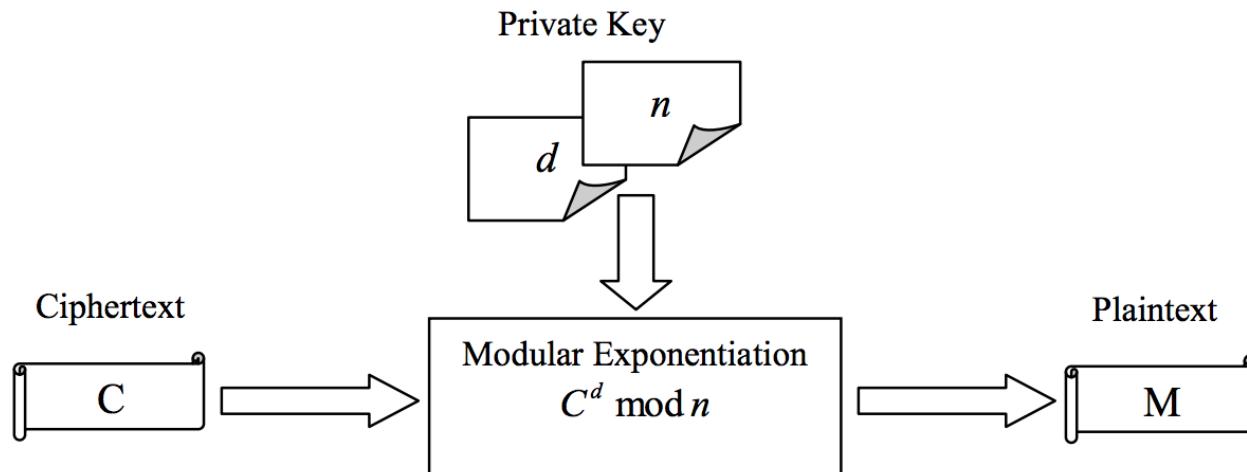
1. Public Key Cryptography(3) - RSA(4)

1. Generate two different primes p and q
2. Calculate the modulus $n = p \times q$
3. Calculate the totient $\phi(n) = (p - 1) \times (q - 1)$
 오일러 함수
4. Select for public exponent an integer e such that $1 < e < \phi(n)$
 and $gcd(\phi(n), e) = 1$
 최대공약수
5. Calculate for the private exponent a value for d such that
$$d = e^{-1} \bmod \phi(n)$$
6. $Public\ Key = [e, n]$
7. $Private\ Key = [d, n]$

1. Public Key Cryptography(3) - *RSA*(5)



(a) RSA Encryption



(b) RSA Decryption

1. Public Key Cryptography(3) - RSA(6)

RSA를 사용한 암복호화 예

공개 키: $KU = \{e, n\}$

개인 키: $KR = \{d, n\}$

(단, e, d, n 은 prime number(소수))

ex)

$KU = \{5, 119\}$

$KR = \{77, 119\}$

$M = 19$ (암화화하려는 data)

암호화된 data

$$C = M^e \% n = 19^5 \% 119 = 2476099 \% 119 = 66$$

복호화된 data

$$M = C^d \% n = 66^{77} \% 119 = 19$$

1. Public Key Cryptography(3) - RSA(7)

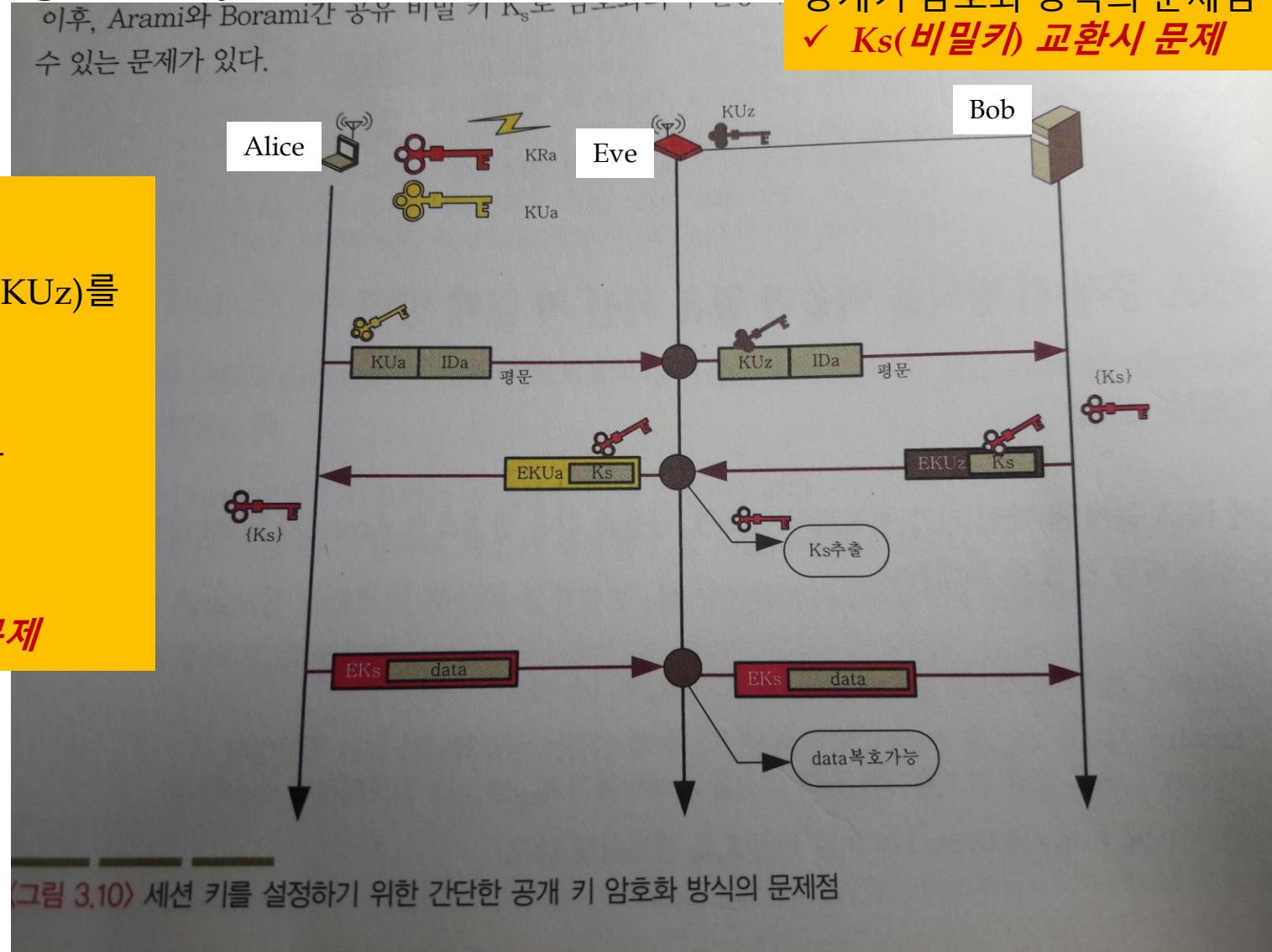
이후, Arami와 Borami간 풍유 비밀 키 K_s 를 모조하는 경우에
수 있는 문제가 있다.

공개키 암호화 방식의 문제점
✓ K_s (비밀키) 교환시 문제

<Man-in-the-Middle Attack>

- 1) A => B에게 공개키(KUa) 전달
 - 2) Z가 중간에서 이를 가로챈 후, 자신의 공개키(KUz)를 B에게 전달
 - 3) B는 비밀 키(K_s)를 생성하여 Z에게 전달
 - 4) Z는 자신의 개인키(KRz)로 K_s 추출(복호화)
 - 5) Z는 A의 공개키로 K_s 를 암호화하여 A에 전달
 - 6) A는 실제 data를 K_s 로 암호화하여 B에 전달
- ✓ 이 내용은 Z가 모두 볼 수 있음.

(* 상대방에 대한 인증 절차가 빠져서 생기는 문제

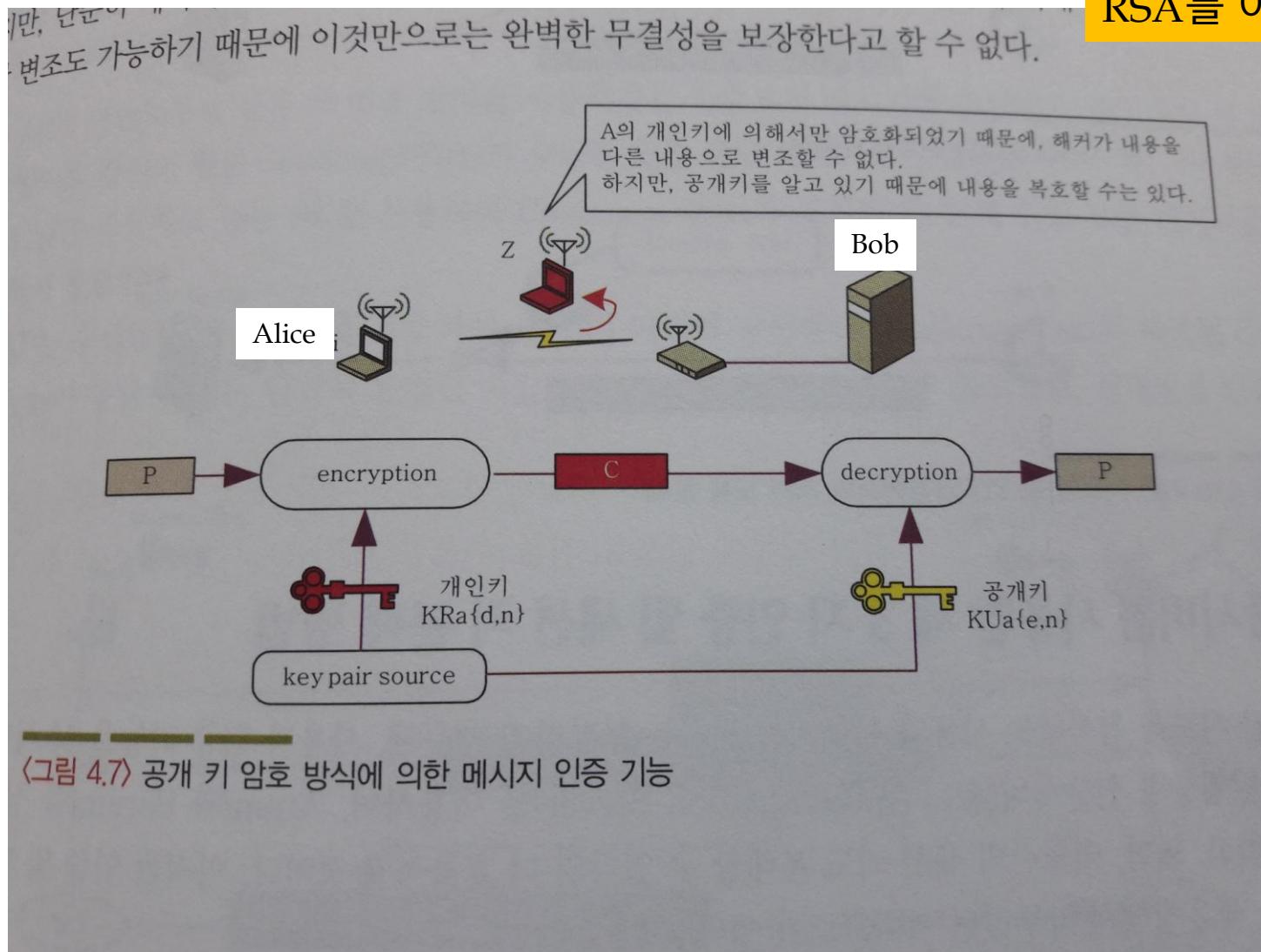


1. Public Key Cryptography(4) - RSA(8)

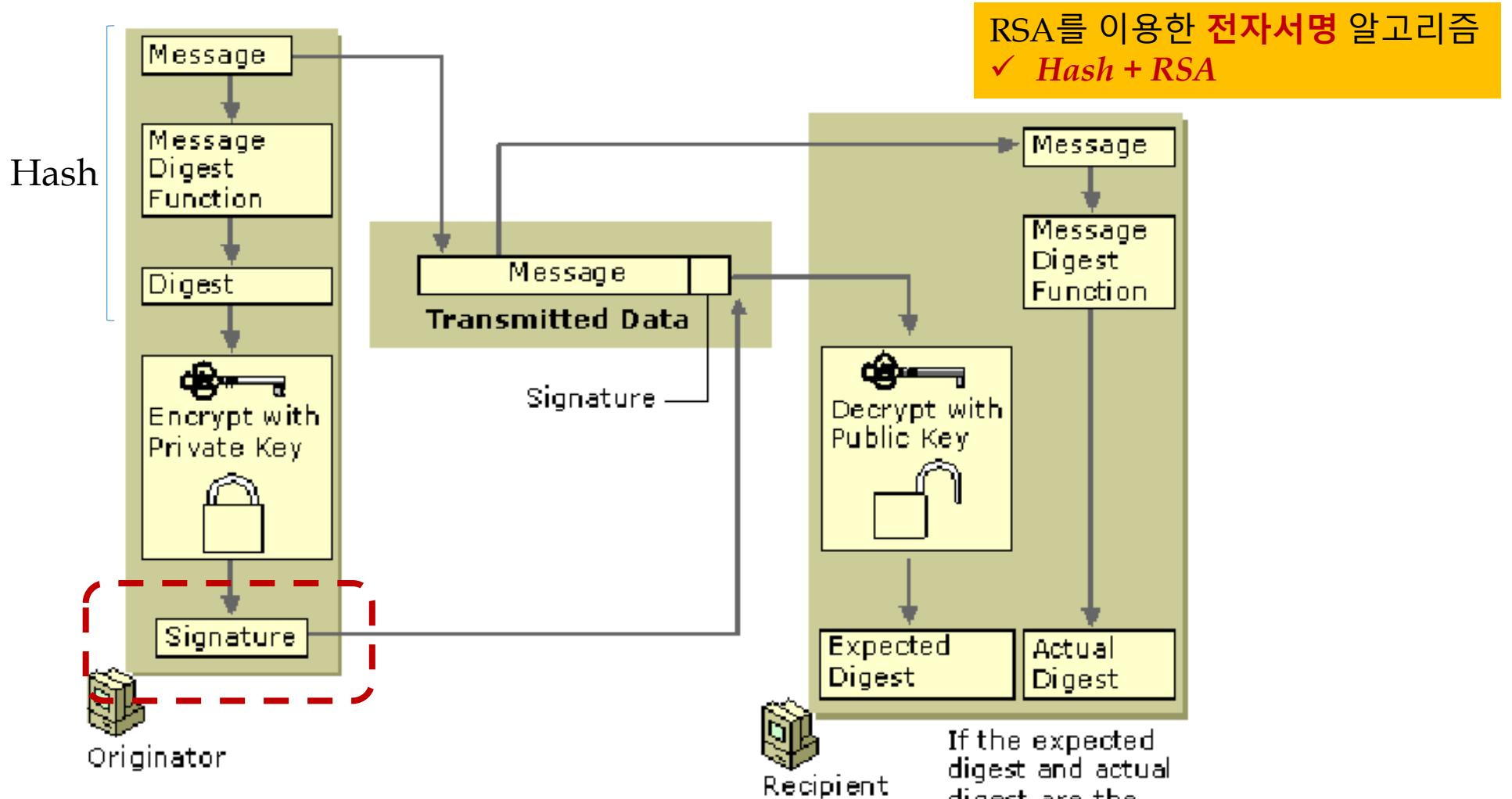
- <인증 - Authentication>
- 1) Message 인증
 - ✓ *Hash*
 - ✓ *MAC(Message Authentication Code), HMAC(Hashed MAC)*
 - ✓ **공개키 방식을 이용한 암호화**
- 2) 사용자/시스템 인증
 - ✓ 커버로스
 - ✓ *Shared key* 인증
 - ✓ 네트워크 접근용 사용자 인증: PAP, CHAP, EAP, RADIUS

1. Public Key Cryptography(4) - RSA(9)

RSA를 이용한 메시지 인증

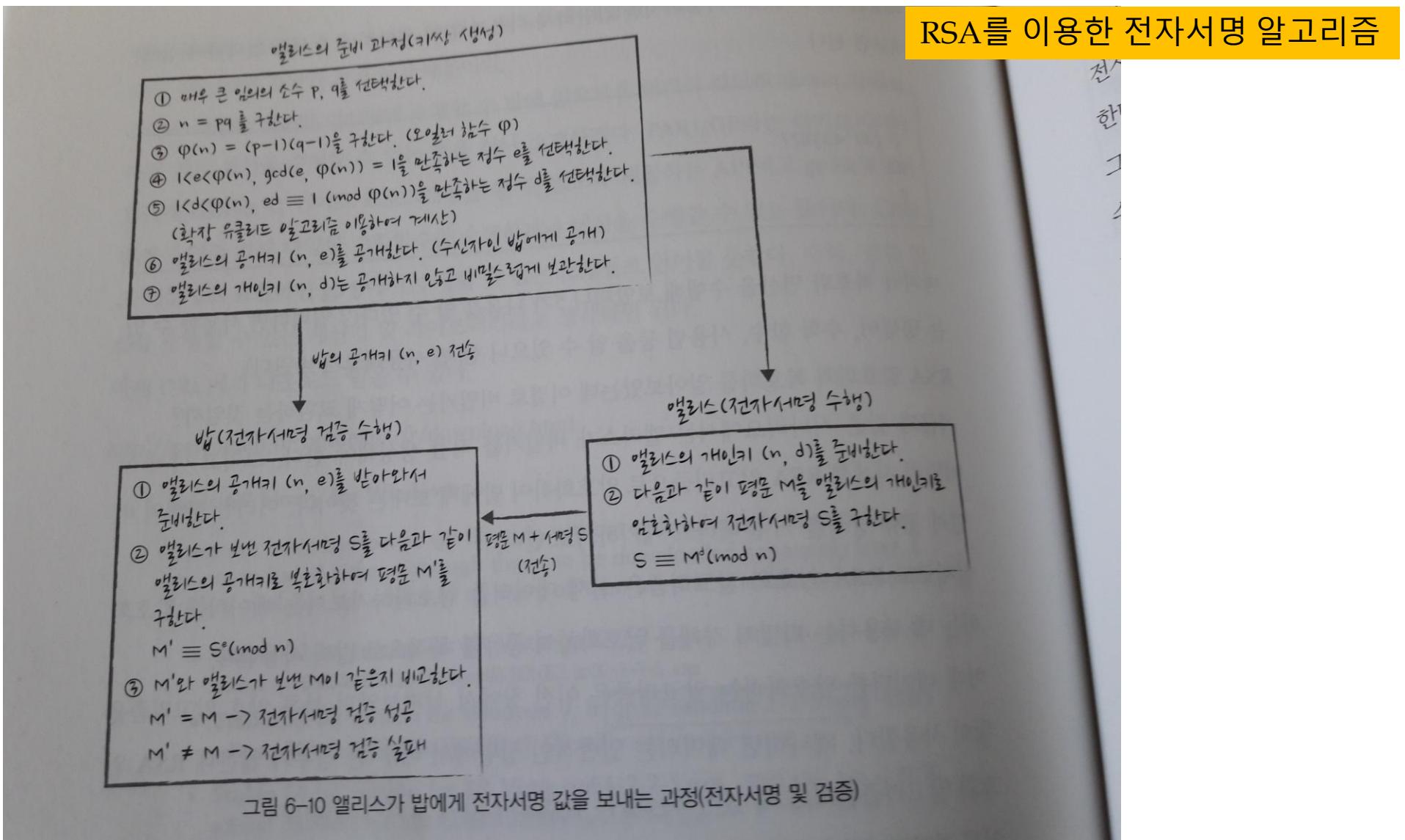


1. Public Key Cryptography(4) - RSA(10)

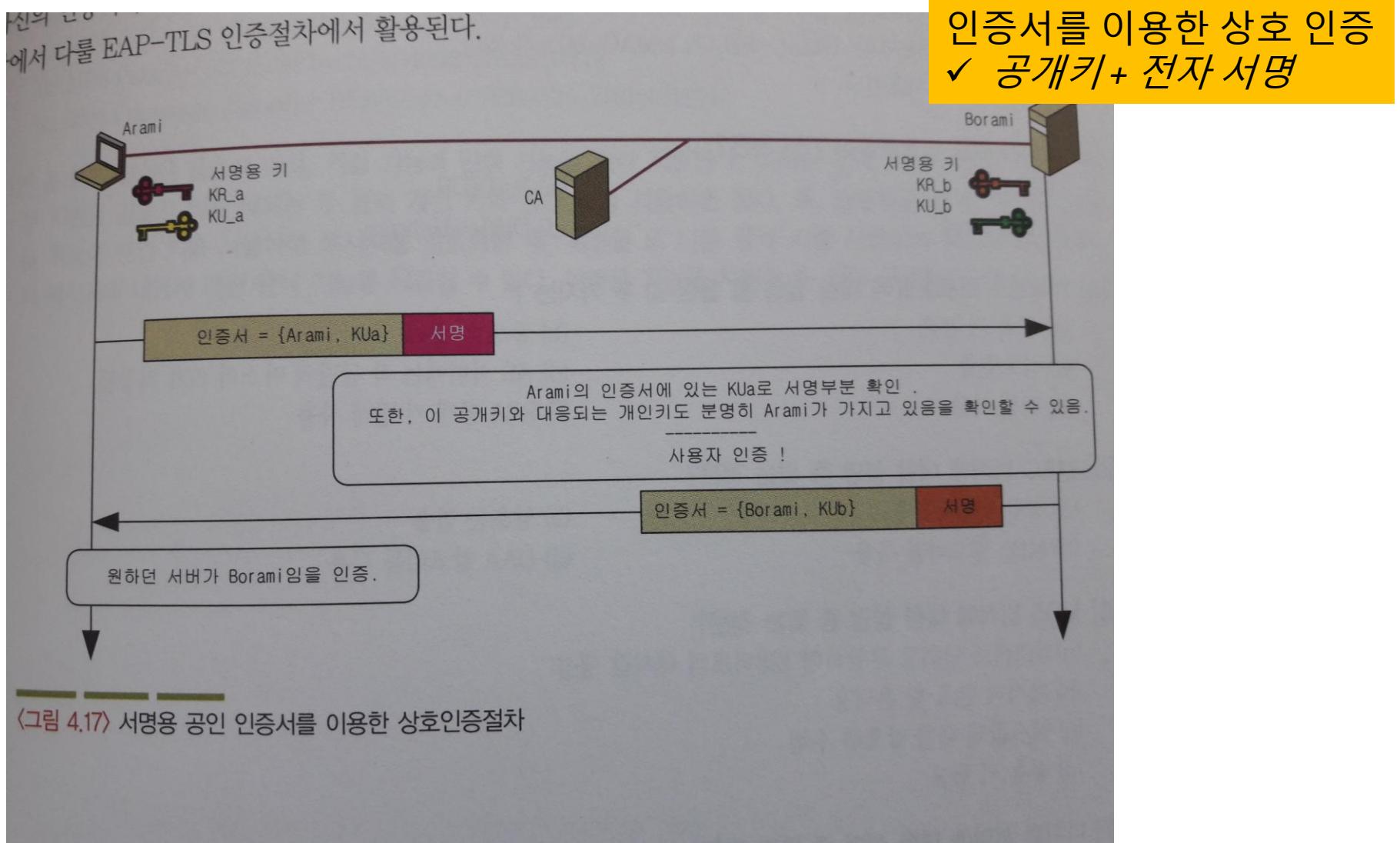


서명 알고리즘으로는 RSA이외에도 DSA도 많이 사용됨.

1. Public Key Cryptography(4) - RSA(11)



1. Public Key Cryptography(4) - RSA(12)



1. Public Key Cryptography(5) - *ECC(1)*

타원곡선(Elliptic Curves) 이론은 약 150년 전부터 수학에서 광범위하게 연구되어 왔으며, 최근 Andrew Wiles의 Fermat's Last Theorem 증명에서 중요하게 사용되었다[1]. 이러한 타원곡선 이론은 1985년 Koblitz와 Miller에 의해 타원곡선 암호시스템(ECC : Elliptic Curve Cryptosystem)이 발표된 이후 지금까지 꾸준한 연구의 대상이 되고 있다. 초기에 Koblitz와 Miller가 타원곡선 암호이론을 발표할 때만 해도 타원곡선군에서의 연산의 구현이 어려워 상용화에 대해 회의적이었지만, 그 후 많은 연구들에 의해 구현이 용이하게 되었고 그 결과로 타원곡선을 이용한 암호는 주목을 받기 시작했다. 이러한 구현의 문제가 해결되고 또한 기존의 RSA/DSA 암호시스템과 비교해서 훨씬 빠르고 효율적이며 짧은 키 길이를 갖는 등 여러 가지 장점을 갖고 있어서 타원곡선 암호시스템에 대한 관심은 더욱 증대되고 있다.

ECC는 암호화 보다는 키교환과 서명용으로 주로 사용되고 있음.

- ✓ RSA의 대체라는 의미보다는 RSA가 적용될 수 없는 특별한 상황에 초점을 맞추는 추세
 - 스마트 카드, 전자 화폐(bitcoin), 무선 통신 기기, 초소형 컴퓨터 등.
- ✓ RSA는 타원곡선 암호보다 암호체계 자체의 이해와 구현이 매우 용이, 안정성 면에서 오랜 기간 검증 받아옴.

1. Public Key Cryptography(5) - ECC(2)

타원곡선 상의 두점의 합을 찾는 문제

$$P = (x_1, y_1)$$

$$Q = (x_2, y_2)$$

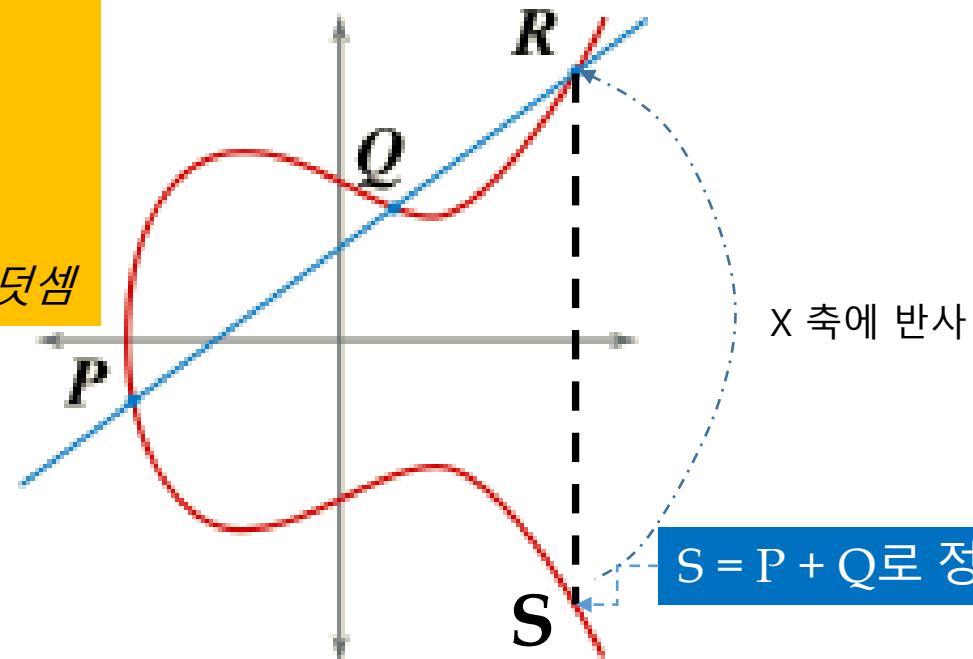
$$S = (x_3, y_3)$$

$$S = P + Q$$

✓ 타원 곡선 상에서의 덧셈 연산(정의)

✓ 타원 곡선에서 필요한 유일한 수학은 덧셈

타원 곡선 방정식



$$E : y^2 = x^3 + ax + b$$

1. Public Key Cryptography(5) - ECC(3)

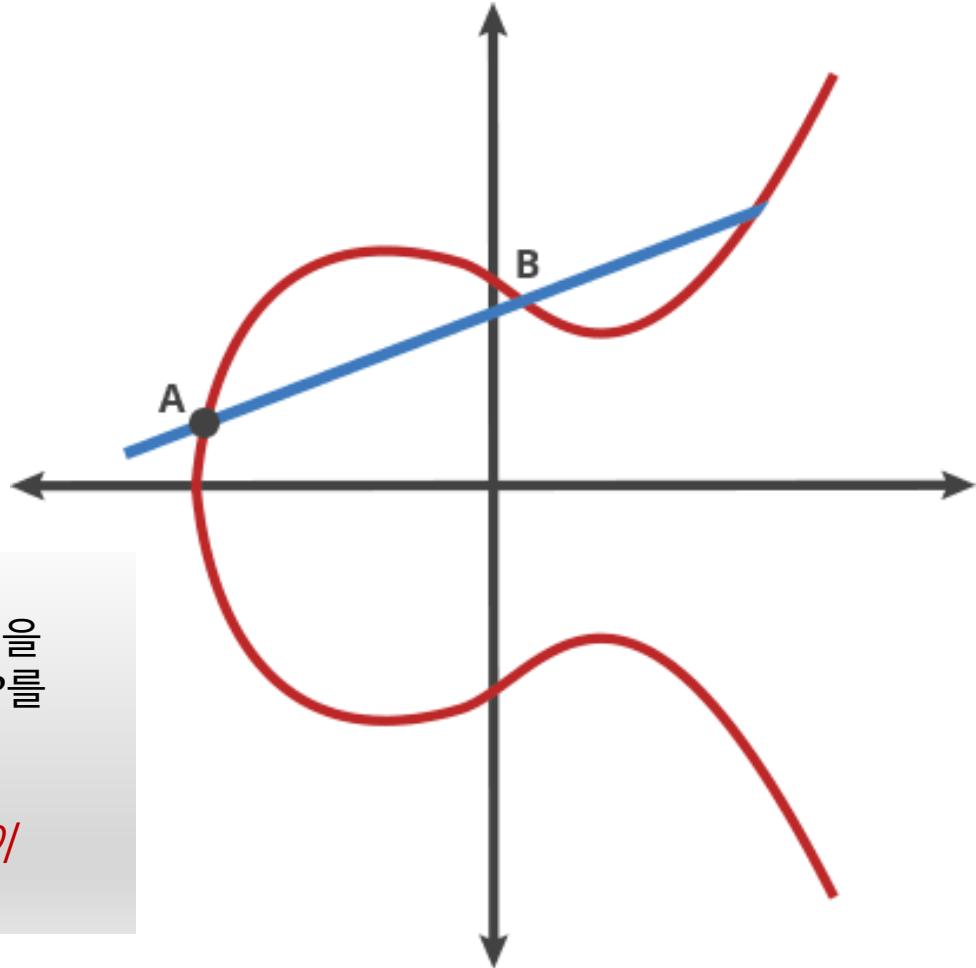
<https://steemit.com/kr/@icoreport/key-2-ecc>

$$\begin{aligned} Q &= nP \\ n &= \log_P Q ? \end{aligned}$$

<타원 곡선 암호 알고리즘의 key point>

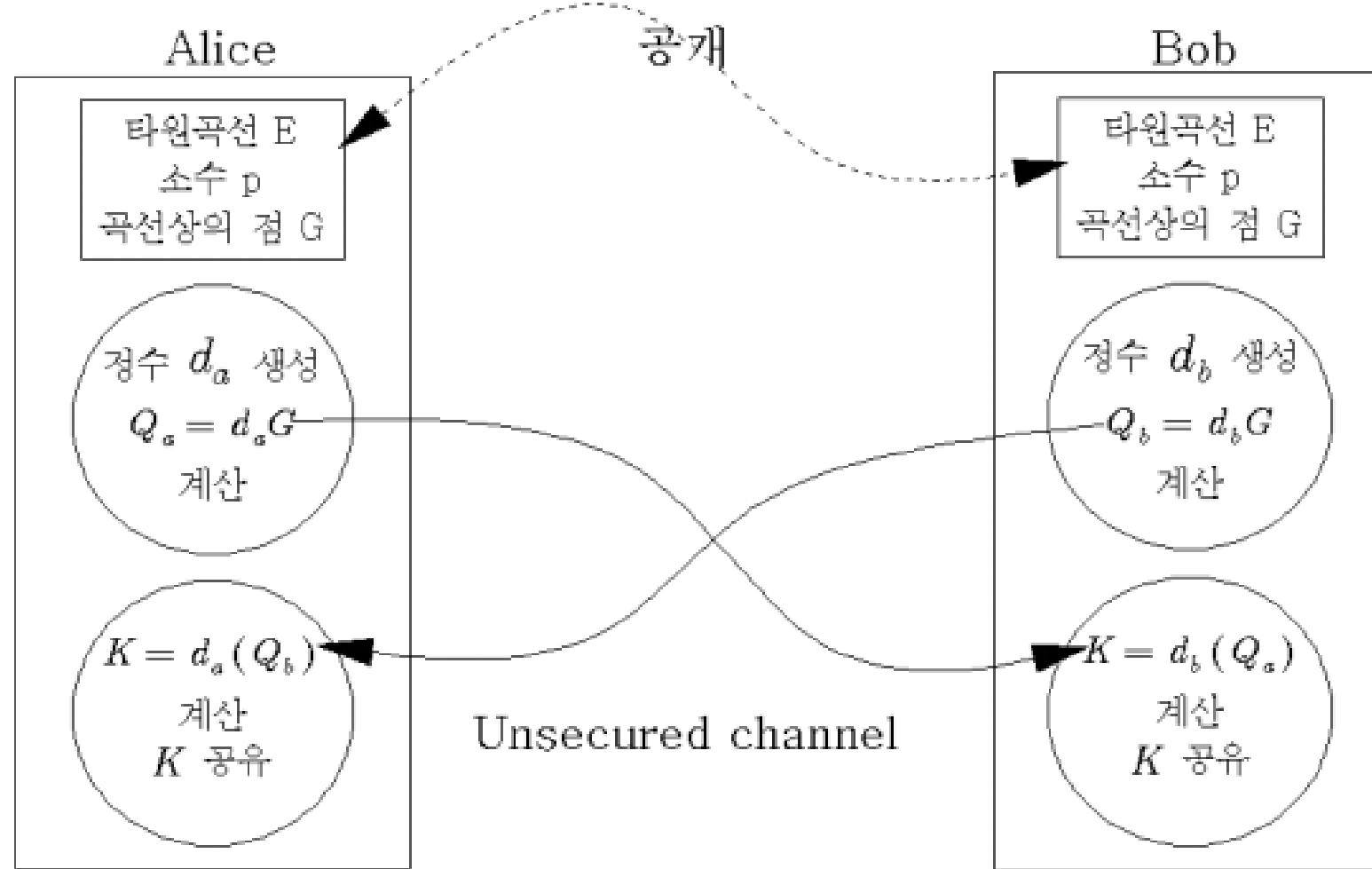
유한체 F_q 위에 정의된 타원곡선 E 와 그 위의 점 P 가 있을 때, 기점을 P 로 갖는 E 의 이산대수문제란, E 위의 점 Q 가 주어졌을 때 $Q = nP$ 를 만족하는 n 을 찾는 문제이다.

✓ (쉽게 말하면) 타원 곡선 상의 임의의 두 점 P 와 Q 가 존재하고, 적당한 자연수 n 이 주어졌을 때, n 과 P 를 알면 Q 를 구하는 것이 쉽지만, P 와 Q 만을 아는 상태에서 n 을 구하는 것은 쉽지 않다.



1. Public Key Cryptography(5) - ECC(4)

ECDH: Elliptic Curve + Diffie-Hellman



1. Public Key Cryptography(5) - ECC(5)

는 것이 쉽지 않은데 이를 타원 곡선 이산대수 문제라고 한다.

타원 곡선 디피-헬먼 키 교환 알고리즘은 아래와 같다.

ECDH: Diffie-Hellman + Elliptic Curve

ECDH는 DH(Diffie-Hellman) 알고리즘을 타원곡선군 위에 그대로 구현한 것으로 유한체 위의 **DH 알고리즘에 비해 키 크기가 작고 여러 가지 공격에 강함.**

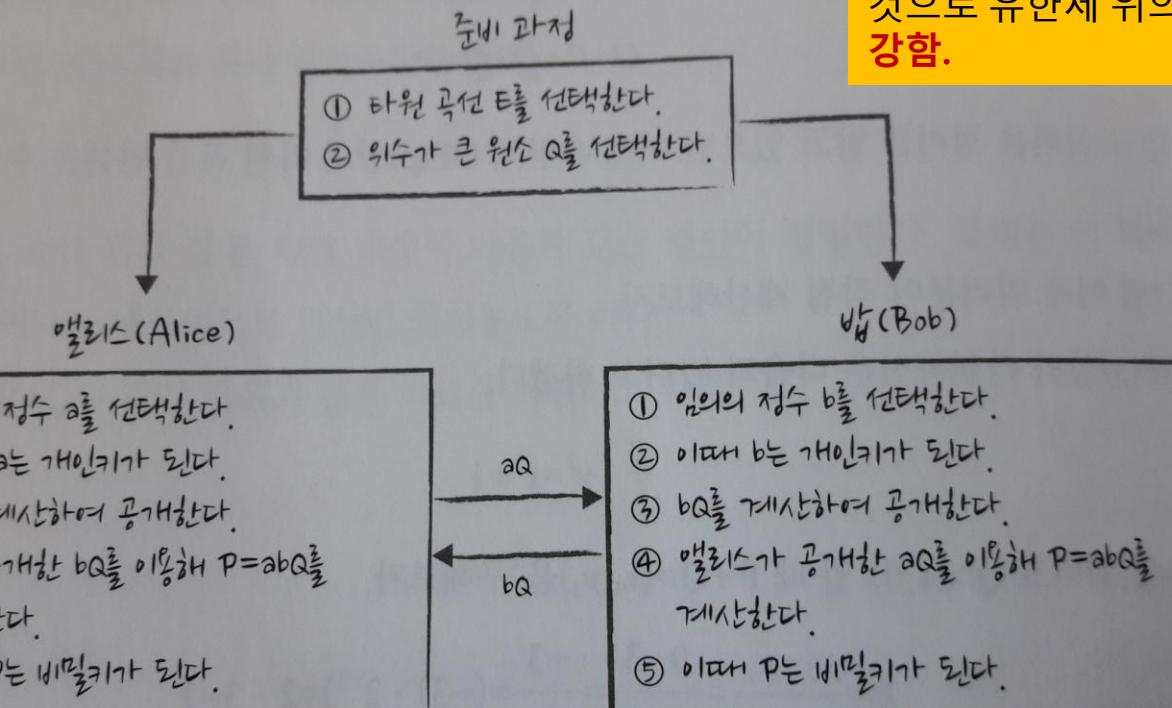
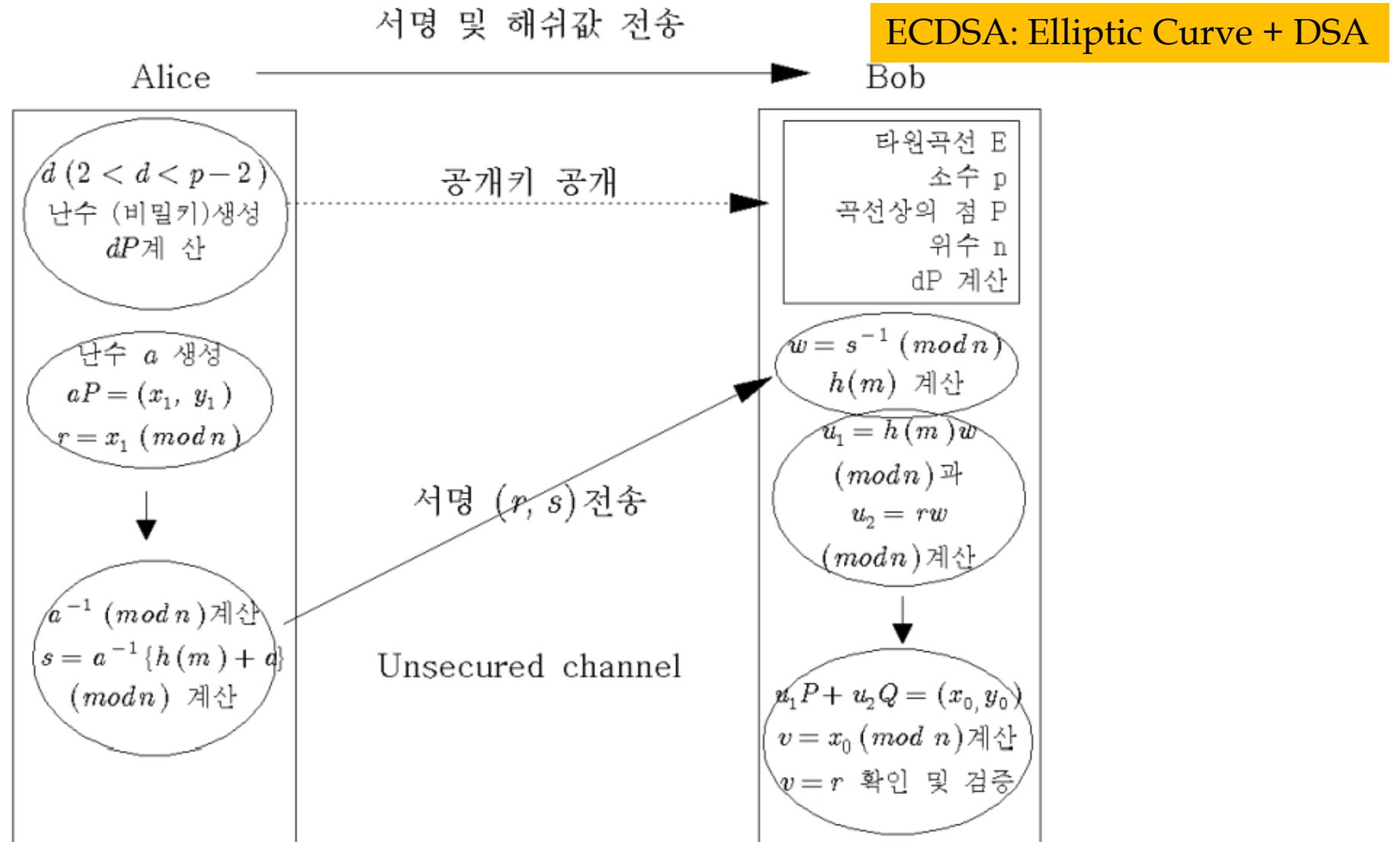


그림 6-18 타원 곡선 디피-헬먼 키 교환 알고리즘 수행 과정

그림을 살펴보면 타원 곡선 기본에서 살펴본 연산들이 사용되는 것을 알 수 있다.

1. Public Key Cryptography(5) - ECC(6)



1. Public Key Cryptography(5) - *ECC(7)*

<http://repository.kmou.ac.kr/bitstream/2014.oak/10390/1/000002174041.pdf>

(RSA : 공개키 n , DSA : p , 타원곡선 : 정의체)

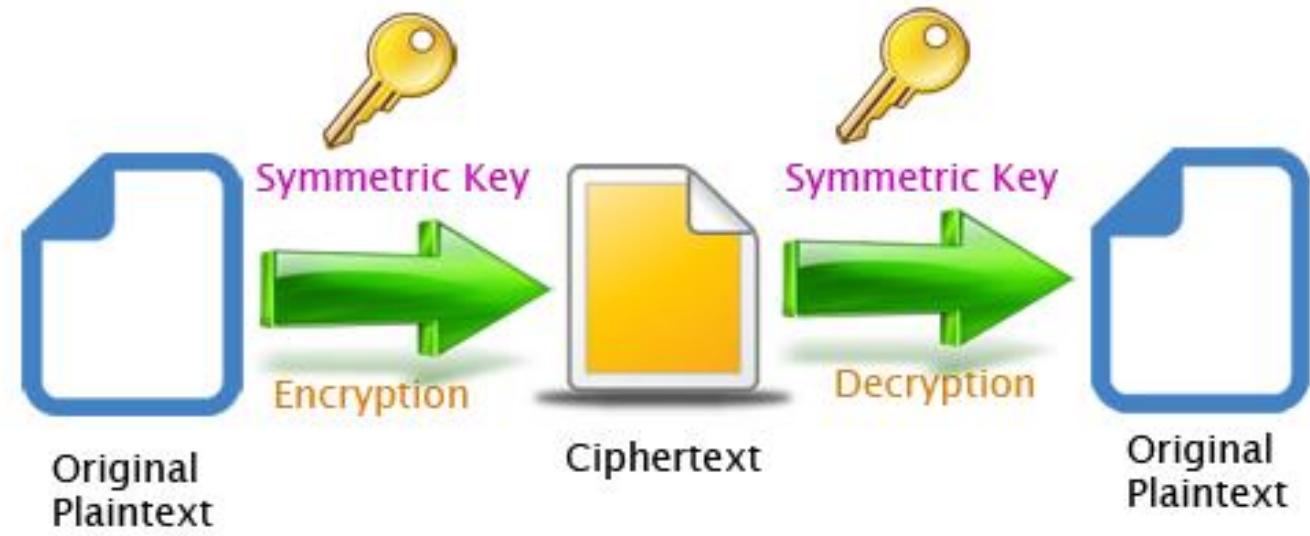
Time to break in MIPS year	RSA/DSA (bits)	ECC (bits)	RSA vs. ECC (key size ratio)
10^4	512	106	5:1
10^8	768	132	6:1
10^{12}	1024	160	7:1
10^{20}	2048	210	10:1
10^{36}	5120	320	17:1
10^{168}	120000	1200	100:1

RSA, DSA가 1024 bit key가 필요한 상황에서 ECC는 160bit 면 충분함.

2. Symmetric Cipher Algorithm(1)

- Block Cipher, Stream Cipher
- 3DES, AES, ARIA, Blowfish .. RC4..
- CBC, ECB, PCBC, CFB, OFB, CTR mode ..
- 파이스텔 network, SPN
(round 함수 적용 방법)
- ...

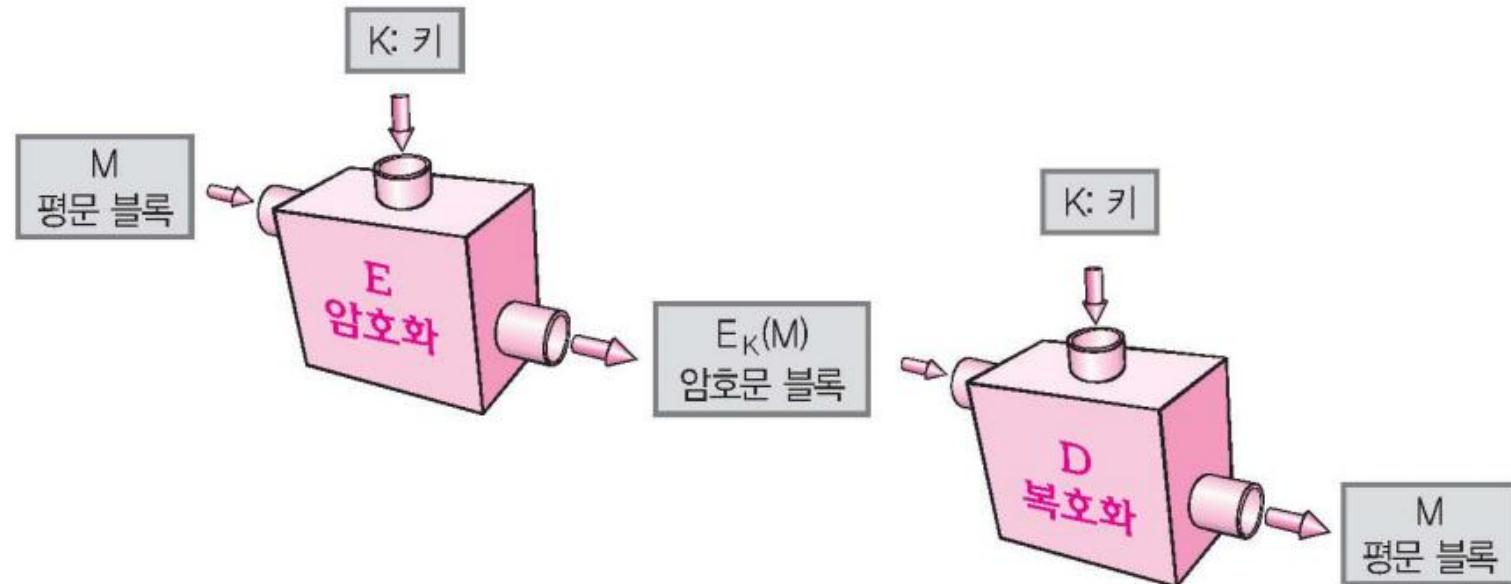
대칭키 암호 알고리즘에서는 비밀키를 어떻게 안전하게 교환하는가가
아주 중요한 문제이다.



Plaintext + key를 함께 암호화

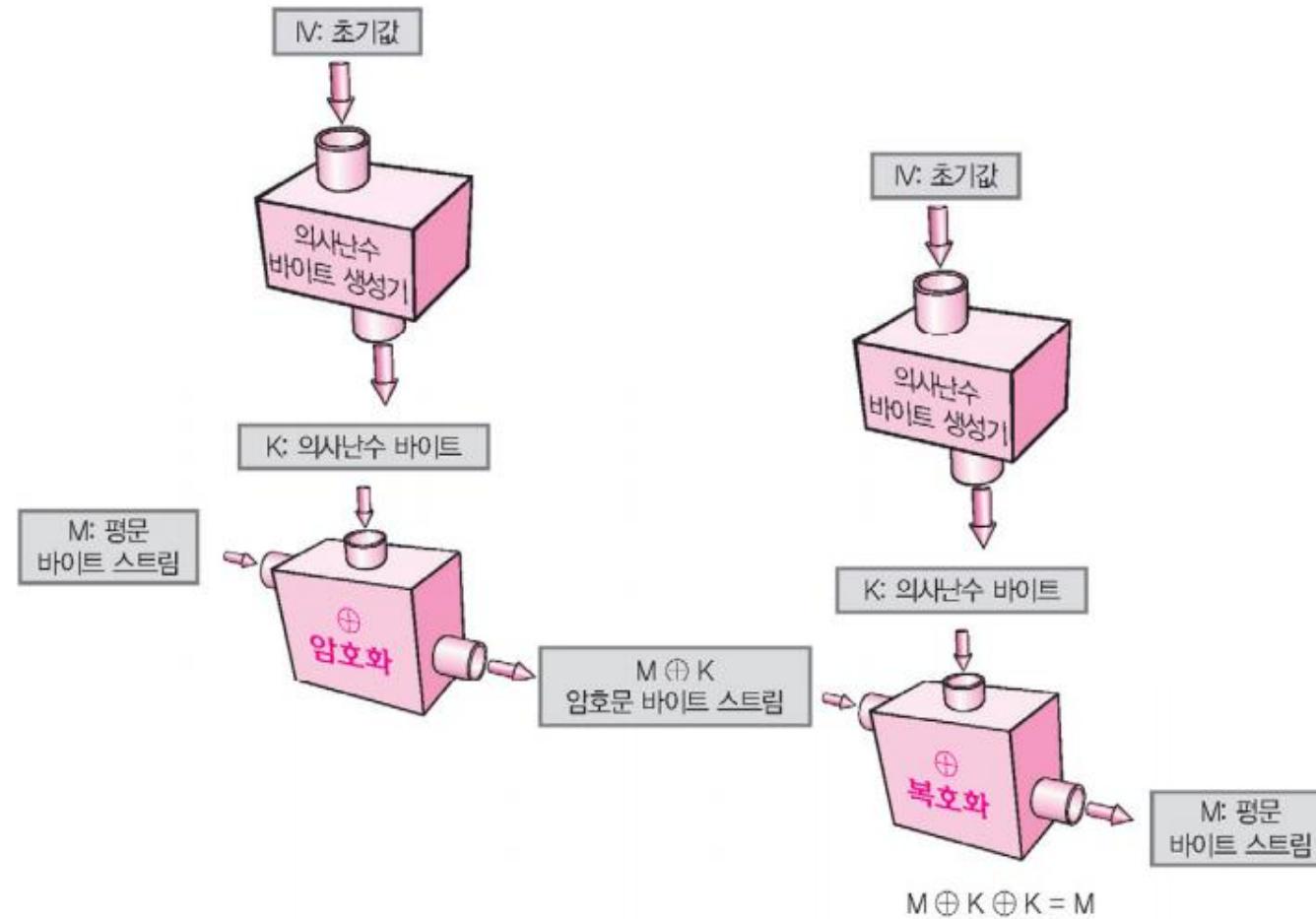
Ciphertext + key를 함께 복호화

2. Symmetric Cipher Algorithm(2)



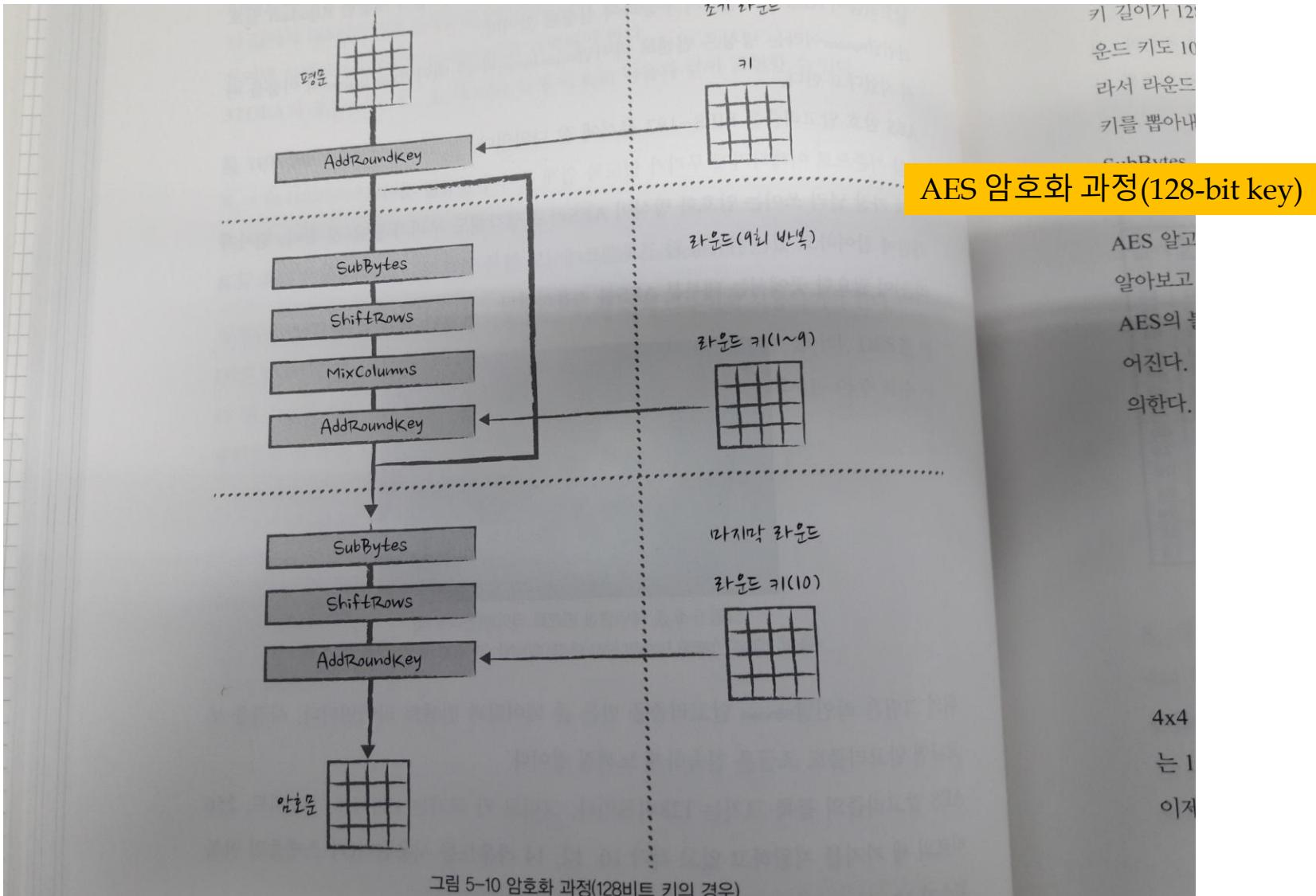
Block Cipher 동작 원리

2. Symmetric Cipher Algorithm(3)



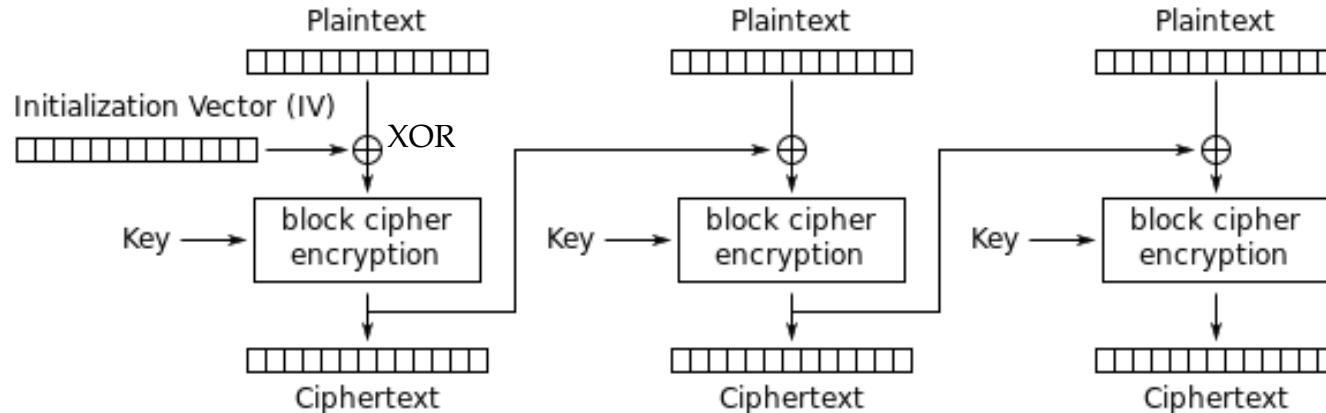
Stream Cipher 동작 원리

2. Symmetric Cipher Algorithm(4)



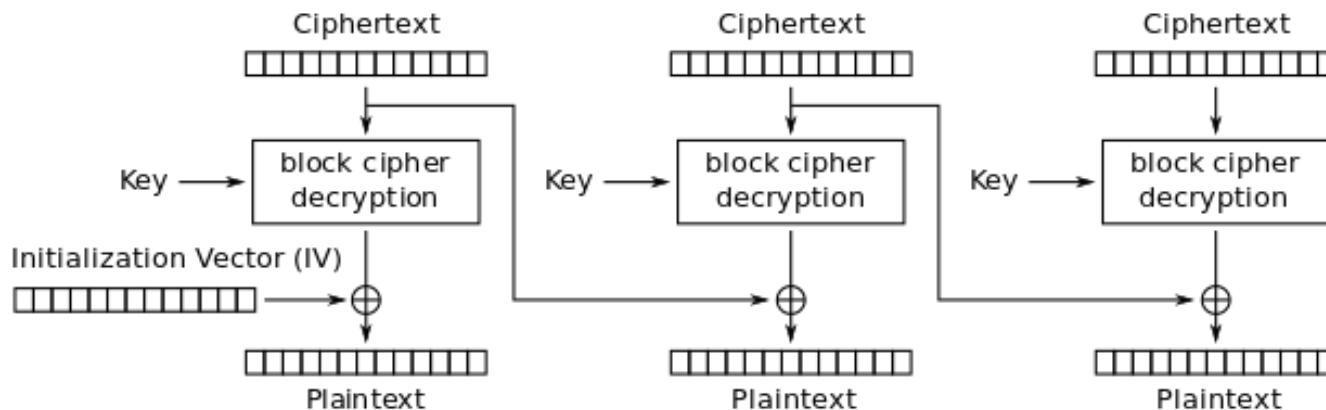
2. Symmetric Cipher Algorithm(5)

Block Cipher CBC mode 동작 원리



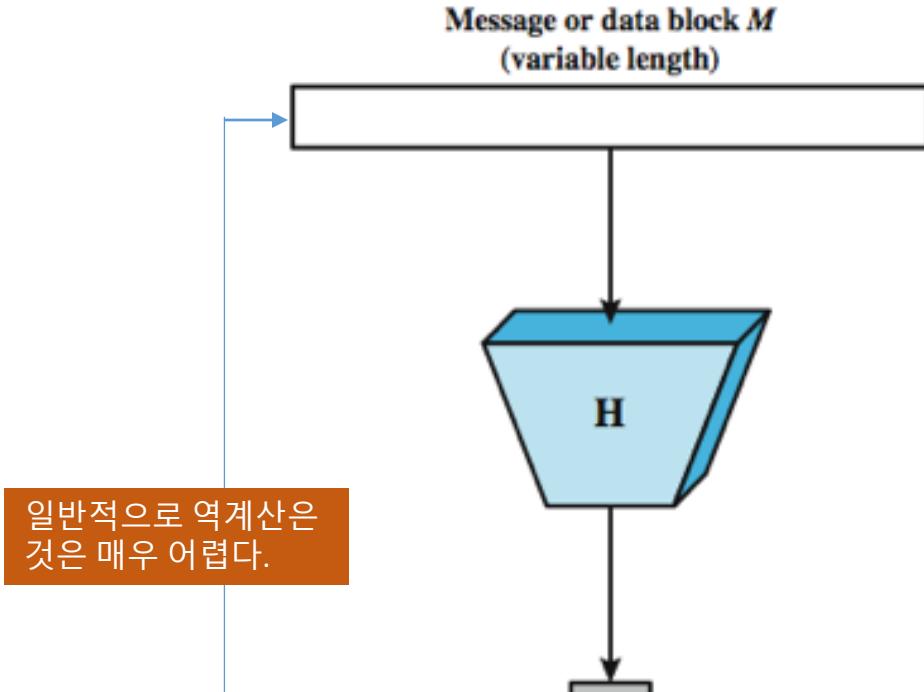
AES-128-CBC
AES-256-CBC

Cipher Block Chaining (CBC) mode encryption

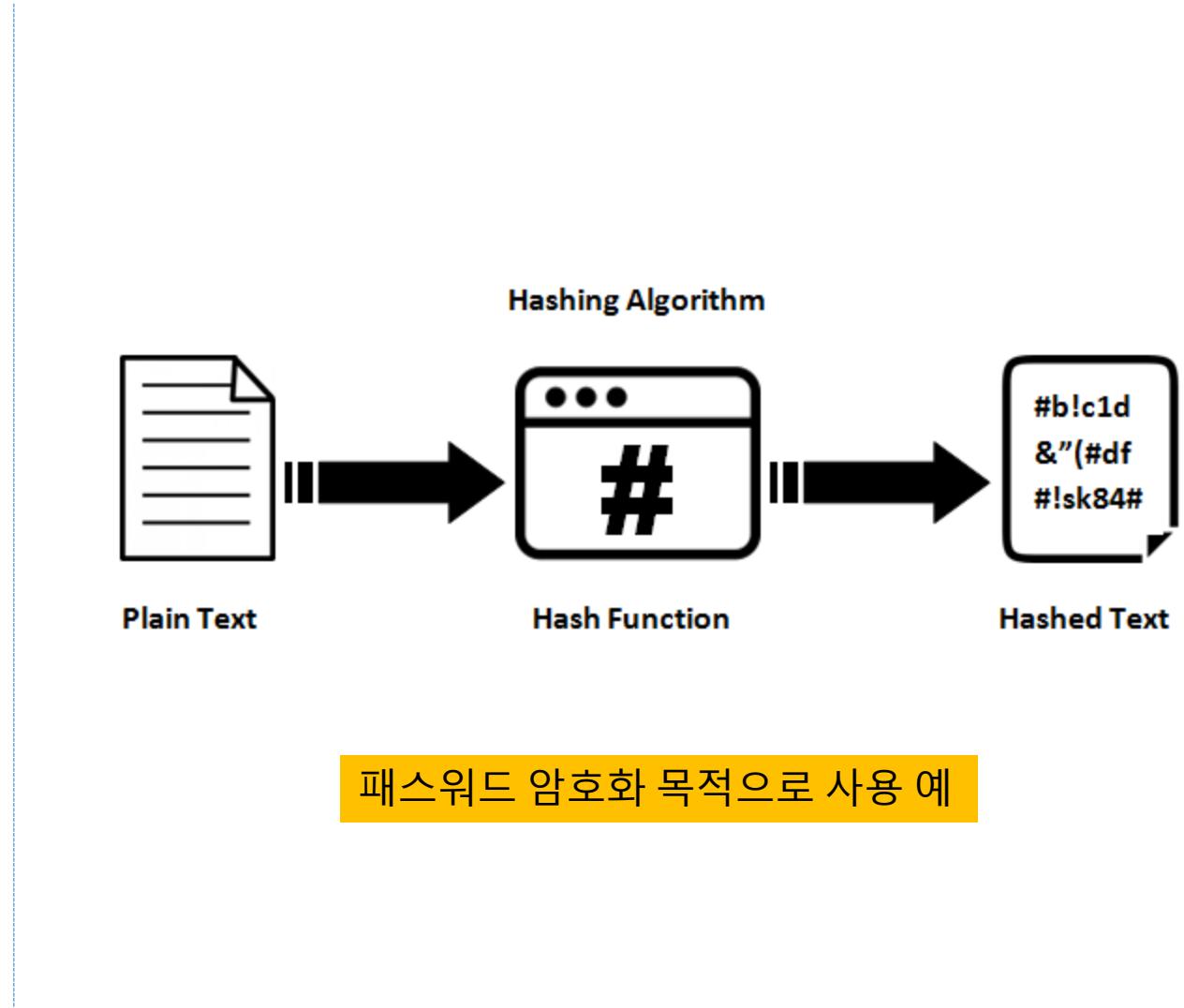


Cipher Block Chaining (CBC) mode decryption

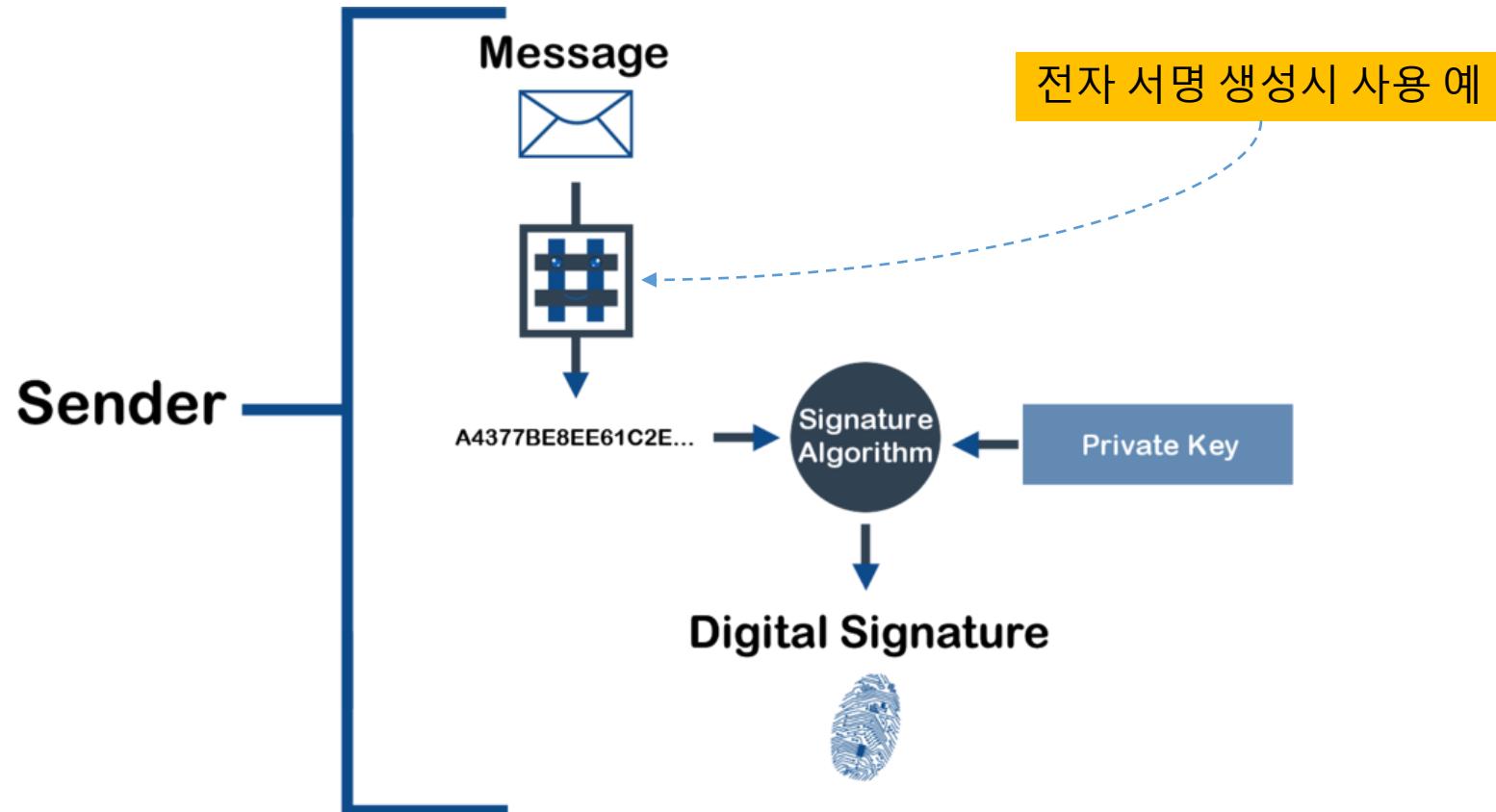
3. Hash Algorithm(1) - *Overview*



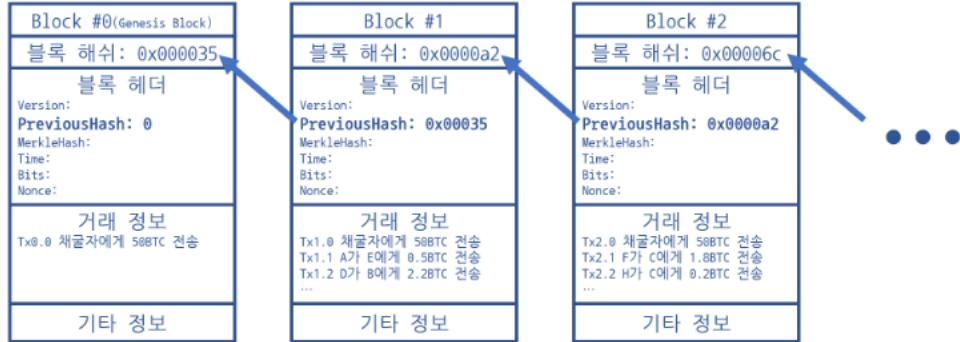
데이터 무결성 검증 용



3. Hash Algorithm(2) - 전자 서명



3. Hash Algorithm(3) - *Block Chain*



Blockchain의 Block Hash(block header에 대한 hash)를 구하는 과정(SHA-256이 사용됨)

- ✓ Blockchain은 Hash와 전자서명을 기초로 함.
- ✓ Nonce 값을 구하는 채굴(mining) 작업에는 엄청난 횟수의 hash 계산이 필요함.

3. Hash Algorithm(4) - 히시 교표

알고리즘	출력 비트 수	내부 상태 크기 ^[c 1]	블록 크기	Length size	Word size
GOST	256	256	256	256	32
HAVAL	256/224/192/160/128	256	1,024	64	32
MD2	128	384	128	-	32
MD4	128	128	512	64	32
MD5	128	128	512	64	32
PANAMA	256	8,736	256	-	32
RadioGatún	Up to 608/1,216 (19 words)	58 words	3 words	-	1~64
RIPEMD	128	128	512	64	32
RIPEMD-128/256	128/256	128/256	512	64	32
RIPEMD-160	160	160	512	64	32
RIPEMD-320	320	320	512	64	32
SHA-0	160	160	512	64	32
SHA-1	160	160	512	64	32
SHA-224/224	256/224	256	512	64	32
SHA-512/384	512/384	512	1,024	128	64
Tiger(2)-192/160/128	192/160/128	192	512	64	64
WHIRLPOOL	512	512	512	256	8

한다.
표 1-1에서는 대표적인 해시 함수와 해시 값의 길이를 볼 수 있다.

해시 함수	해시 값 길이(비트)	보안 강도	권고 여부
MD5	128	80비트 미만	권고하지 않음
SHA-1	160	80비트 미만	권고하지 않음
HAS-160	160	80비트	권고하지 않음
SHA-2	SHA-224	224	112비트
	SHA-256	256	128비트
	SHA-384	384	192비트
	SHA-512	512	256비트

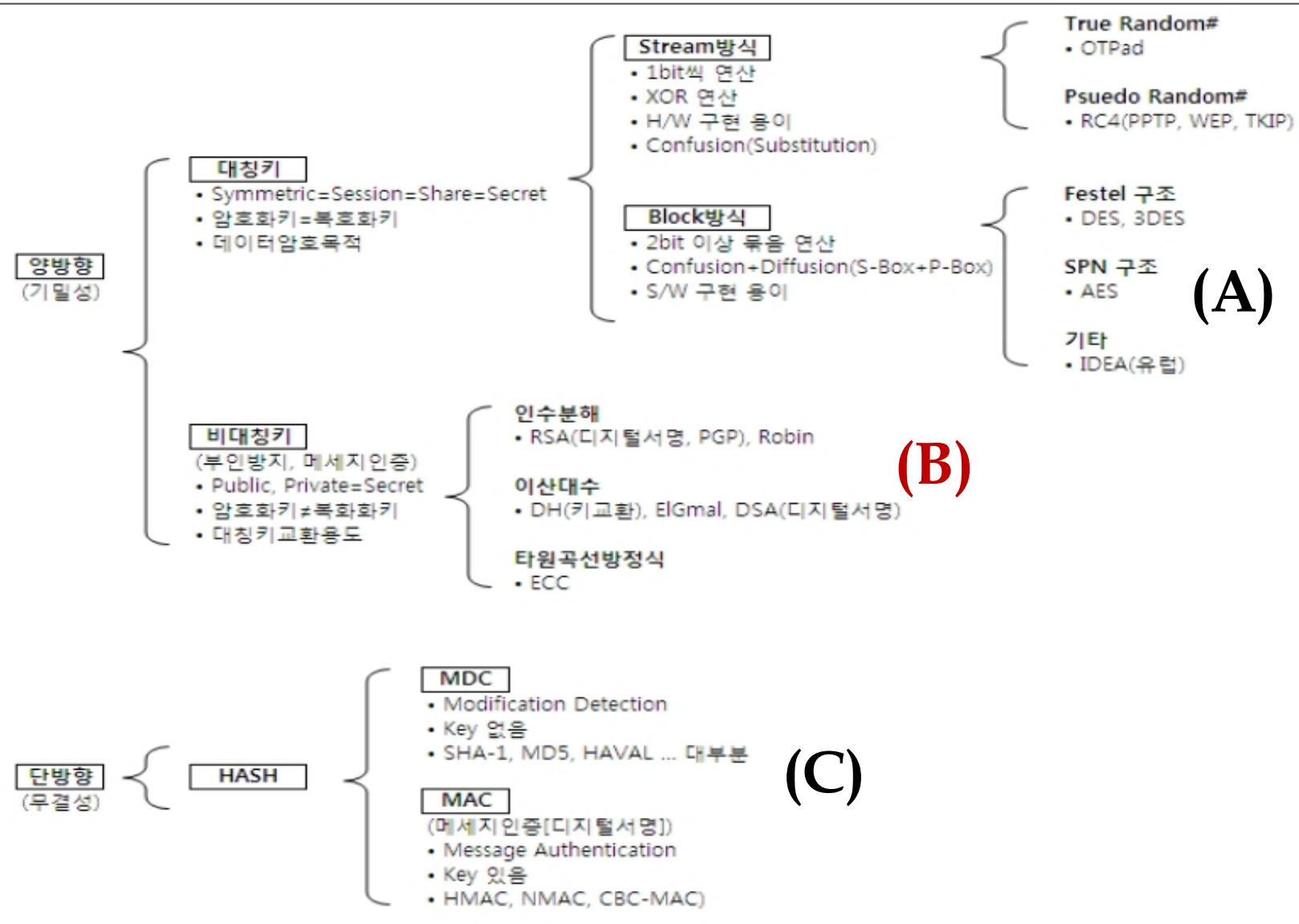
표 1-1 해시 함수 종류와 보안 강도

1. ↑ Merkle-Damgård 방식에서 각 라운드 시 생성되는 내부 해시의 크기.

2. ↑ 라운드 수가 표시되지 않은 경우 전체 라운드에 대한 공격을 의미한다.

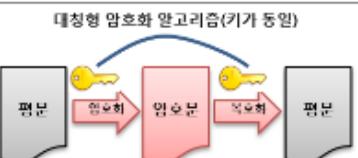
Appendix

A. Cryptography 분류(1)



A. Cryptography 분류(2)

암호화 기술	주요 내용	적용 알고리즘
SPN	<ul style="list-style-type: none"> - Substitution-Permutation Network(대치-치환망 구조) - 전치와 치환을 이용하여 관용 암호방식의 문제 해결 - 128비트를 4x4 행렬로 나타내어 행렬을 이용한 암호화 	AES ARIA
Feistel	<ul style="list-style-type: none"> - N비트의 블록을 N/2씩 둘로 나누고, R번의 라운드만큼 반복적으로 적용, 이전 블록 암호문과 평문을 Exclusive-OR 한 형태 	DES SEED
인수분해	<ul style="list-style-type: none"> - 두 큰 소수 p와 q의 곱셈은 쉬우나 n으로부터 p와 q를 추출하기 어려운 점 이용 	RSA
타원곡선	<ul style="list-style-type: none"> - PKI 기반인 RSA의 문제점인 속도와 안정성을 해결하기 위해서 타원기반의 안정성과 효율성을 기반으로 생성된 알고리즘 	ECC
이산대수	<ul style="list-style-type: none"> - 이산대수의 계산은 어렵지만 그 역함수/지수함수의 계산은 빠르게 수행하는 특징을 이용 	Diffie-Hellman DSA
해쉬 알고리즘	<ul style="list-style-type: none"> - 임의의 길이를 가지고 있는 메시지를 받아들여 고정된 길이의 출력값으로 바꾸어주는 알고리즘 - 원래의 입력값을 찾아내는 것은 불가능 	MD-5 SHA-1 SHA-2

구분	대칭키	비대칭키
개념도	 <p>대칭형 암호화 알고리즘(키가 동일)</p>	 <p>비 대칭형 암호화 알고리즘(키가 다른 키)</p>
대표적 알고리즘	DES, SEED, AES	RSA, ECC
키의 관계	암호키 = 복호키	암호키 ≠ 복호키
키의 수	두 사람이 한 개의 비밀키를 공유	전송 당사자간에 각각 키쌍(Private Key, Public Key)를 공유
키의 종류	암호화키 : Secret Key 복호화키 : Secret Key	암호화키 : Public Key 복호화키 : Private Key
구현 방식	블록, 스트림암호화	소인수분해, 이산대수, 근저백터
키의 관리	모든 전송 당사자간 암/복호화를 위한 키를 공유해야 함	인증기관을 통해 전송 당사자 별 Private Key 발급
부인방지여부	대칭키로 인하여 부인방지 불가능	키의 이원화로 부인방지 가능
속도	비트 단위 암호화로 상대적으로 빠른 속도 제공	큰 소수를 찾거나, 곡률 방정식 등의 연산으로 속도가 느림
용도	개인파일암호화, 특정 불특정그룹 내의 통신에 사용	다수의 정보교환(Key)에 주로 사용
장점	구현 용이, 변형 가능	암호해독이 어려움, 전자서명
단점	상대적으로 쉽게 해독가능하며, 키관리가 어려움	해독시간이 상대적으로 오래 걸림

A. Cryptography 분류(3)

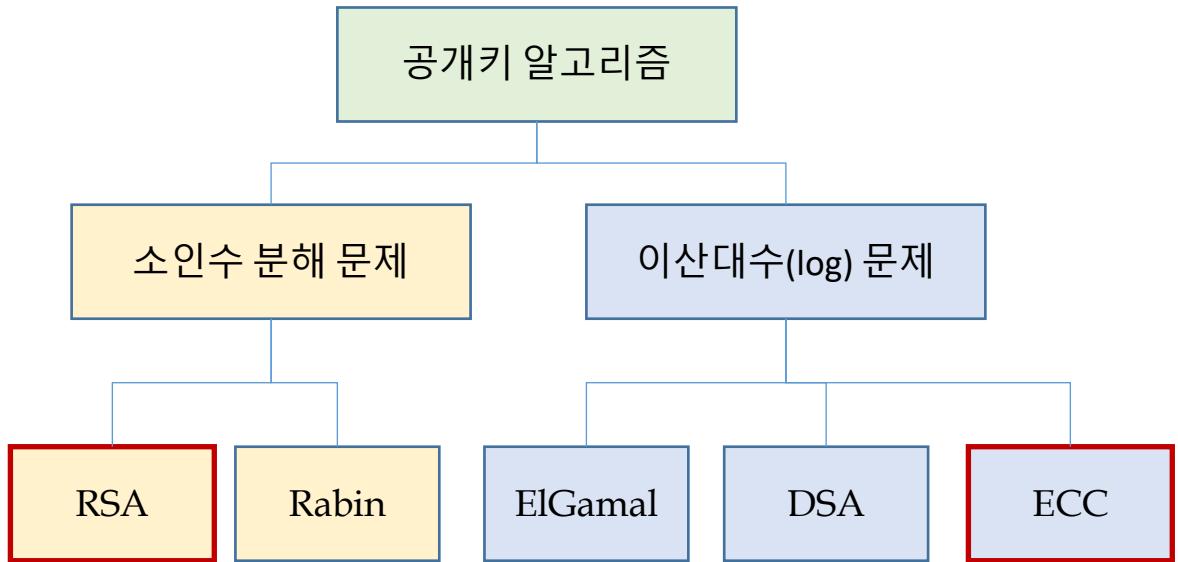
가. 비밀키 암호화 알고리즘 특성 비교

구분	DES	3DES	AES	ARIA	SEED	HEIGHT
키 크기	56	168	128/192/256	128/192/256	128/256	128
평문블록크기	64	64	128	128	128	64
암호문 블록크기	64	64	128	128	128	64
라운드수	16	48	10/12/14	12/14/16	16/24	32
전체구조	Feistel	Feistel	SPN	ISPN	Feistel	변형-Feistel
개발기관	미국 표준 기술 연구소			국가보안기술연구소	정보보호진흥원	고려대

나. 보안강도에 따른 분류

보안강도	NIST(미국)	국내	안전성유지기간
80비트 이상	AES-128/192/256 2TDEA 3TDEA	SEED ARIA-128/192/256	2010년까지
112비트 이상	AES-128/192/256 3TDEA	SEED ARIA-128/192/256	2011년부터 2030년까지 (최대 20년)
128비트 이상	AES-128/192/256	SEED ARIA-128/192/256	2030년 이후 (최대 30년)
192비트 이상	AES-192/256	ARIA-192/256	
256비트 이상	AES-256	ARIA-256	

A. Cryptography 분류(4)



<소인수분해 문제 기반 암호 알고리즘>

$A = B * C$ (단, B와 C는 소수이면서 소인수)
A를 알고 있는 상태에서 B와 C 찾기 ?

<이산대수 문제 기반 암호 알고리즘>

$$A = B^C$$

$$C = \log_B A ?$$

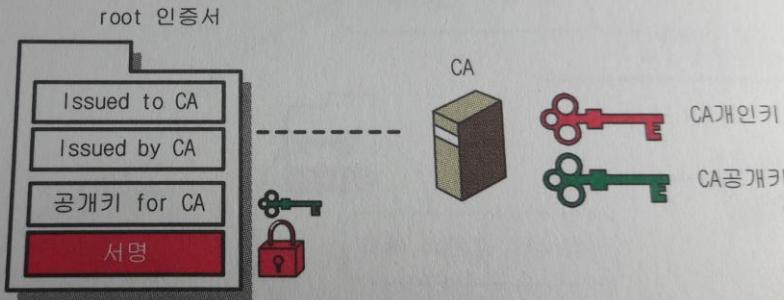
(*) B, C는 100자리 이상의 아주 큰 수임.

B. X.509 인증서(Certificate)(1)

X.509 공인 인증서: 공개키 + 전자 서명
✓ 상대방과의 인증을 위해 사용

서명이 침투되어 있거나, 그 인증서를 신뢰할 수 있는可信한 CA에 서명한 경우 이를 이용해 검사할 수 있다.

이 과정에서, CA의 개인 키로 암호화된 서명 부분을 확인하기 위해서는 CA의 공개 키이다. 이를 위하여, CA가 스스로 발급하고, CA 자신의 비밀 키에 의해 스스로 서명하였으나, CA는 자신의 개인 키로 서명하였다. 이렇게 CA 자신이 생성하여 자신을 수령한 특별한 인증서도 확보되어야 한다. 이렇게 CA 자신이 생성하여 자신을 루트인증서(Self-signed Digital Signature)라고 하며, 그 내용은 <그림 4.15>와 같다.

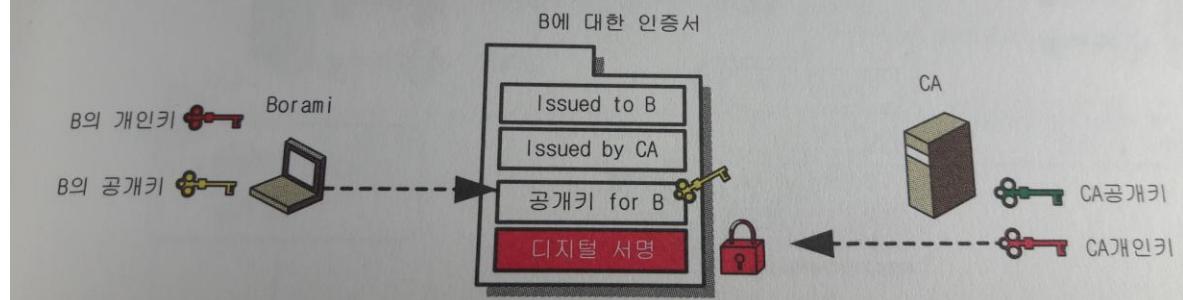


<그림 4.15> 루트 인증서의 예

이 인증서를 사용하여 Arami가 Borami에게 암호화된 메시지를 전송하는 절차는 다음과 같다.

(1) 공인 인증서

사용자의 공개 키를 신뢰성 있게 배포하기 위하여 {사용자 ID, 유효기간, 사용자의 공개 키, (Certificate Authority)의 개인 키에 의한 디지털 서명}으로 구성된 인증서를 발급하고 보관하는 절차에 대한 표준 중의 하나가 X.509이며, 이 절차에 의해 발급되는 인증서를 X.509 인증서라고 한다. 예를 들어, 사용자 B에 대한 인증서의 구성은 <그림 4.14>와 같다.

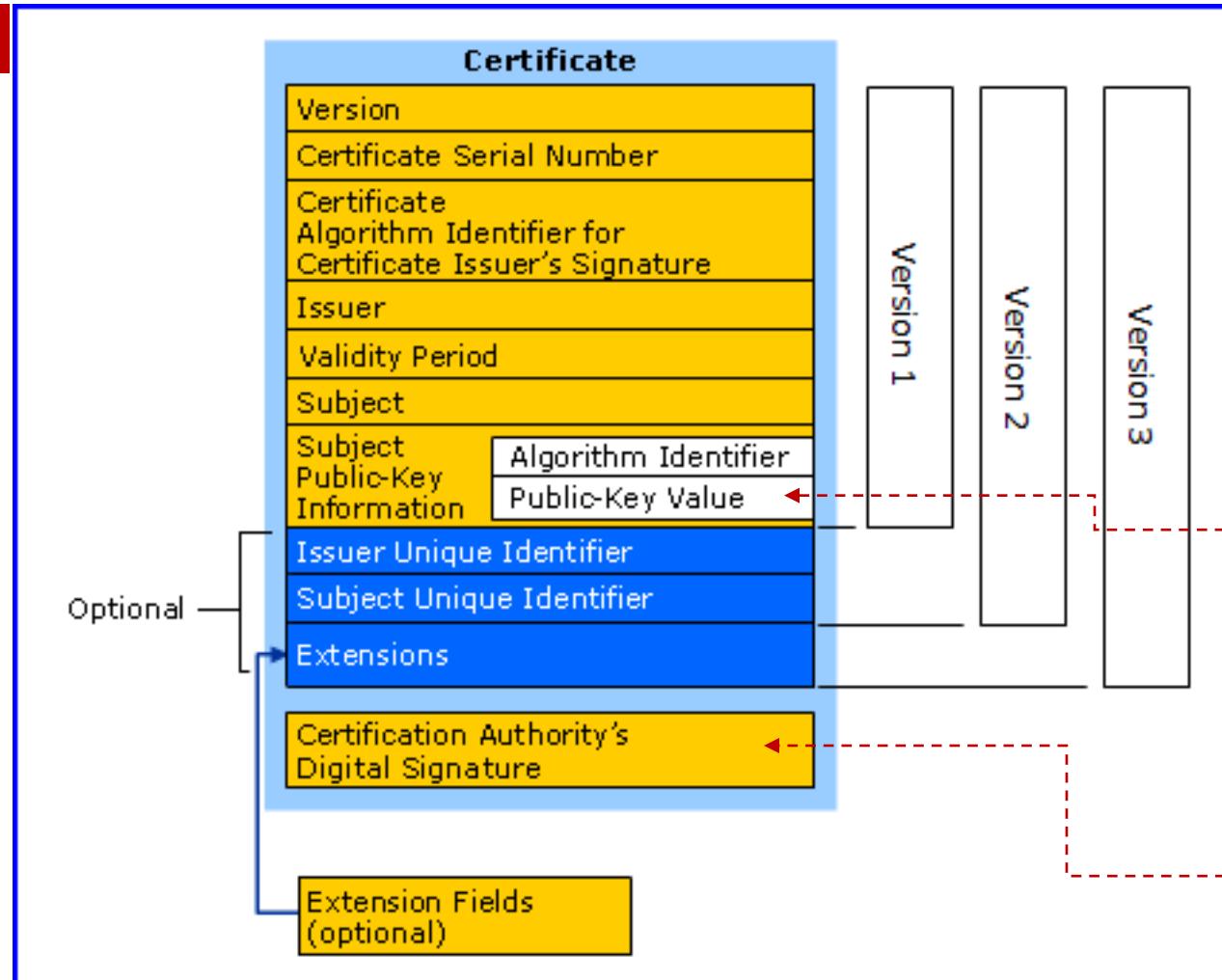


<그림 4.14> 사용자 B에 대한 인증서의 예

이 인증서에 첨부된 사용자 Borami의 공개 키는 노출되어 있지만, CA의 개인 키는 첨부되어 있으므로, 이 인증서를 수신한 측은 공개된 CA의 서명용 공개 키를 사용해 서명 여부를 확인할 수 있다.

B. X.509 인증서(Certificate)(2)

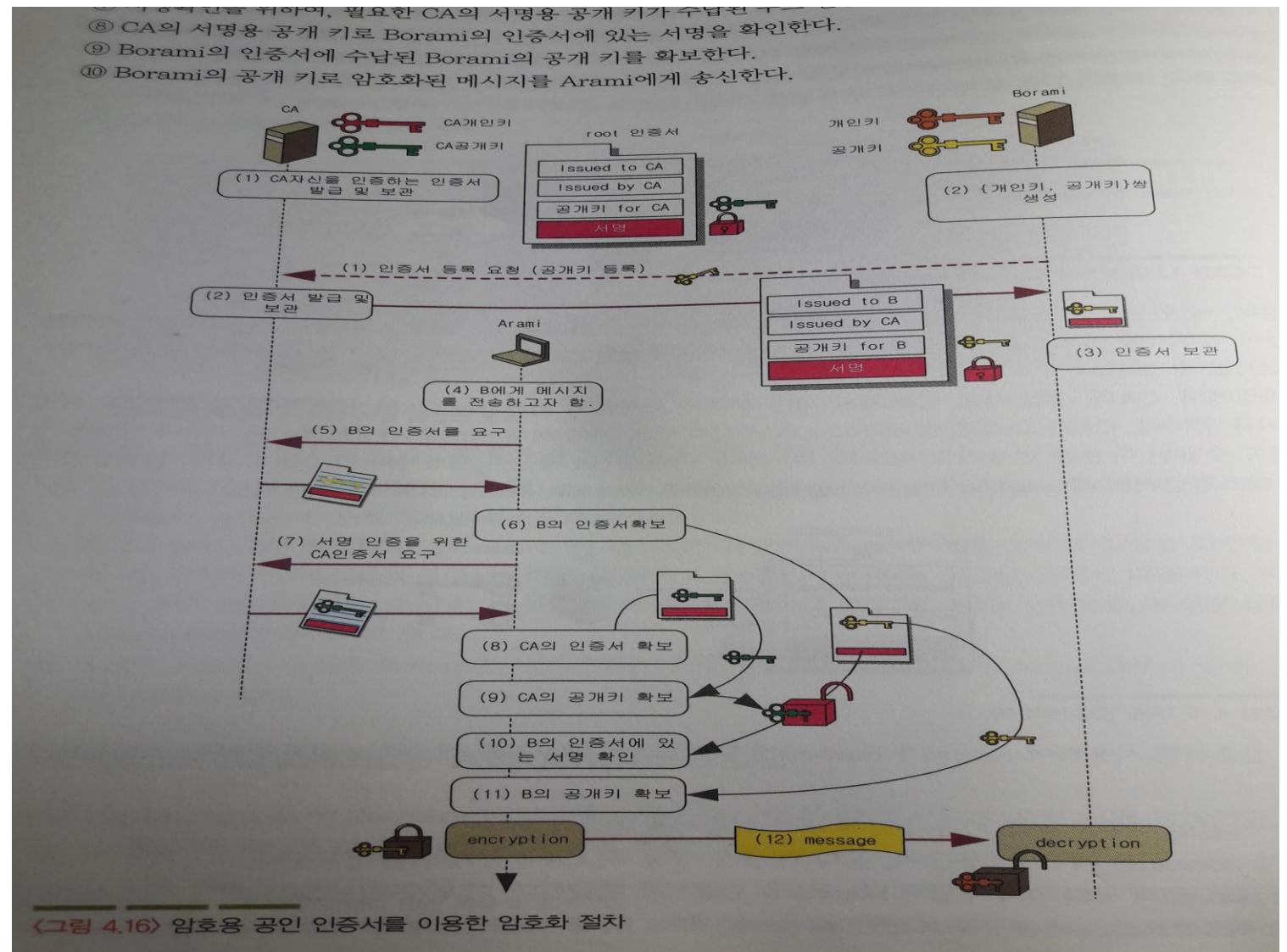
X.509 인증서



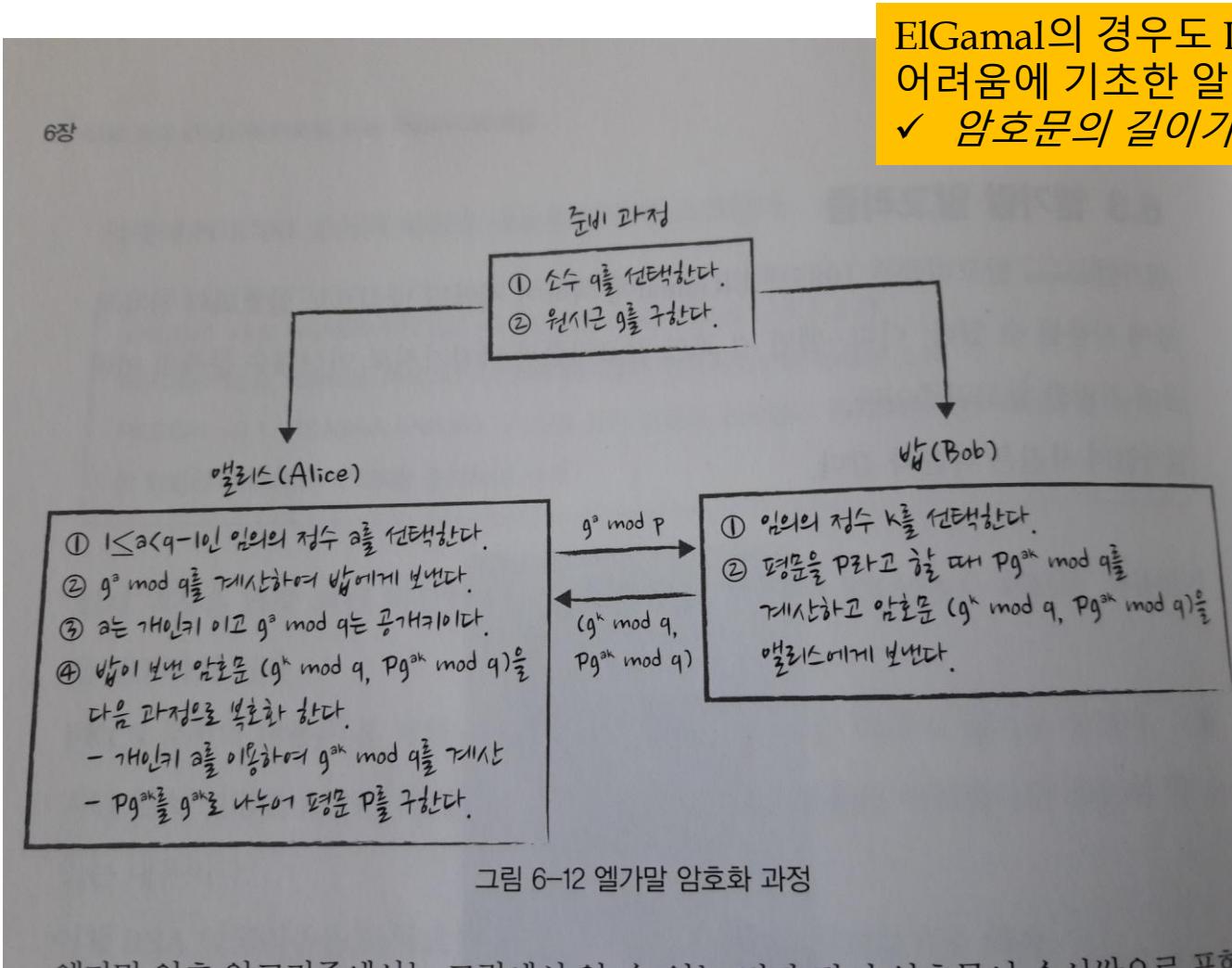
공개키

전자 서명(CA에 의한)

B. X.509 인증서(Certificate)(3)



C. Public Key Cryptography - *ElGamal*

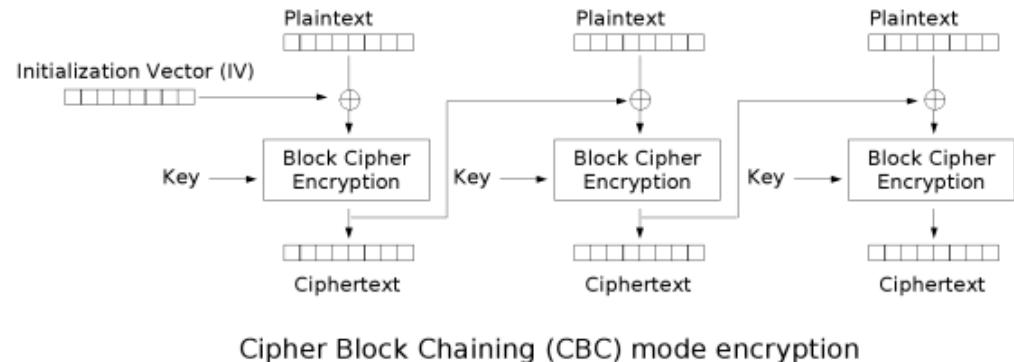


ElGamal의 경우도 DH처럼 이산 대수(log)의 어려움에 기초한 알고리즘이.
✓ 암호문의 길이가 평문의 두배가 되는 단점 보유

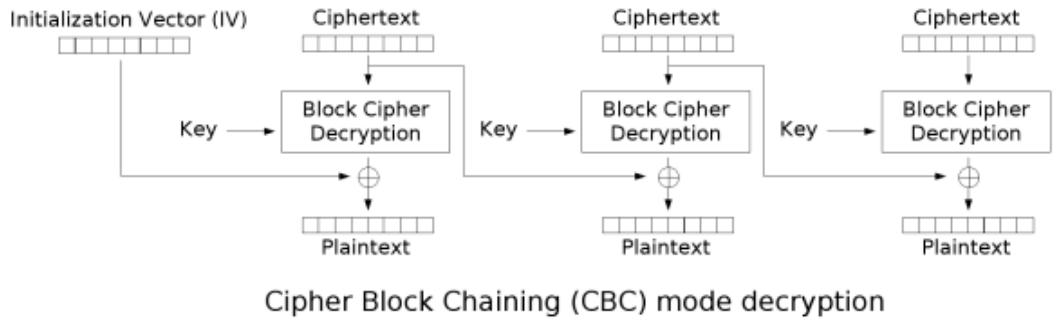
D. Symmetric Cipher Algorithm(1) - *CBC Mode*

CBC 방식은 각 block이 암호화되기 전에 이전 블록의 암호화 결과와 XOR되며, 첫 block의 경우에는 초기화 Vector가 사용된다.

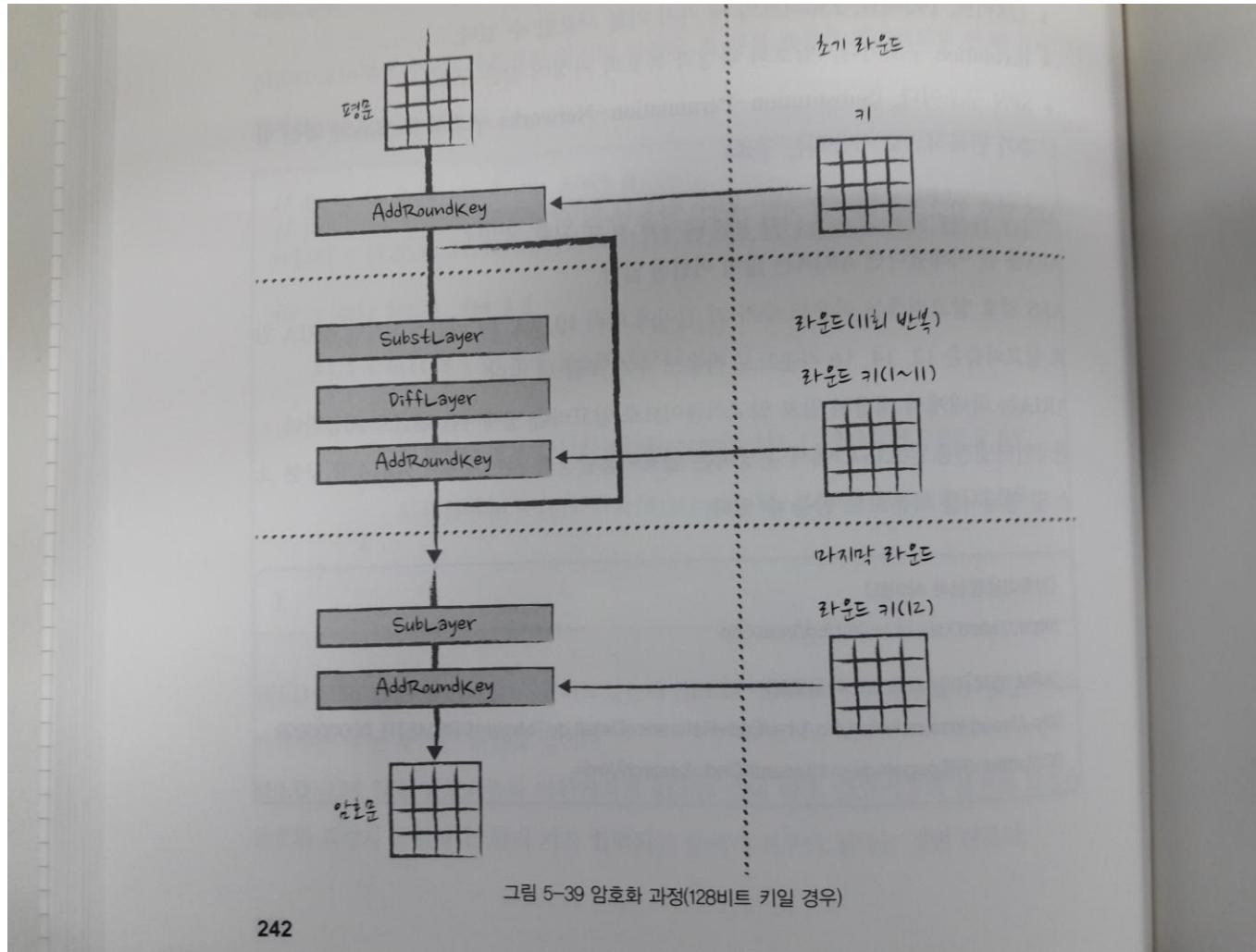
초기화 Vector가 같은 경우 Output 결과가 항상 같기 때문에, 매 암호화마다 다른 초기화 Vector를 사용해야 한다.



Block Cipher CBC mode 동작 원리



D. Symmetric Cipher Algorithm(2) - ARIA



References

- 1. 무선 LAN 보안 프로토콜, 윤종호, 교학사
- 2. 스토리로 이해하는 암호화 알고리즘, 김수민, 로드북
- 3. Implementing SSL_TLS Using Cryptography and PKI, Joshua Davies, Wiley Publishing, Inc
- 4. Network security: Private Communication in a Public World 2nd edition, Charlie Kaufman, Prentice Hall.
- 5. Cryptography Decrypted, H.X. Mel/Doris Baker
- 6. and google ~

Thank You