교과목 : 정보보호

6. Secrete sharing, Randomness

2023학년도 2학기
Suk-Hwan Lee

Topics to Discuss

- Shamir's secret sharing scheme
  - ✓ Simple procedure to split a secret among users
- Proactive secret sharing

- Randomness
  - ✓ Need random keys, or random large prime
  - ✓ Discuss some of problems of actually generating random numbers

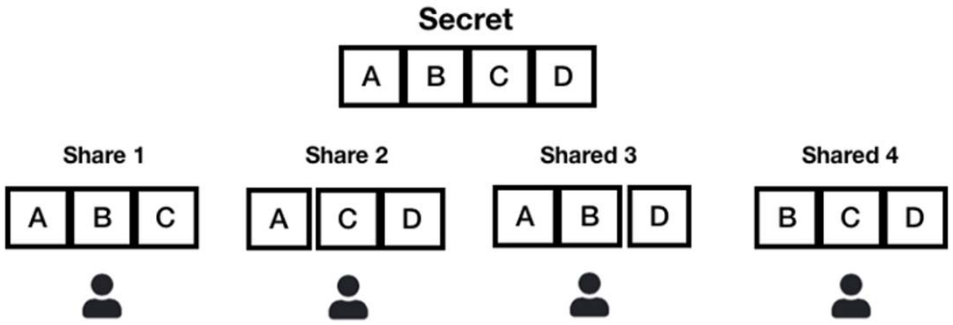
- Image/Video Encryption / Hashing
- Information hiding : Steganography, Watermarking
- *Future study topic*

Secret sharing

Secret sharing

- Password나 Secret Key 보호가 매우 중요함
- Secret Key 유실 막기 위하여 여러 곳에 백업
  - ✓ Secret key 유출 가능성이 커짐

➢ 다음 조건을 만족하는 Secret Sharing이 연구되어 왔음
  - ❖ Shares : 하나의 secret을 여러 조각으로 분산시켜 저장함
  - ❖ Threshold : 원래 secret을 Recover 하기 위해서는 반드시 일정한 수 이상의 share가 필요함



Problem

- 사람의 수와 threshold값이 커질수록 필요한 조각의 수가 매우 커진다. 100명의 사람이 secret을 나누어 갖고, threshold를 50으로 설정하게 되면 secret은 $\binom{100}{50} \approx 1.009 \times 10^{29}$개가 필요하고, 각 사람은 $\binom{99}{49} \approx 5.04 \times 10^{28}$개의 조각을 가지고 있어야 한다.

출처 : Secret Sharing – 비밀 공유하기, https://medium.com/codechain-kr/secret-sharing-비밀-공유하기-356fd756323a

1. Shamir's secret sharing

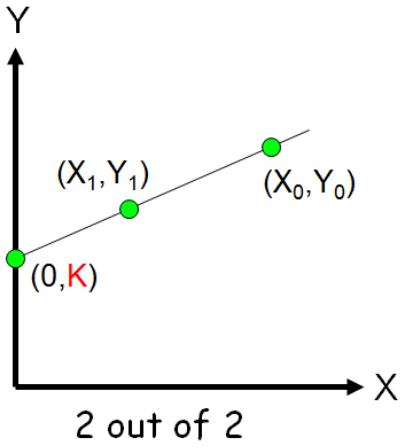
- Efficient Secret Sharing
  - Shamir's scheme
  - Blakley's scheme
  - Using the Chinese remainder theorem

- Proactive Secret Sharing (사전 비밀 공유) — 불규칙적인 주기로 참가자들에게 배포된 공유(share)값을 새롭게 변경
- Verifiable Secret Sharing (VSS) (1987) — Shared secret 소유자들이 자신의 share를 검증할 수 잇음
- Publicly Verifiable Secret Sharing (2004) — 자신의 Share뿐만 아니라 다른 사람의 Share도 검증 가능
- Computationally Secure Secret Sharing
- Multi-secret and space efficient (batched) secret sharing

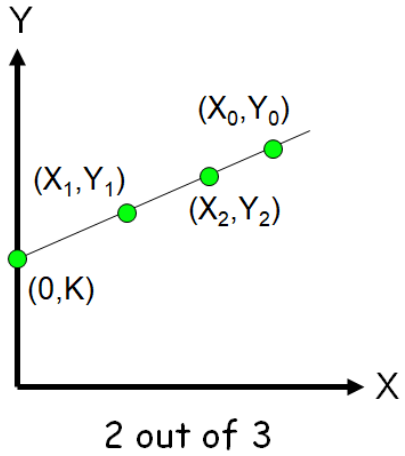
출처 : Wikipedia, Secret Sharing

Shamir's Secret sharing (SSS)

- 임의의 좌표값이 서로 다른 t개의 점을 지나는 x의 t-1차 다항식은 유일하게 결정된다는 사실을 이용



- Let symmetric key is K
- Two points determine a line
- Give (X_0,Y_0) to Alice
- Give (X_1,Y_1) to Bob
- Then Alice and Bob must cooperate to find secret S
- Also works in discrete case
- Easy to make "m out of n" scheme for any $m \leq n$

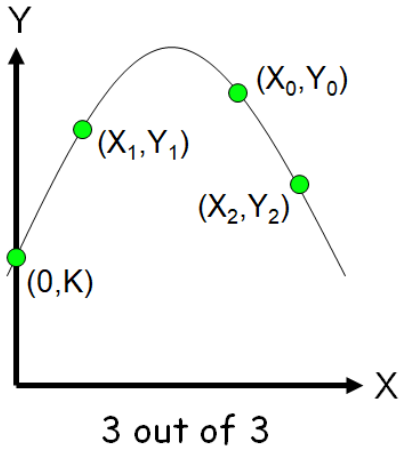


- Give (X_0,Y_0) to Alice
- Give (X_1,Y_1) to Bob
- Give (X_2,Y_2) to Charlie
- Then any two of Alice, Bob and Charlie can cooperate to find secret S
- But no one can find secret S
- A "2 out of 3" scheme

출처 : Secret Sharing - 비밀 공유하기, https://medium.com/codechain-kr/secret-sharing-비밀-공유하기-356fd756323a

Shamir's Secret sharing (SSS)

- 임의의 좌표값이 서로 다른 t개의 점을 지나는 x의 t-1차 다항식은 유일하게 결정된다는 사실을 이용



- Give (X_0,Y_0) to Alice
- Give (X_1,Y_1) to Bob
- Give (X_2,Y_2) to Charlie
- 3 points determine a parabola
- Alice, Bob and Charlie must cooperate to find secret S
- A "3 out of 3" scheme
- Can you make a "3 out of 4" ?

출처 : Secret Sharing - 비밀 공유하기, https://medium.com/codechain-kr/secret-sharing-비밀-공유하기-356fd756323a

Shamir's Secret sharing (SSS)

- Sharing 단계

1) S는 비밀값이고, 비밀값을 복구하기 위해 모여야 하는 정보의 수 t와 비밀을 나눠줄 사람 수 n을 정한다. (t<=n)

2) 다항식 q(x)를 임의로 결정한다 (a_0=S)

$$q(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \qquad (a_0 = S \mid a_1, \ldots, a_{t-1} \geq 1)$$

3) 다항함수 y = q(x) 위에 존재하는 n개의 점 (1, q(1)), (2, q(2)), …, (n, q(n)) 들은 shared secret이 된다.

(t, n) threshold secret sharing : n개의 비밀 조각들을 제공하고 비밀을 재구성하려면 최소 t조각의 임계 값이 필요하다. 이게 바로 SSS의 기본 이론이다. t-1차 다항식이 후보로 존재하여 원래의 다항식을 추측할 수 없다. 즉, t개보다 적은 비밀 조각이 모이더라도 원래의 비밀 값은 알 수 없다.

출처 : Secret Sharing – 비밀 공유하기, https://medium.com/codechain-kr/secret-sharing-비밀-공유하기-356fd756323a
SSS, Shamir's Secret Sharing) – 이론, https://ehdvudee.tistory.com/27

Shamir's Secret sharing (SSS)

- Recovering 단계

1) 다항함수 y= q(x) 위에 있는 t개의 점(shared secret)을 모은 뒤 위 polynomial interpolation을 통해 q(x)를 알아낸다.

2) q(0)은 원래 비밀값 S이다.

Polynomial interpolation은 주어진 점들을 모두 지나는 다항식을 찾는 것으로, 아래 식과 같다

$$q(x) = \frac{(x-x_1)(x-x_2)...(x-x_m)}{(x_0-x_1)(x_0-x_2)...(x_0-x_m)}y_0 + \frac{(x-x_0)(x-x_2)...(x-x_m)}{(x_1-x_0)(x_1-x_2)...(x_1-x_m)}y_1 + ... + \frac{(x-x_0)(x-x_{21})...(x-x_{m-1})}{(x_m-x_1)(x_m-x_2)...(x_m-x_{m-1})}y_m$$

$$q(x) = \sum_n^{i=0}(\prod_{0\le j\le n, j\ne i}\frac{x-x_j}{x_i-x_j})y_i$$

라그랑주 보간법 (Lagrangian Interpolation)

출처 : Secret Sharing - 비밀 공유하기, https://medium.com/codechain-kr/secret-sharing-비밀-공유하기-356fd756323a

Shamir's Secret sharing (SSS)

- 특징

✓ 모든 구성원이 나눠갖는 비밀이 서로 유일(unique)하며 평등

✓ threshold 이상의 조각만 있으면 secret를 복구할 수 있기 때문에 일부 비밀 조각이 유실되더라도 안전하고, 소수의 배신자가 있더라도 secret을 복구할 수 없다.

✓ 모든 shared secret가 동등하므로, 비밀을 나눌 때 보안 등급에 따라 나누기가 힘듬

✓ 비밀 조각을 제공할 때 일부러 틀린 비밀값을 주는 것을 방지할 수 없음. 비밀 복구시 누군가 오염된 값을 줘서 비밀이 제대로 복구되었는지 알 수 없음

출처 : Secret Sharing - 비밀 공유하기, https://medium.com/codechain-kr/secret-sharing-비밀-공유하기-356fd756323a

Shamir's Secret sharing (SSS) 예시

Sharing

- Secret : 1234 (S=1234)
- 6개 조각(n=6)으로 분리, 6조각 중 3조각(t=3)이 모이면 Secret 생성

$$q(x) = 1234 + 166x + 94x^2 \quad (a_0 = 1234;\ a_1 = 166; a_2 = 94)$$

- 6개 조각 공유 조각 배포 $D_i = (x_i, q(x_i))$

$$D_0 = (1,1494), D_1 = (2,1942), D_2 = (3,2578), D_3 = (4,3402), D_4 = (5,4414), D_5 = (6,5614))$$

출처 : SSS, Shamir's Secret Sharing) – 이론, https://ehdvudee.tistory.com/27
https://github.com/timtiemens/secretshare

Shamir's Secret sharing (SSS) 예시

Recovering

- Secret은 3개의 조각만 있으면 만들 수 있음. 2,4,5를 통해 Secret 생성 예시

$$(x_0, y_0) = (2,1942), (x_1, y_1) = (4,3402), (x_2, y_2) = (5,4414)$$

- 라그랑주 다항식

$$l_0 = \frac{x-x_1}{x_0-x_1} \cdot \frac{x-x_2}{x_0-x_2} = \frac{x-4}{2-4} \cdot \frac{x-5}{2-5} = \frac{1}{6}x^2 - \frac{3}{2}x + \frac{10}{3}$$

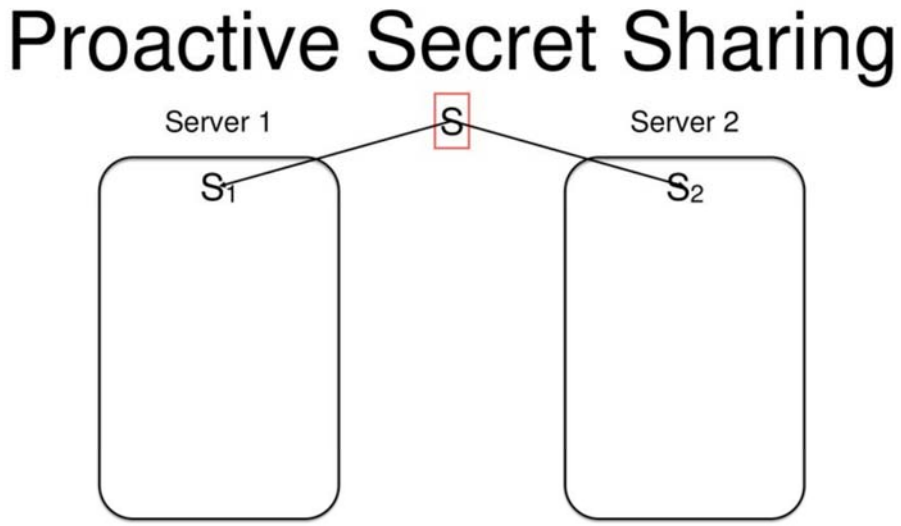
$$l_1 = \frac{x-x_0}{x_1-x_0} \cdot \frac{x-x_2}{x_1-x_2} = \frac{x-2}{4-2} \cdot \frac{x-5}{4-5} = -\frac{1}{2}x^2 + \frac{7}{2}x - 5$$

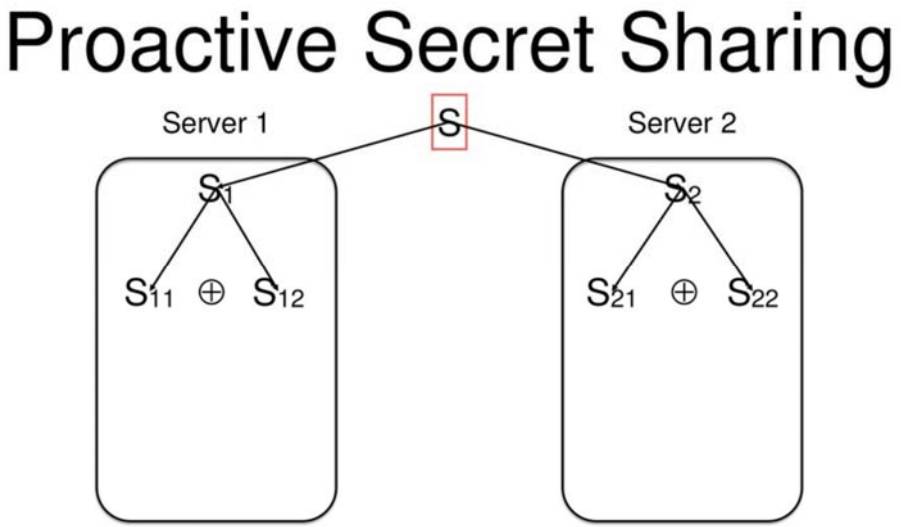
$$l_2 = \frac{x-x_0}{x_2-x_0} \cdot \frac{x-x_1}{x_2-x_1} = \frac{x-2}{5-2} \cdot \frac{x-4}{5-4} = \frac{1}{3}x^2 - 2x + \frac{8}{3}$$

$$q(x) = \sum_{i=0}^{2} y_i l_i(x)$$
$$= 1942 l_0(x) + 3402 l_1(x) + 4414 l_2(x)$$
$$= 1234 + 166x + 94x^2$$

출처 : SSS, Shamir's Secret Sharing) – 이론, https://ehdvudee.tistory.com/27
https://github.com/timtiemens/secretshare

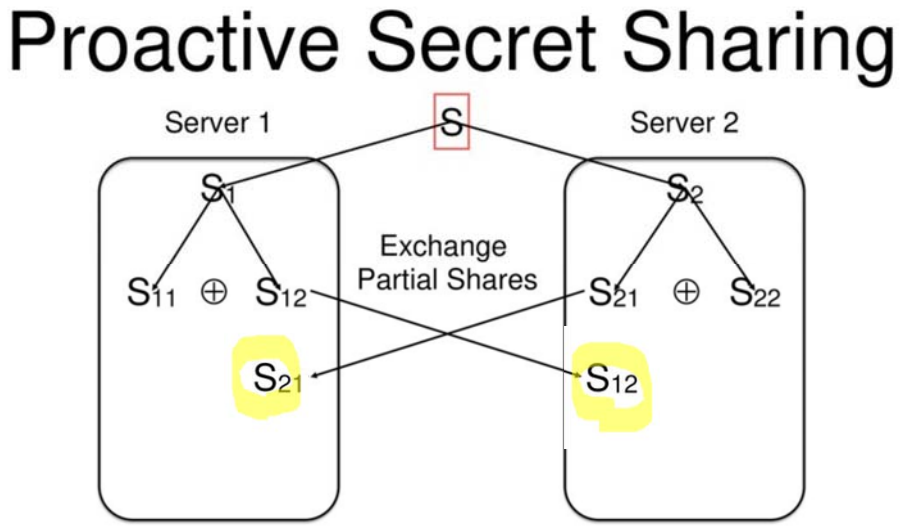
2. Proactive secret sharing

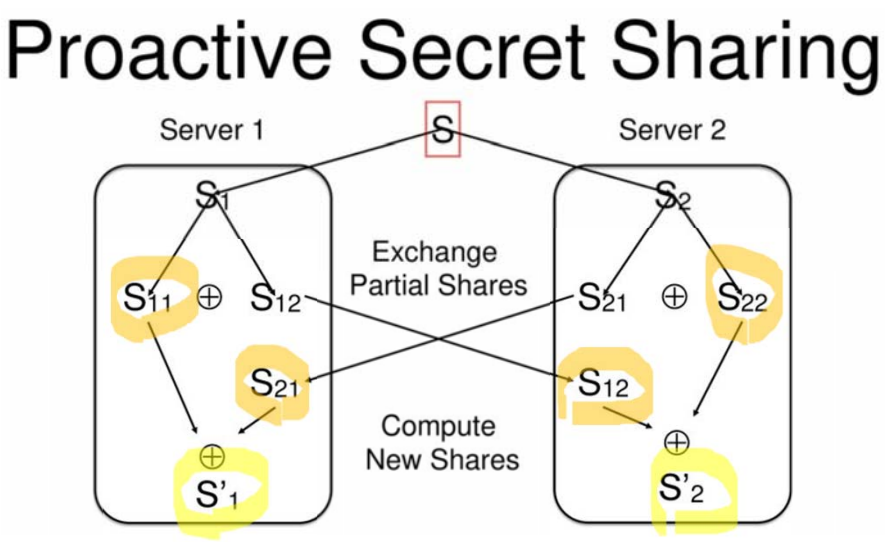




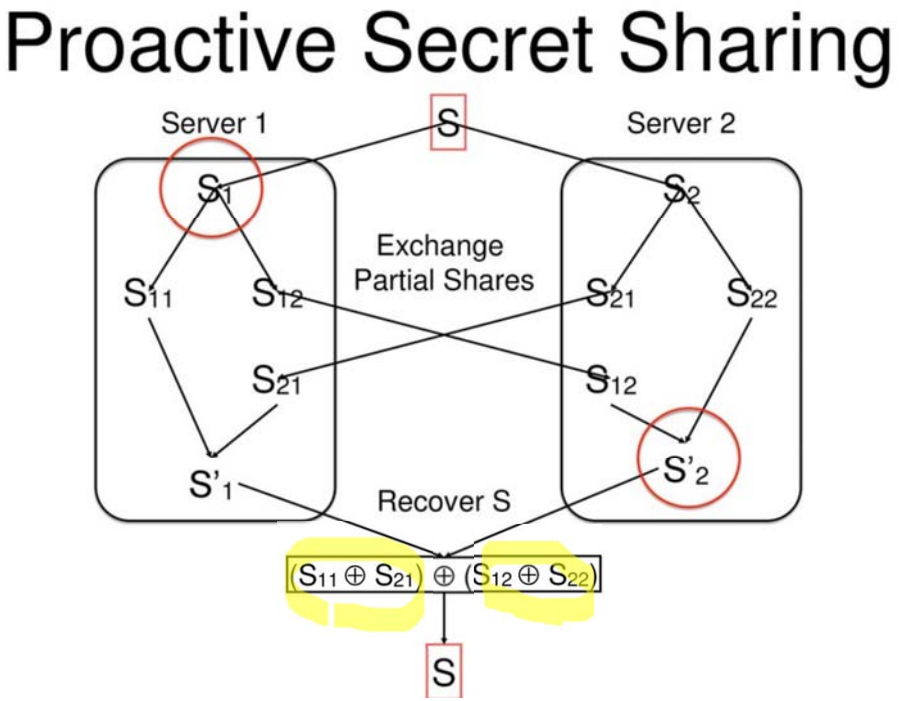
Goal: without changing the secret, periodically update shares in a way that old shares are invalidated.

출처 : Nisarg Raval, "Secret Sharing," Cornell, CS512 lecture notes, https://www.cs.cornell.edu/courses/cs513/2000SP/SecretSharing.html





출처 : Nisarg Raval, "Secret Sharing," Cornell, CS512 lecture notes, https://www.cs.cornell.edu/courses/cs513/2000SP/SecretSharing.html



출처 : Nisarg Raval, "Secret Sharing," Cornell, CS512 lecture notes, https://www.cs.cornell.edu/courses/cs513/2000SP/SecretSharing.html

Proactive secret sharing (사전 비밀 공유) : Herzberg et al (1995)

- 참가자(Participant)의 공유값(shared value)이 attacker에게 steal 또는 copy 되는 것을 방지하기 위해 제안됨
  - 일반적인 비밀 공유 기법은 dealer에 의해 한번 공유된 값이 지속적으로 사용되므로, 공격자가 합법적인 참가자의 공유값을 알게 될 경우 합법적인 참가자로 위장하여 비밀 복원 과정에 참여할 수 있음

➢ Proactive secret sharing은 참가자들에게 분배한 공유값의 주기적인 Update 및 재분배를 통해 위장과 같은 공격으로 안전하게 비밀 공유 수행

- 최초 비밀키(K)로부터 n개의 공유값 생성, 분배, 복원 과정은 Shamir의 (k,n)-threshold 방법과 동일

  복원하는 역할은 dealer가 수행, dealer는 합법적으로 인증된 자로 가정

출처 : 현승일, 신상호, 유기영, "A proactive secret sharing scheme over GF(2^8), Journal of Korea Multimedia Society, Vol. 16, no. 5, pp. 577-590, 2013

공유값의 Update 과정

Step 1 : dealer는 모든 참가자 n명에 대한 Random한 다항식 n개 생성

$$\delta_i^t(z) = \delta_{i,1}^t z^1 + \delta_{i,2}^t z^2 + \cdots + \delta_{i,k-1}^t z^{k-1} (\mathrm{mod}\ p)$$

$i(i = 1,2,\ldots,n)$는 i 번째 참가자를 의미하고, t 는 공유값의 t 번째 update를 의미

Step 2 : dealer는 생성된 n개의 다항식을 통해 $\mu_{i,j}^t = \delta_i^t(j)$ 값을 계산

j는 [0,k-1] 범위 내에서 순차적으로 계산

Step 3 : dealer는 생성된 $\mu_{i,j}^t = \delta_i^t(j)$ 값을 참가자들에게 broadcast 통해 전송하고,
참가자 i 는 dealer가 보낸 값들 중 $\mu_{i,j}^t$ 혹은 인덱스 j 가 $j = i$ 인 경우에 해당하는 $\mu_{i,j}^t$ 값을 선택하여 받음

Step 4 : 임의의 참가자 i 는 기존의 공유값 x_i^t와 dealer로부터 전송받은 $\mu_{i,j}^t$와 $\mu_{m,j}^t$ (단, m 은 $[0,i),(i,k-1]$) 들을 이용하여 아래 식에 의하여 공유값을 update

$$x_i^{t+1} = x_i^t + \sum_{m=1}^{k-1} \mu_{m,j}^t \ (\mathrm{mod}\ p)$$

출처 : 현승일, 신상호, 유기영, "A proactive secret sharing scheme over GF(2^8), Journal of Korea Multimedia Society, Vol. 16, no. 5, pp. 577-590, 2013

Random Numbers in Cryptography

출처 : 부산대학교, Prof. Howon Kim, Computer Security,

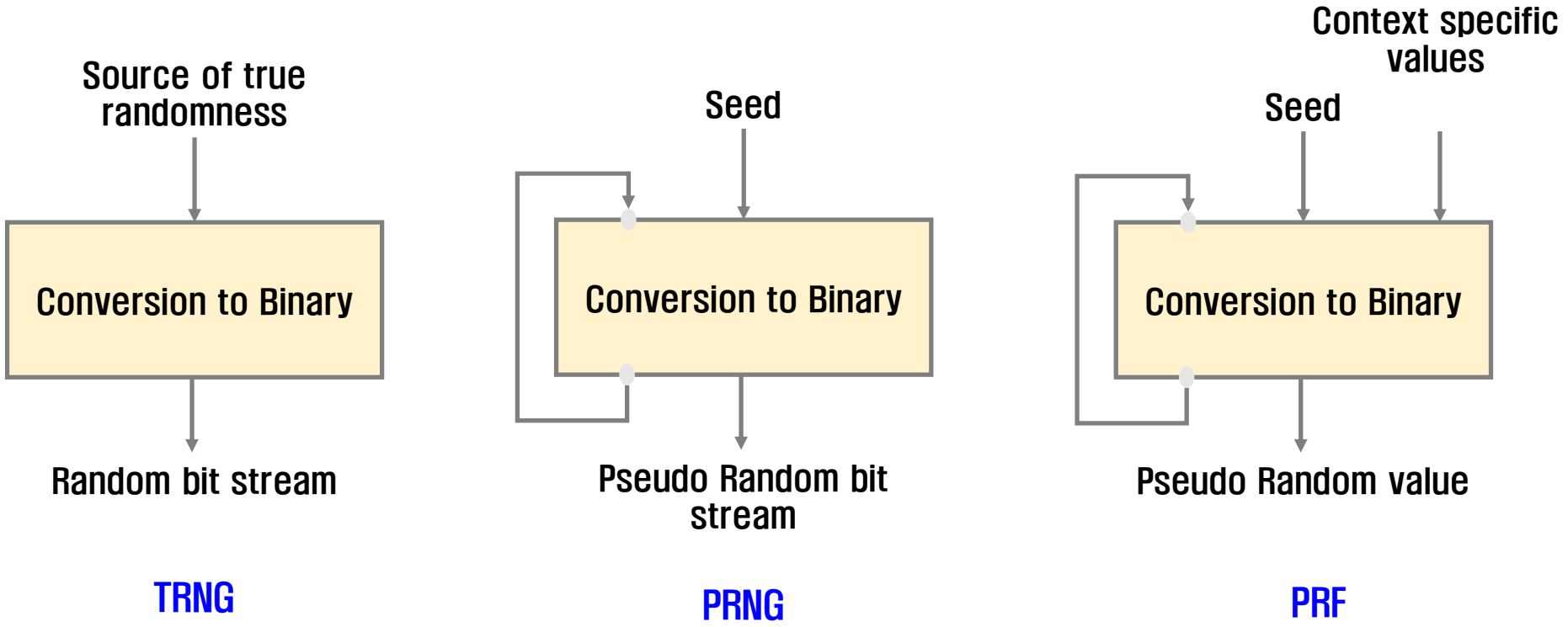
Random Numbers

- Random number : 특정한 배열 순서나 규칙을 가지지 않는 연속적인 임의의 수
- Pseudo random number : 컴퓨터에 의해 만들어지는 난수로 매우 긴 주기를 가지고 있는 숫자 열
- Use of Random Numbers
  - ✓ Key distribution and authentication schemes
  - ✓ Generation of session keys or keys for RSA
  - ✓ Generation of bit stream for stream ciphers
- Randomness
  - ✓ Uniform distribution : 수열의 비트 분포가 균일, 0과 1의 출현 빈도가 거의 동일
  - ✓ Independence : No sub-sequence can be inferred from others
- Unpredictability
  - ✓ Hard to predict next value in sequence

TRNG, PRNG, and PRF

- TRNG(True Random Number Generator)
  - ✓ 실제적으로 랜덤한 소스를 입력으로 사용
    - 키보드 입력 타이밍 패턴 및 마우스 움직임
    - 디스크의 전기적 활동, 시스템 클럭의 순간 값
- PRNG(Pseudo random number)
  - ✓ 고정값 seed를 입력받아 결정적 알고리즘을 사용하여 출력 비트열 생성
  - ✓ 제한이 없는 비트열 생성에 이용
    - 알고리즘과 seed를 알고 있는 공격자는 비트열 재생성 가능
- PRF(Pseudo Random Function)
  - ✓ 고정된 길이의 의사 난수 비트열을 생성에 사용

Random and Pseudo-Random Number Generators



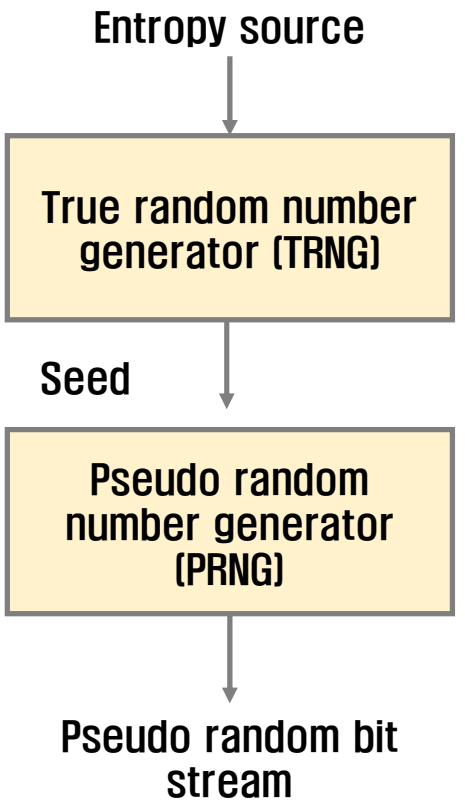
Requirements of PRNG

- Hard to determine pseudo-random stream if don't know seed (but know algorithm)
- Randomness
  - 생성된 비트 스트림이 결정적일지라도 랜덤하게 보여야 함
  - Uniformity (균일성) : 난수 또는 의사 난수 비트열의 생성에 0과 1은 거의 동일 분포
  - Scalability (확장성) : 비트열이 랜덤하면 무작위로 추출된 어떤 비트열도 랜덤해야 함
  - Consistency (일관성) : 생성기의 동작은 초기값 전반에 대하여 일관되어야 함
- Unpredictability
  - Forward unpredictability : 이전 비트들에 대한 정보가 있어도 다음 출력 비트는 예측할 수 없어야 함
  - Backward unpredictability : 생성된 어떠한 값의 정보를 통해서도 seed를 결정할 수 없어야 함

Requirements of PRNG

- Seed Requirements
  - Seed는 예측 불가능해야 함
  - Seed는 난수 또는 의사 난수이어야 함
  - TRNG에 의해 seed 생성 (SP800-90에서 권고)

Generation of Seed Input to PRNG



알고리즘 설계

- **특정 목적을 위한 알고리즘**
  - ✓ 의사난수 비트 스트림 생성을 목적으로 특정하고 유일하게 설계된 알고리즘
  - ✓ 다양한 PRNG 응용에 사용

- **기존 암호 알고리즘에 기반을 둔 알고리즘**
  - ✓ 암호 알고리즘은 입력을 난수로 만드는 효과가 있음
  - ✓ 대칭블록암호
  - ✓ 비대칭암호
  - ✓ 해쉬함수와 메시지 인증 코드

Linear congruential generator (선형 합동 생성기)

- Lehmer에 의해 처음 제안
- Parameters
  - m, modulus, $m > 0$
  - a, multiplier, $0 < a < m$
  - C, increment, $0 \leq c < m$
  - X_0, seed, $0 \leq X_0 < m$

m is prime number or power of prime number

- Generate sequence of pseudo-random numbers, $\{X_n\}$

$$X_{n+1} = (aX_n + c) \bmod m$$

- Choice of a, c and m is important
  - m should be large, prime, e.g. $2^{31}-1$
  - If $c = 0$, few good values of e.g. 7^5=16807

Linear congruential generator (선형합동생성기)

- 생성된 수의 노출 가능성
  - ✓ 공격자가 선형 합동 알고리즘 사용과 parameters를 알 경우, 알고리즘에 의해 생성된 수를 하나만 알아도 나머지 수를 모두 알 수 있게 됨
  - ✓ 선형 합동 알고리즘의 사용 여부를 안다면, 생성된 수의 순서만으로도 parameters를 알아낼 수 있음

- 해결 방법
  - ✓ 내부 시스템 클럭 사용
    - 매번 (현재의 클럭 값 mod m)을 새로운 seed로 하여 생성
    - 현재 클럭 값을 난수에 더하여 mod m의 값을 사용

Blum Blum Shub (B.B.S.) Generator

By Lenore Blum, Manuel Blum, and Michael Shub (1986)

- 어떠한 특정 목적의 알고리즘에서도 암호학적 강도를 증명하는 가장 강력하게 통용되는 수단
- 암호학적으로 안전한 의사난수 비트 생성기로 불림
  - ✓ CSPRNG (*Cryptographically* Secure Pseudo Random Bit Generator)
- Iterative equation에서 LSB(least significant bit) 사용
- Unpredictable given any run of bits
- Slow, since very large numbers must be used
- Two slow for cipher use, good for key generation
- n의 소인수 분해 문제에 대한 어려움에 기반
  - ✓ n이 주어졌을 때, n의 두 소수 인수 p와 q를 알아야 함

Blum Blum Shub Generator

- It takes the form:

$$x_i = x_{i-1}^2 \bmod M$$

where $M = pq$ is the product of two large primes

- At each step, the output is derived from x_i
- The output is commonly either the bit parity of x_i or one or more of least significant bits of x_i

- Parameters
  - ✓ p, q : large prime numbers such that $p \equiv q \equiv 3 \pmod 4$
  - ✓ $n = pq$ that
  - ✓ s, random number relatively prime to n

    gcd(n, s)=1인 s 선정

- Generate sequence of bits, B_i:

$$x_0 = s^2 \bmod n$$

for $i = 1$ to ∞

$$x_i = (x_{i-1})^2 \bmod n$$

$$B_i = x_i \bmod 2$$

Example Operation of Blum Blum Shub Generator

- $n = 192649 = p \times q = 383 \times 503, s = 101355$ $(gcd(n,k) = 1)$

  $(p = 383 \bmod 4 = 3, q = 503 \bmod 4 = 3)$

i	X_i	B_i	i	X_i	B_i
0	20749		11	137922	0
1	143135	1	12	123175	1
2	177671	1	13	8630	0
3	97048	0	14	114386	0
4	89992	0	15	14863	1
5	174051	1	16	133015	1
6	80649	1	17	106065	1
7	45663	1	18	45870	0
8	69442	0	19	137171	1
9	186894	0	20	48060	0
10	177046	0			

$$x_0 = s^2 \bmod n$$

$$x_i = (x_{i-1})^2 \bmod n$$

$$B_i = x_i \bmod 2$$

Blum Blum Shub Generator (Note)

1. The seed x_0 should be an integer that is co-prime with M, ie. p and q are not factors of x_0 and not 0 or 1.
2. The two primes p and q should both be congruent to 3 (mod 4) and $\gcd(\varphi(p-1), \varphi(q-1))$ should be small,

   where φ is the Euler function which is, in this case, number of integers k in the range $1 \le k \le n$ for which the gcd of k is $gcd(n,k) = 1$

- Remark : it is possible to calculate x_k any directly using:

$$x_k = (x_0^{2^k \bmod \lambda(M)}) \bmod M$$

where is the Carmichael function:

$$\lambda(M) = \lambda(pq) = lcm(p-1, q-1)$$

Let $p = 11, q = 19$ and the seed $s = 3$

We can expect a large cycle because

$$gcd(\varphi(p-1), \varphi(q-1)) = 2$$

The generator starts to evaluate x_1 using $x_0 = s$

And creates the sequence

$$x_1, x_2, x_3, \ldots, x_6 = 9, 81, 82, 36, 42, 92$$

The following table shows the possible outputs in bits for different bit selection methods used to determine the output

Even parity bit	Odd parity bit	Least significant bit
011010	100101	110000

In number theory, **Euler's theorem** (also known as the **Fermat–Euler theorem** or **Euler's totient theorem**) states that if *n* and *a* are coprime positive integers, then *a* raised to the power of the totient of *n* is congruent to one, modulo *n*, or:

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

where $\varphi(n)$ is Euler's totient function. In 1736, Leonhard Euler

- gcd : greatest common divisor
- lcm : least common multiple

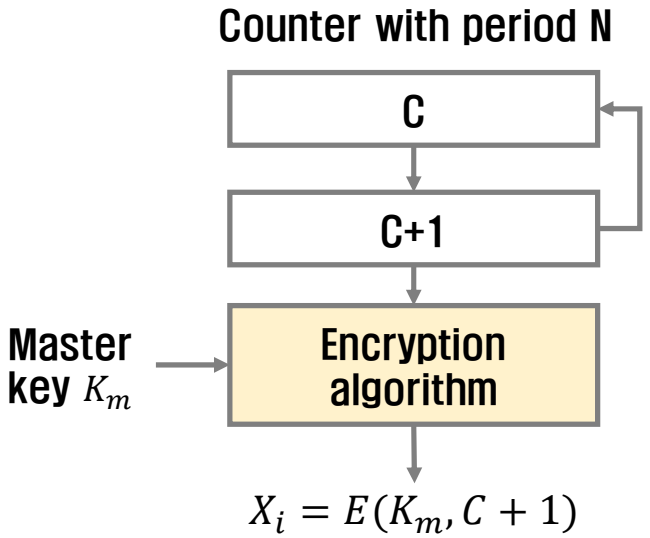
Using Block Ciphers as PRNGs

- Use symmetric block ciphers (e.g. AES, DES) to produce pseudo-random bits
- Used for creating session keys from master key K_m

- Counter Mode → (강의자료 3장 6 Block cipher modes – CTR 참조)

$$X_i = E(K_m, C + 1)$$

  - ✓ A counter with period N provides input to the encryption logic
  - ✓ Example, if 56-bit DES key are to be produced, then a counter with period 2^{56} can be used.
  - ✓ After each key is produced, the counter is incremented by one. Thus, the pseudorandom numbers produced.



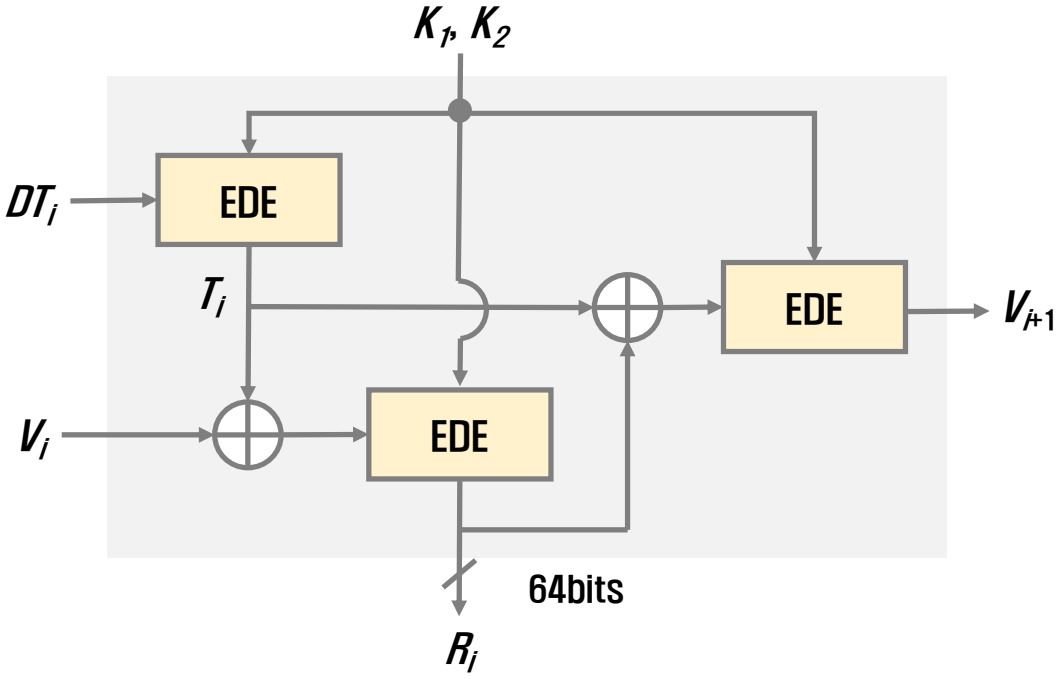
- Other : OFB (Output Feedback) Mode

ANSI X9.17 PRNG

- Cryptographically secure PRNG using Triple DES
- Input : Two pseudorandom inputs drive the generator
  - ✓ 1) a 64-bit representation of the current date and time - Updated on each number generation
  - ✓ 2) a 64-bit seed value - Initialized to some arbitrary value and is updated during the generation process
- Keys : Make use of three triple DES encryption module
  - ✓ All three make use of the same pair of 56-bit DES keys, K_1 and K_2
- Output
  - ✓ A 64-bit pseudorandom number, R_i
  - ✓ A 64-bit seed value, V_{i+1}

ANSI X9.17 PRNG

- Cryptographically secure PRNG using Triple DES



DT_i: Current date & time
V_i: seed value
R_i: Pseudorandom number (output)
K_1, K_2: DES keys (in this Ex, Triple DES keys)

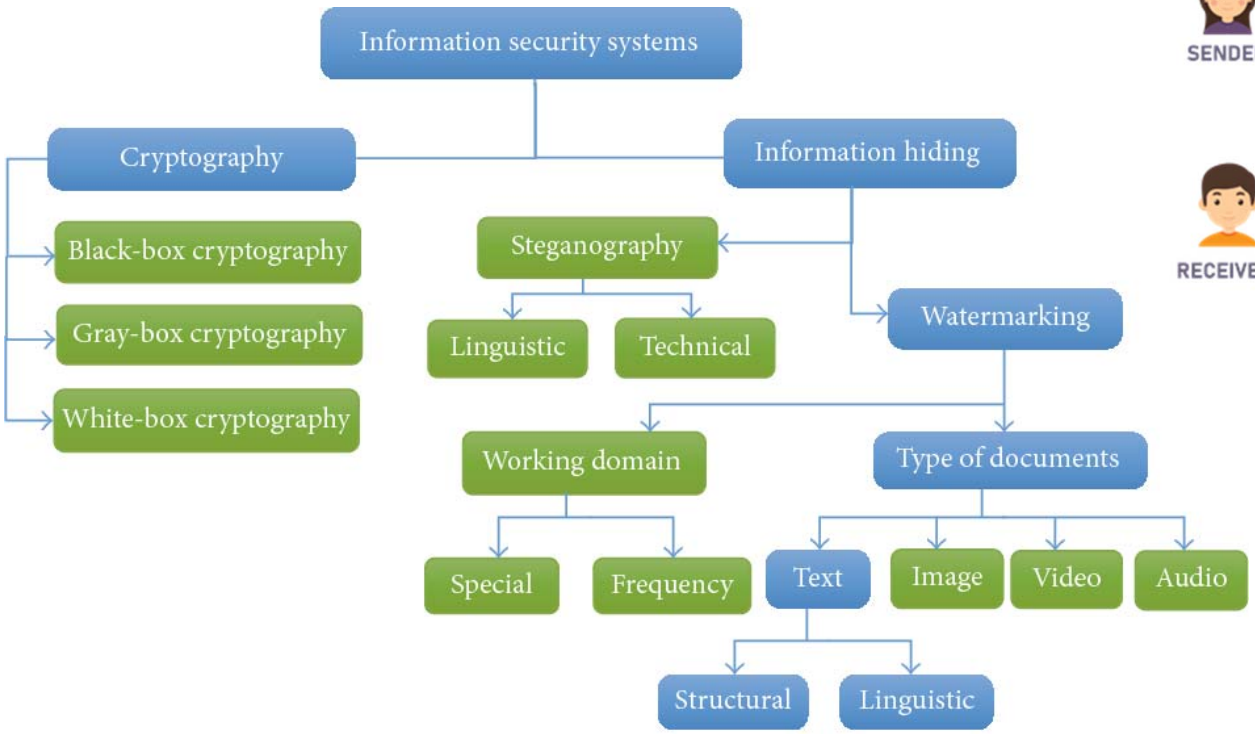
1. Compute $T_i = DES_k(DT_i)$
2. Output $R_i = DES_k(T_i \text{ XOR } V_i)$

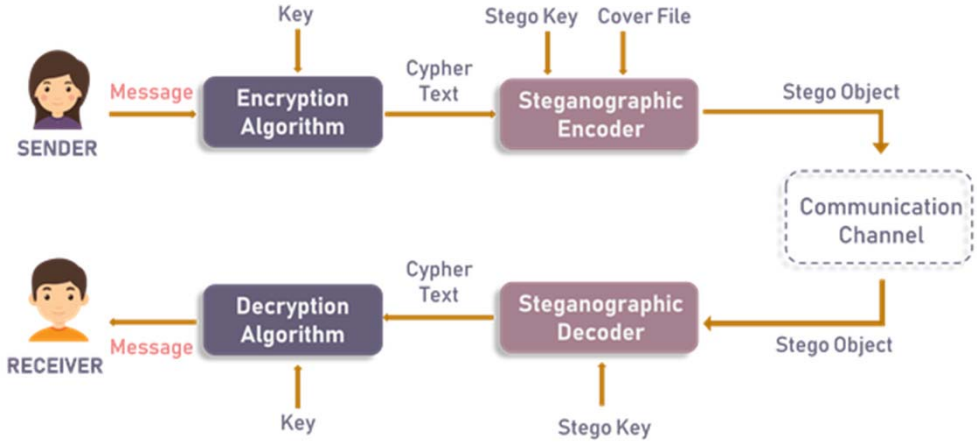
Update the seed
to $V_{i+1} = DES_k(R_i \text{ XOR } T_i)$

EDE – Triple DES (Enc–Dec–Enc.)

x9.17 will be improved by using AES

Information security





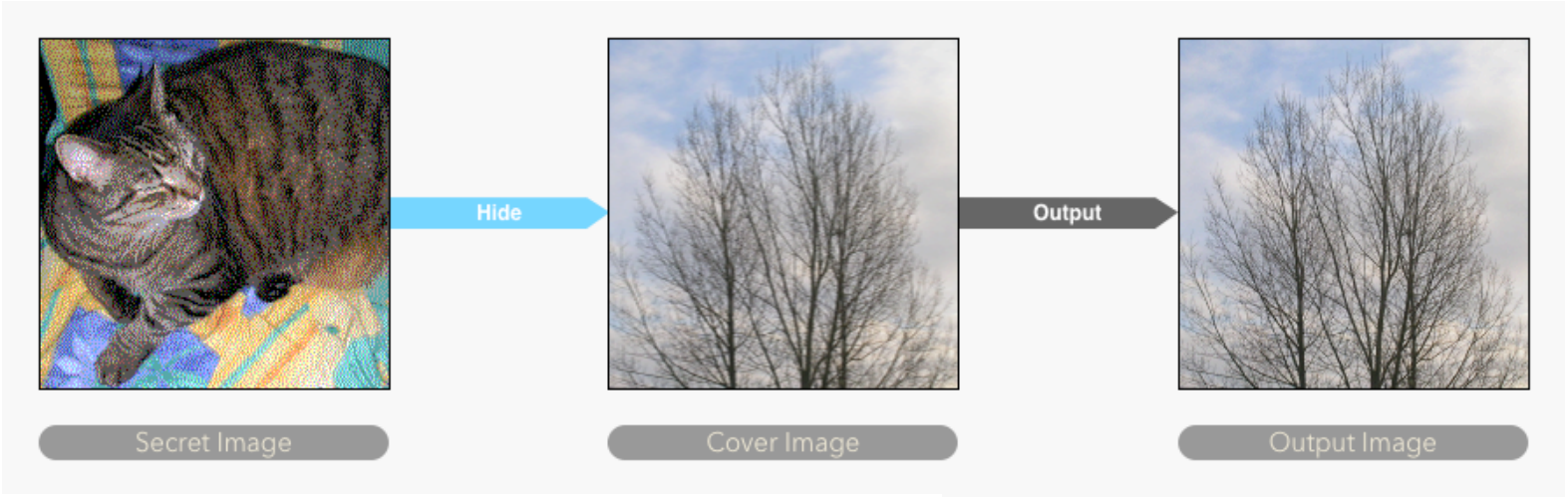
General process of steganography

Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection

A digital watermark is a kind of marker covertly embedded in a noise-tolerant signal such as audio, video or image data.

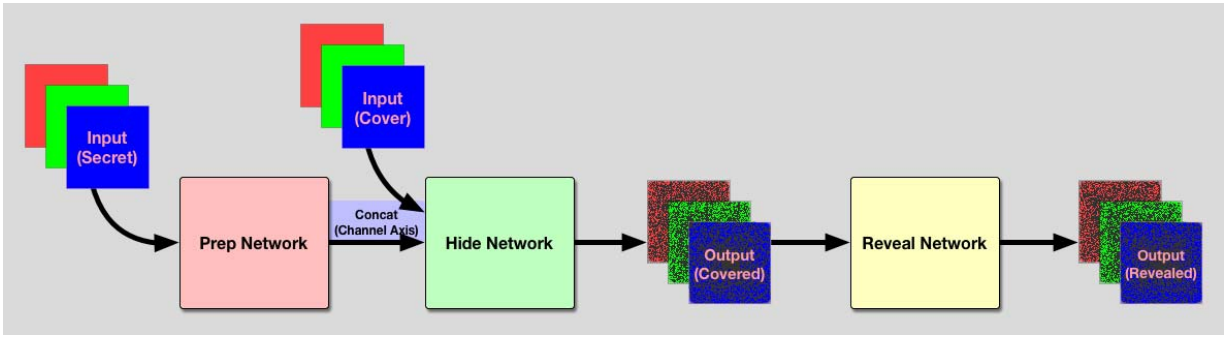
3) Information Hiding Introduction

Example of Image Steganography : NIPS 2017 Paper Hiding Images in Plain Sight: Deep Steganography.



Secret image를 Cover image에 은닉

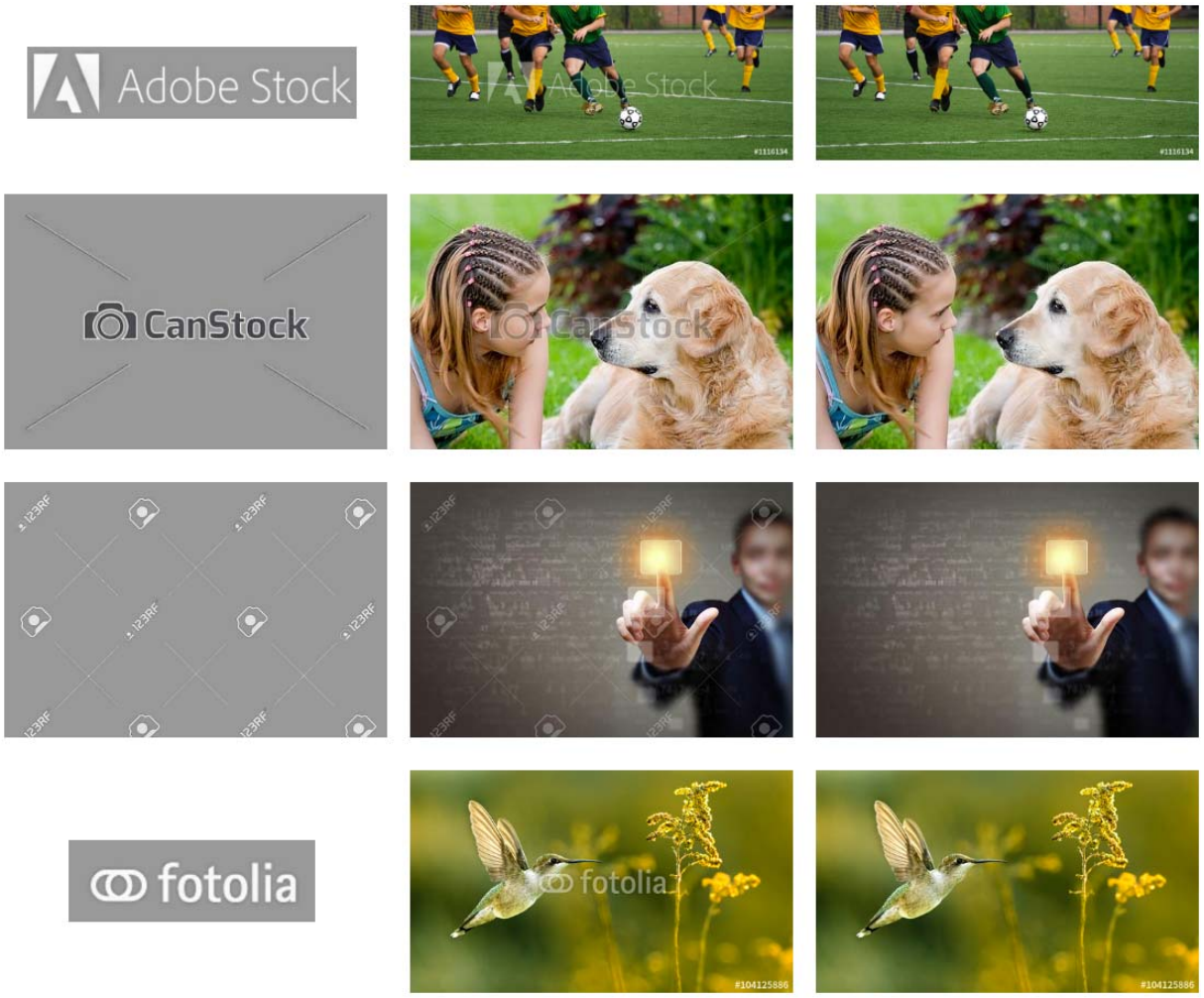
비밀통신, 데이터보호, 컨텐츠 인증, 포렌식 등 가능



Network Architecture

3) Information Hiding Introduction

Example of Image Watermarking : Google AI blog, "Making Visible Watermarks More Effective"



Visible / Invisible watermarking

Copyright protection,
콘텐츠 위변조 판별

Audio
Image
Video
3D Graphics model
Text Document
DNA sequence
GIS Vector map

For all digital data