제 2 장 기초 암호화 기법

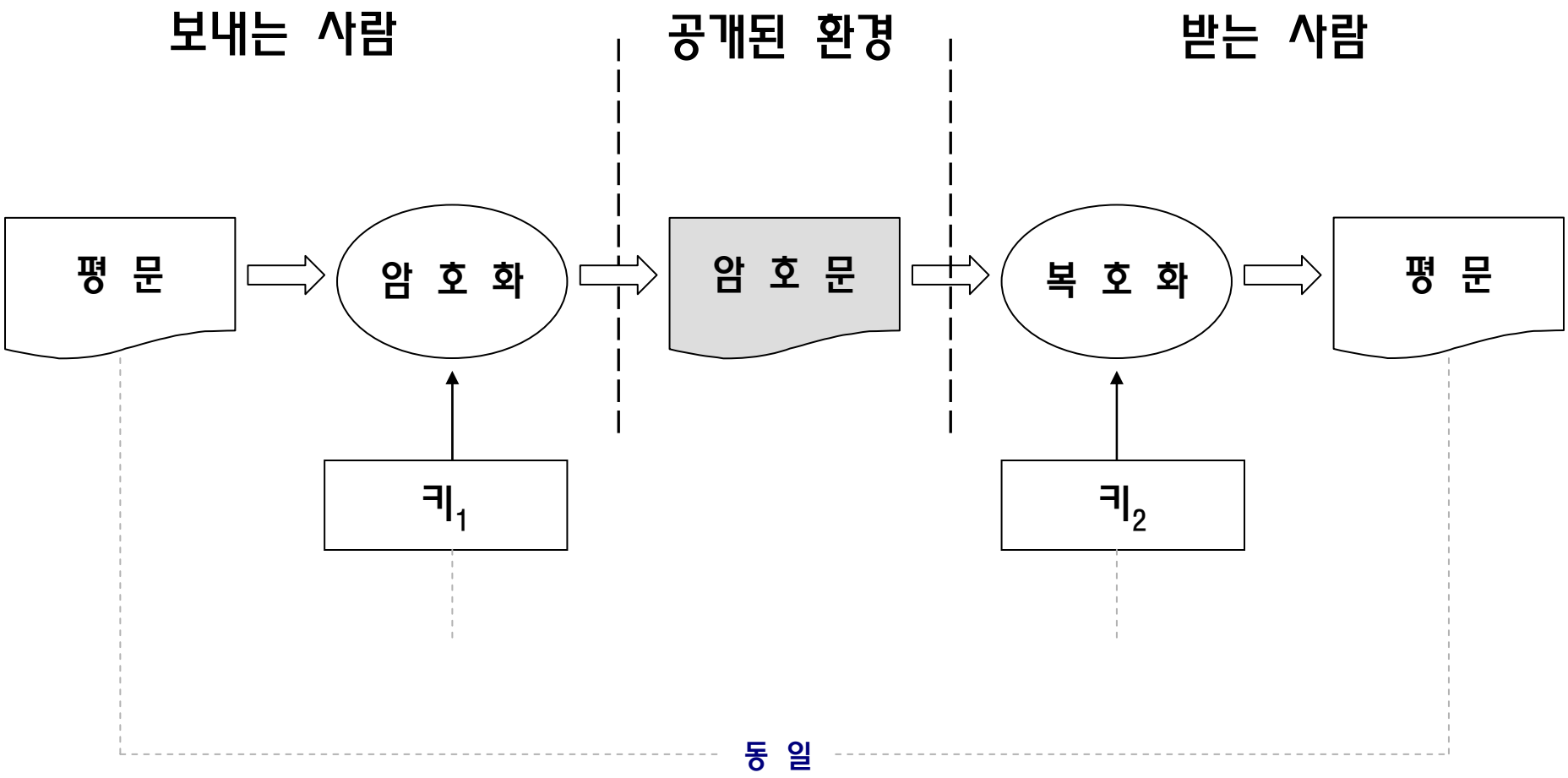
컴퓨터 시스템 보안

금오공과대학교 컴퓨터공학부
최태영

평문과 암호문

- 평문 (plaintext) : 암호화 되기 전의 *읽을 수 있는* 문장
- 암호문 (ciphertext) : 암호화에 의해서 *읽을 수 없게* 된 문장
- 암호화 (encryption) : 평문을 암호문으로 바꾸는 과정
  - 암호화 알고리즘 : 암호화 수행 과정
- 복호화 (decryption) : 암호문을 평문으로 바꾸는 과정
  - 복호화 알고리즘 : 복호화 수행 과정

암호작성법 (Cryptography)



암호 관련 추가 용어

- 암호 알고리즘 (cryptography algorithm)
  - 암호화 알고리즘 + 복호화 알고리즘
- 키 (key)
  - 암호 알고리즘에 사용되는 비밀 정보
  - 합법적인 사용자와 그렇지 않은 자를 구분
  - 복호화에 사용되는 키는 비밀 보관해야 함
- 암호해독 (cryptanalysis)
  - 키를 모르는 상태에서 암호문을 평문으로 바꾸거나 키를 찾아내는 것
- 암호해독자 (cryptanalist)
  - 암호해독을 시도하는 자

암호작성법의 수식화

$$C = E(P, K_e)$$

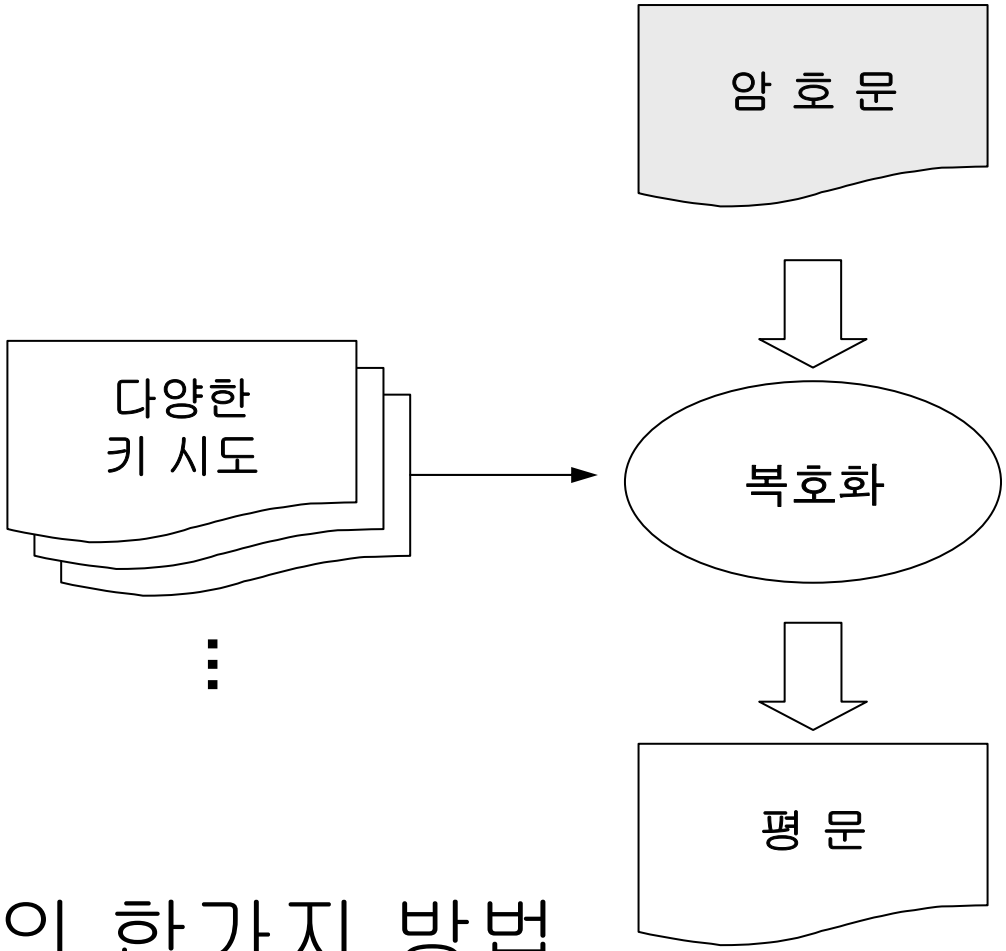
- P : 평문
- C : 암호문
- K_e : 암호화에 사용되는 키
- E : 암호화 알고리즘

$$P = D(C, K_d)$$

- K_d : 복호화에 사용되는 키
- D : 복호화 알고리즘

$$C' \neq C \text{ if } P' \neq P \text{ and } C' = E(P', K_e)$$

브루트포스 공격 (bruteforce attack)

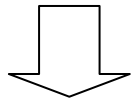


암호해독의 한가지 방법

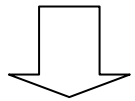
시저 암호 (Caesar cipher)

- 시저 암호 : 각 알파벳 문자를 자신의 세 번째 뒤 알파벳으로 바꾸는 암호 방식

```
general you attack the enemy at the west field at three o'clock am
```



a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c



```
jhghudo brx dwwdfn wkh hghpb dw wkh zhvw ilhog dw wkuhh r'forfn dp
```

시저 암호의 일반화

- 치환 암호 (substitution cipher)
- 단순 치환 암호 (simple substitution cipher)
  - *k*-번째 다음 알파벳으로 변경
  - 1-번째 알파벳 암호화: $t'_i = (t_i + k) \bmod 26$
  - 공격이 쉬움 (다음 페이지)
  - 키 공간이 작음 : 가능한 키의 개수가 25개
- 모노알파벳 암호 (monoalphabetic cipher)

암호문	u	s	b	s	f	o	z	m	c	i
K=1	t	r	a	r	e	n	y	l	b	h
K=2	s	g	z	g	d	m	x	k	a	g
.										
.										
.										
K=12	i	g	p	g	t	c	n	a	q	w
K=13	h	f	o	f	s	b	m	z	p	v
K=14	**g**	**e**	**n**	**e**	**r**	**a**	**l**	**y**	**o**	**u**
K=15										
.										
.										
.										

모노알파벳 암호

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
M	G	F	N	O	P	S	Z	E	T	Y	W	B	U	I	V	R	Q	X	A	D	J	H	L	K	C

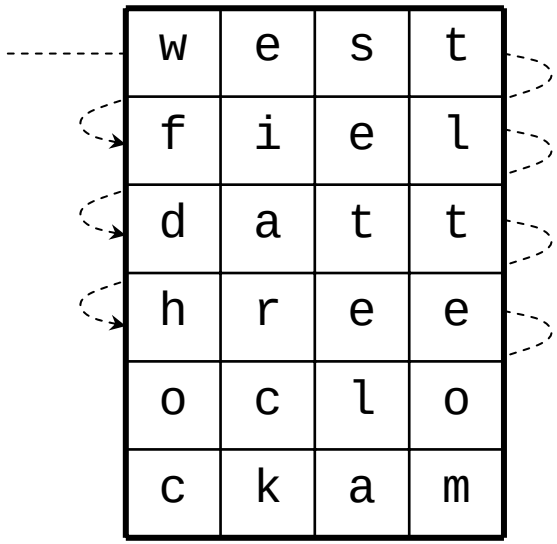
- 각 알파벳을 다른 알파벳으로 바꾸는 암호화 방식
- 키 공간의 크기는 $26! \approx 4 \times 10^{26}$
  - 초당 10^{10}개 키 처리 컴퓨터에서 10^{11}년 소요
- 사용빈도가 높은 알파벳이나 문자열이 존재: E, T, TH, ST, ...
  - 암호문에서 사용빈도가 높은 알파벳을 E, T 등으로 교체 시도

전치 암호 (transposition cipher)

- 평문에서 알파벳들의 위치를 어떤 규칙을 사용하여 변경하는 암호 알고리즘
- 대표적인 전치 암호로는 행렬에 열 방향으로 문자들을 입력한 뒤 행 방향으로 읽어서 암호문으로 출력하는 방법이 있음
- Columnar 전치 암호
  - 열의 입력 순서와 행의 출력 순사가 다름

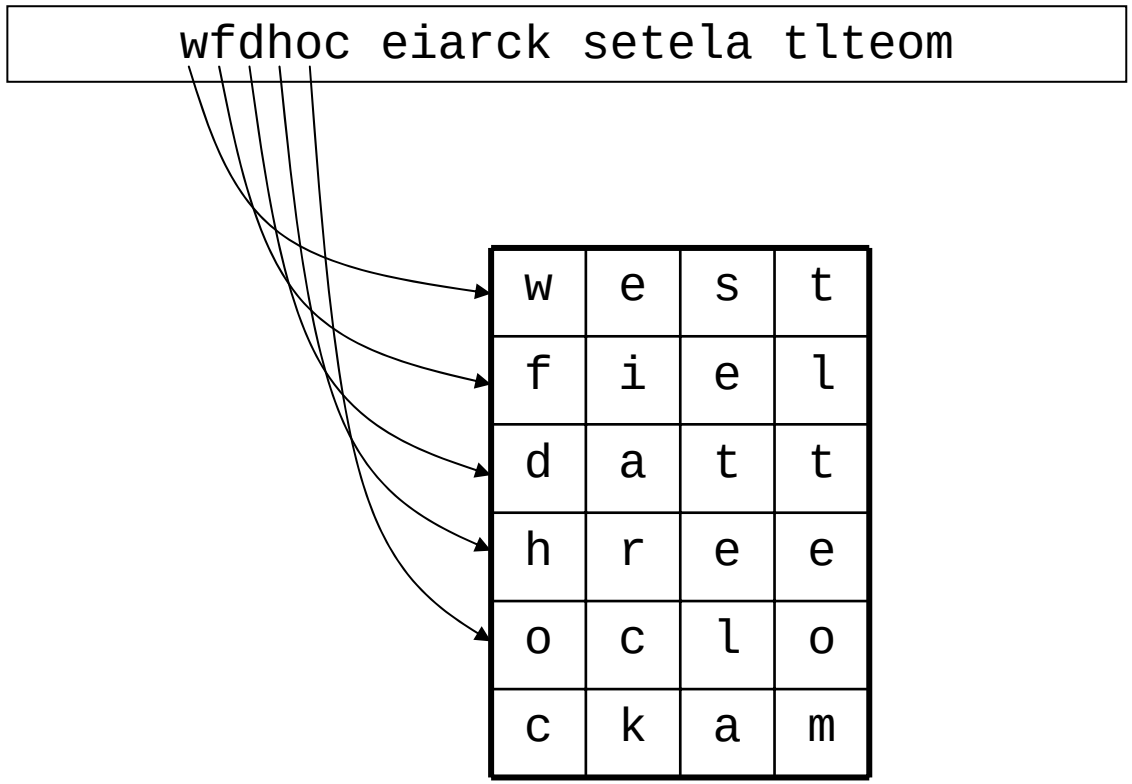
전치암호의 암호화

west field at three oclock am



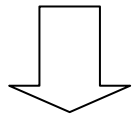
wfdhoc eiarck setela tlteom

전치암호의 복호화



Columnar 전치암호의 예

	2	1	3	4
4	h	r	e	e
3	d	a	t	t
1	w	e	s	t
5	o	c	l	o
2	f	i	e	l
6	c	k	a	m

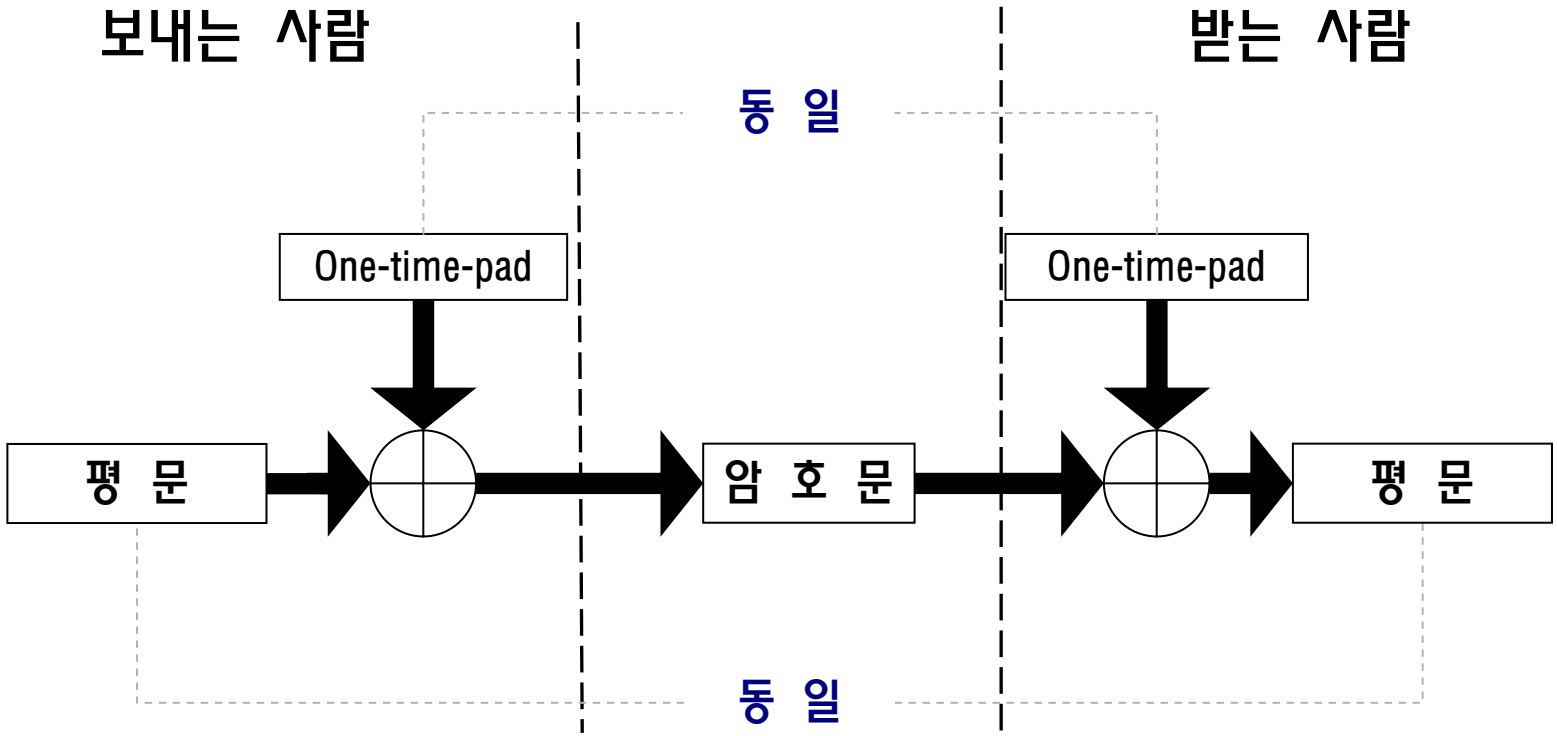


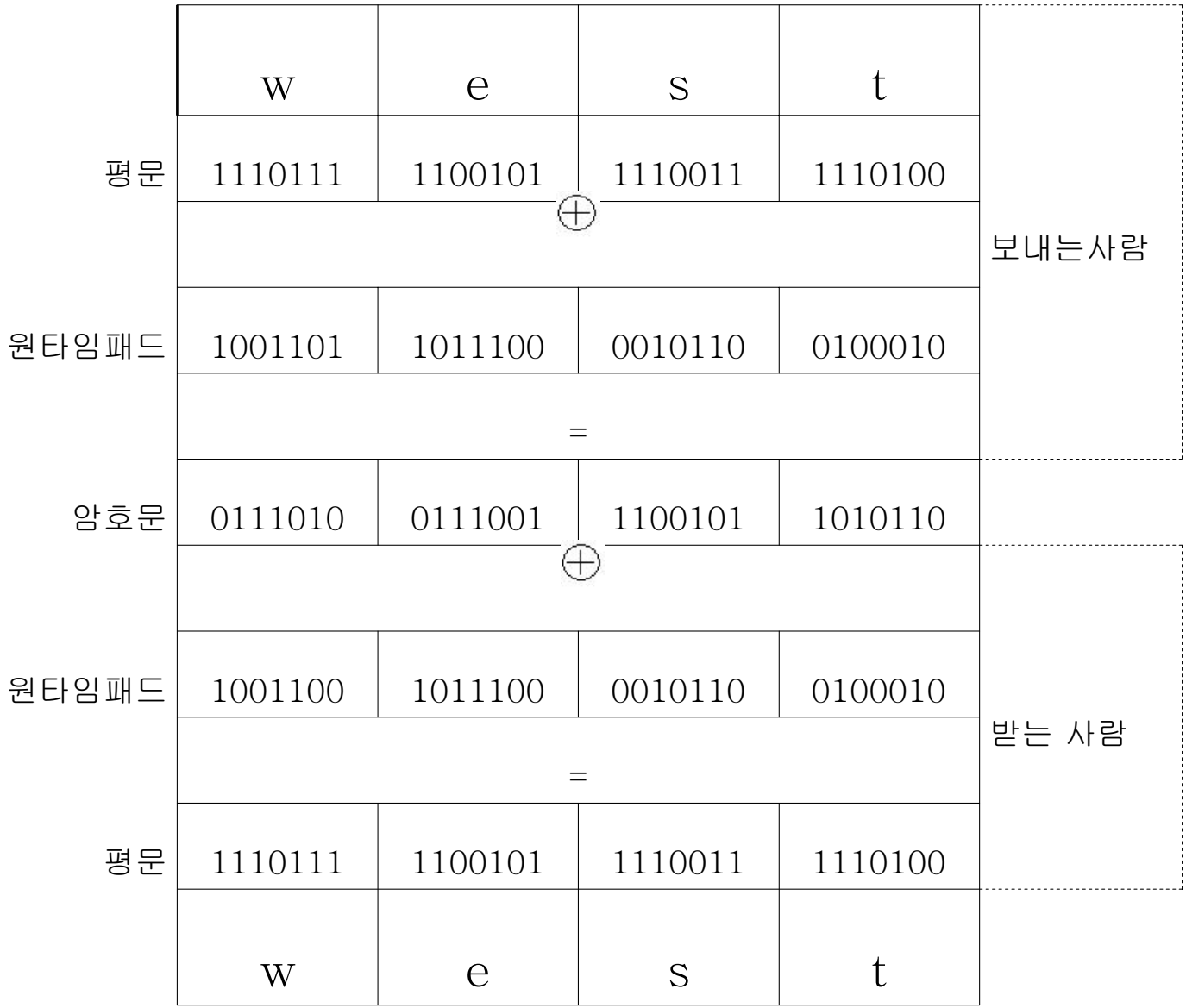
raecik hdwofc etslea ettolm

일회성 암호

- One-Time Pad, OTP
- Vernam cipher
- 평문의 각 비트를 난수 비트 스트림 (ramdom bit stream)과 xor 연산을 수행

일회성 암호의 구조

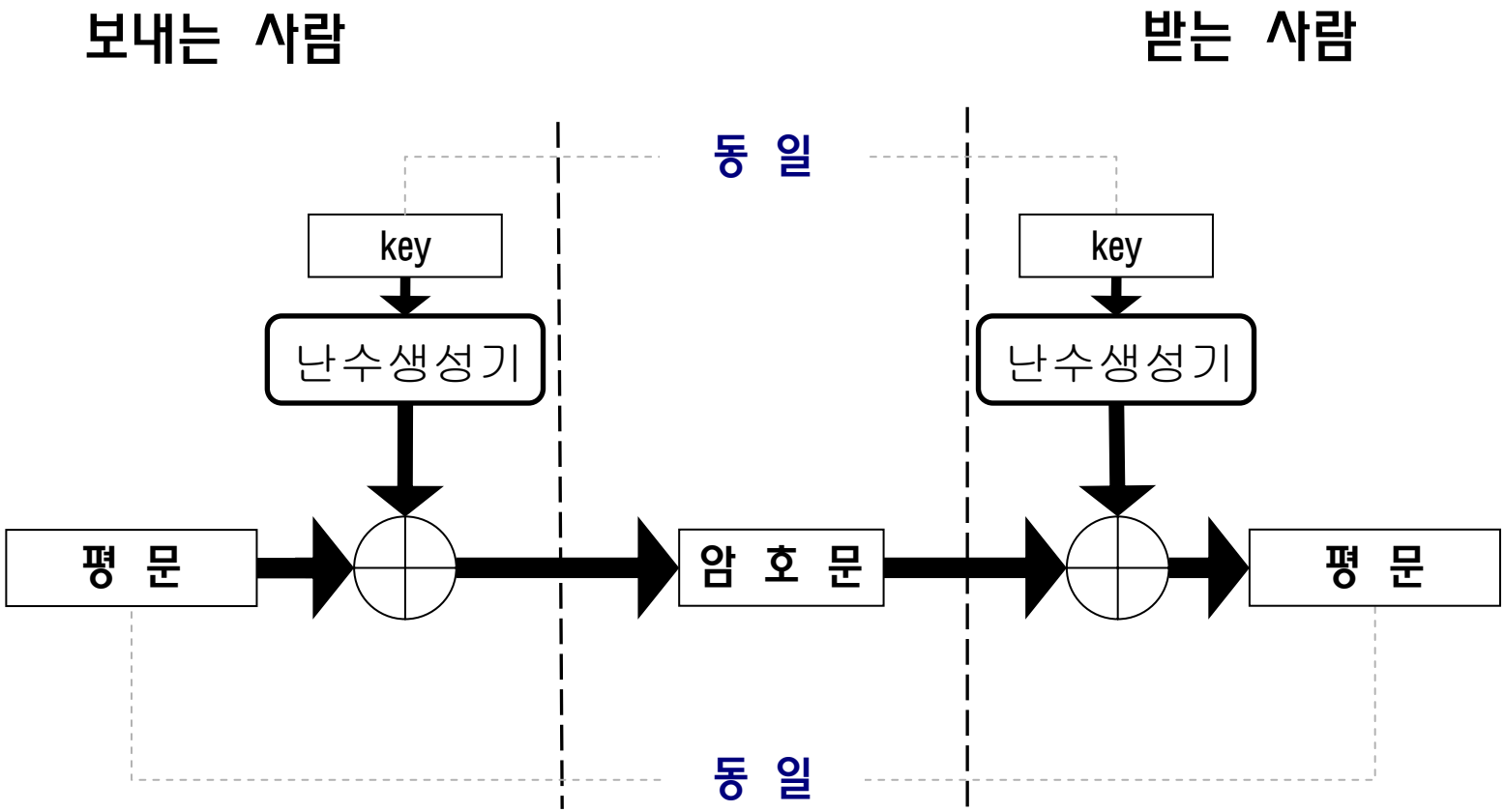



w e s t
평문
1110111 1100101 1110011 1110100
⊕
보내는사람
원타임패드
1001101 1011100 0010110 0100010
=
암호문
0111010 0111001 1100101 1010110
⊕
원타임패드
1001100 1011100 0010110 0100010
=
받는 사람
평문
1110111 1100101 1110011 1110100
w e s t

일회성 암호의 문제점

- 난수 비트 스트림은 평문의 크기와 같아야 함
- 난수 비트 스트림을 상대방에게 **안전**하게 제공하는 것은 어려움
- 유사 난수 스트림을 사용함 → 안전성 하락

변형된 일회성 암호



키의 특성에 따른 암호 분류

- 대칭키 암호 (Symmetric cipher)
  - 암호화와 복호화에 사용되는 키가 **동일**하거나, 한 키로부터 다른 키를 쉽게 생성
- 비대칭키 암호 (Asymmetric cipher)
  - 암호화와 복호화에 사용되는 키가 **다르**며, 한 키로부터 다른 키를 유도하기가 어려움
  - 공개키 : 암호화에 사용, 공개되는 키
  - 개인키 : 복호화에 사용, 비공개

암호화 처리 크기에 따른 분류

- 블록암호 (block cipher)
  - 2 비트 이상의 일정크기 평문 블록을 한꺼번에 암호화
  - 혼돈과 확산이 잘됨
- 스트림암호 (stream cipher)
  - 한 번에 한 비트씩 암호화
  - 일회성암호
  - 혼돈만 적용됨

혼돈과 확산

- 암호학자 Claude Shannon이 제안
- 혼돈 (confusion)
  - 키와 암호문이 얽혀 있는 정도
- 확산 (diffusion)
  - 평문이 암호문에 흩뿌려져 있는 정도
  - 전송되는 암호문에 오류가 발생할 경우 그 오류가 전파 (propagate)되는 경향이 있음

OpenSSL의 키 생성

- 커맨드라인 명령어를 통한 키 생성
  - 대칭키 생성
  - 공개키 생성 (RSA 키 생성)
- 라이브러리를 이용한 키 생성
  - 대칭키 생성
  - 공개키 생성

대칭키 생성

- 예제: openssl rand 8 -out key.sec
- 문법: **openssl rand** [**-out** *file*] [**-rand** *file*(s)] [**-base64**] *num*
  - -out : 저장 파일 지정
  - -rand : 시드 파일 지정
  - -base64 : 출력가능 형태 여부 지정
  - num : 생성할 키의 바이트 길이

RSA 공개키 생성

- 예시 : openssl genrsa -out privkey.pem
- 문법 : openssl genrsa [-out *file*] [-passout *arg*] [-des] [-des3] [-idea] [-f4] [-3] [-rand *file*(s)] [-engine *id*] [*numbits*]
  - -out : 개인키 저장 파일 명시
  - -passout : 개인키 파일 암호화 키 명시
  - -rand : 난수 시드 파일 명시
  - -engine : 사용자 개발 암호 알고리즘 명시
  - numbits : 모듈러스 크기 명시

RSA 키 관리

- 예시 : openssl rsa -pubout < privkey.pem > pubkey.pem
- 문법 : openssl rsa [-inform PEM|NET|DER] [-outform PEM|NET|DER] [-in *file*] [-passin *arg*] [-out *file*] [-passout *arg*] [-sgckey] [-des] [-des3] [-idea] [-text] [-noout] [-modulus] [-check] [-pubin] [-pubout] [-engine *id*]
  - -inform -outform : 키 파일 저장 형식

라이브러리를 통한 비밀키 생성

- 예시
  - RAND_seed(seedbuf, 8);
  - RAND_bytes(randbuf, 16);
- 설명
  - seedbuf는 난수 시드를 포함하며 난수 자료구조에 시드값을 전달하는데 사용.
  - randbuf는 생성된 난수가 저장되는 공간
- seedbuf를 만드는 방법으로 현재시간을 사용할 수 있음

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include <string.h>
#include <assert.h>
#include <openssl/rand.h>
#define MAXBUFF 128
void printStr(void *buf, int size){
   /* 한 캐릭터씩 16진수로 출력하는 함수 */
}
void getTimeSubstr(char buff[])
{
   struct timeval atime;
   struct timezone tzone;
   gettimeofday(&atime, &tzone);
   memcpy(buff, &(atime.tv_sec), 4);
   memcpy(buff+4, &(atime.tv_usec), 4);
}
```

```
int main(void)
{
   char seedbuf[MAXBUFF];
   char randbuf[MAXBUFF];

   getTimeSubstr(seedbuf);
   RAND_seed(seedbuf, 8);
   RAND_bytes(randbuf, 16);
   printStr(randbuf, 16);
   exit(0);
}
```

라이브러리를 통한 공개키 생성

- 예시
  - rsaPriv = RSA_generate_key(512, RSA_F4, NULL, NULL);
  - rsaPub = RSAPublicKey_dup(rsaPriv);
  - PEM_write_RSAPublicKey(pubf, rsaPub);
  - PEM_write_RSAPrivateKey(privf, rsaPriv, NULL, NULL, 0, NULL, NULL)

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include <string.h>
#include <assert.h>
#include <openssl/rand.h>
#include <openssl/rsa.h>
#include <openssl/pem.h>
#define MAXBUFF 128
void getTimeSubstr(char buff[])
{
  struct timeval atime;
  struct timezone tzone;

  gettimeofday(&atime, &tzone);
  memcpy(buff, &(atime.tv_sec), 4);
  memcpy(buff+4, &(atime.tv_usec), 4);
}
```

```
int main(void)
{
  RSA *rsaPriv = NULL, *rsaPub = NULL;
  char seedbuf[MAXBUFF];
  FILE *pubf, *privf;

  getTimeSubstr(seedbuf);
  RAND_seed(seedbuf, 8);

  rsaPriv = RSA_generate_key(512, RSA_F4, NULL, NULL);
  if (rsaPriv == NULL){
    fprintf(stderr, "RSA generate key error.\n");
    return(0);
  }
  rsaPub = RSAPublicKey_dup(rsaPriv);
  if (rsaPub == NULL){
    fprintf(stderr, "RSA public key copy error.\n");
    return(0);
  }
```

```
  pubf = fopen("pubkey.pem", "w"); assert(pubf);
  privf = fopen("privkey.pem", "w"); assert(privf);
  if (!PEM_write_RSAPublicKey(pubf, rsaPub))
    fprintf(stderr, “public key to file fails.\n");
  if (!PEM_write_RSAPrivateKey(privf, rsaPriv, NULL,
NULL, 0, NULL, NULL))
    fprintf(stderr, “private key to file
fails.\n");
  fclose(pubf);
  fclose(privf);

  RSA_free(rsaPriv);
  RSA_free(rsaPub);
  exit(0);
}
```