

<https://youtu.be/EsLIm761q1M>

블록체인의 난수 생성 취약점 해결을 위한

딥러닝 기반의 온-체인 난수 생성기 DL-ORNG

ZANMANG Boogies

김현지 임세진 윤세영

목차

1

블록체인에서의 난수 생성기

문제 정의, 기존 해결 방안 및 그에 대한 문제점

2

DL-ORNG

필요성, 시스템 세부 사항 및 성능 분석

3

구현 및 적용 가능성

온-체인에서의 실현 가능성 확인

4

결론 및 기대 효과

블록체인 관점

블록체인 상에서의 난수

난수

난수의 조건

비편향성, 예측 불가능성

난수 생성기의 종류

진짜 난수 생성기 (TRNG) : 물리적 현상으로부터 얻음

유사 난수 생성기 (PRNG) : 알고리즘에 의해 랜덤과 유사하게 생성
(결정론적, 패턴 존재)

블록체인에서의 난수

난수의 중요성

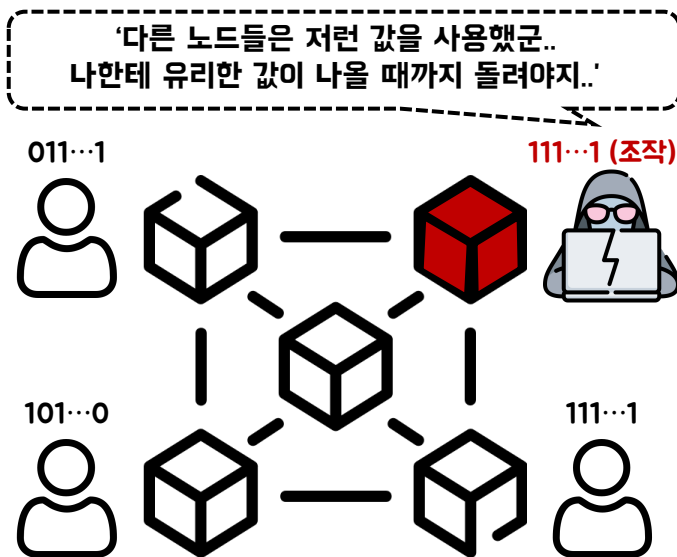
블록체인에서 난수는 매우 중요

지갑 및 거래에 사용되는 키 생성, 스마트 컨트랙트에 사용
→ 블록체인의 안전성에 영향을 줄 수 있는 요소

따라서, 높은 난수성 및 신뢰성을 가진 난수가 필요

블록체인의 특성 상 난수 생성이 어려움

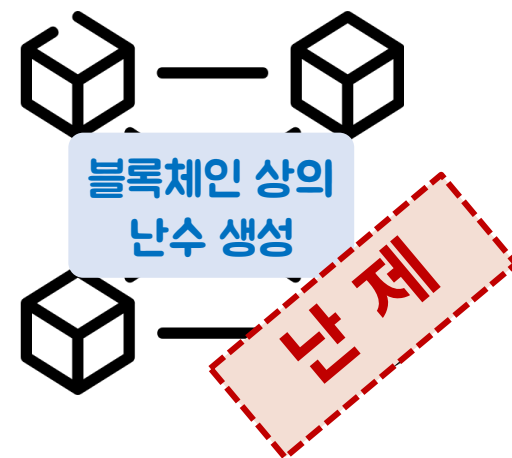
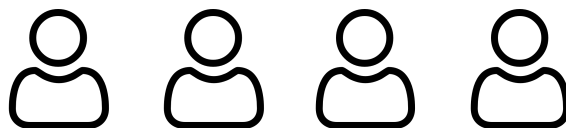
- 1 모든 내역이 공개되므로 악의적인 노드가 그 값들을 보고 유리하게 조작 가능
- 2 조작 방지를 위해 여러 노드의 값을 취합하는 방식 → 결국 마지막에 값을 보내는 노드가 조작 가능
- 3 체인 내부 값 (ex:블록 해시)는 공개 및 결정된 값 → 이를 사용할 경우 난수의 조건인 예측 불가능성 만족 불가
- 4 즉, 블록체인의 특성 (무결성, 투명성)으로 인해 난수 생성 시 문제 발생 → 예측 및 조작이 불가능한 난수 생성은 난제로 분류



+

블록 해시 1 : 0x1234 → 생성된 난수 : 101011
블록 해시 2 : 0x5678 → 생성된 난수 : 111111

아하..! 저 값을 넣으면 저런 난수가 나오는구나



블록체인 상에서의 **난수 생성 조건**

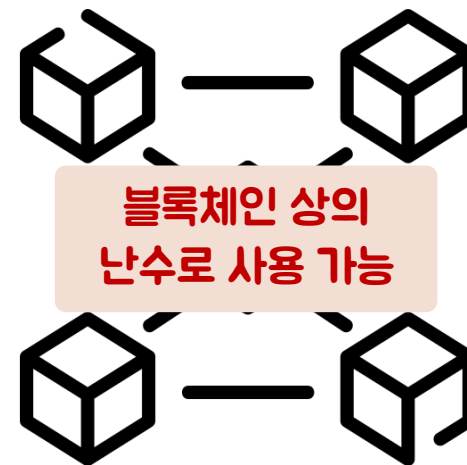
비편향성, 예측 불가능성

일반적인 난수의 조건

+

조작 불가능성

블록체인을 위한 난수의 조건



현재 사용 중인 해결 방안



ETHEREUM

난수생성기 미제공



Commit Reveal

VRF (Verifiable Random Function)

DAPP에서의 자체적 난수 생성 기능

각 플랫폼 및 DAPP에서
다양한 난수 생성 기능 제안

현재 사용 중인 해결 방안 분석

Commit Reveal

- 1 온-체인
- 2 각 노드가 난수 생성을 위해 어떤 값을 전송할 때 암호화해서 보냄
→ 마지막 노드가 조작할 수 없음
- 3 시간 및 메모리 비용이 큼 (오라클 방식보다 큰 비용)
- 4 악의적인 노드 존재 시, 난수 생성 어려움

VRF (Verifiable Random Function)

- 1 오프-체인 + 온-체인 (오라클 사용)
- 2 난수 생성 (오프-체인) → 오라클 → 검증 (온-체인)
- 3 위·변조 발생 가능성 존재 및 해당 값에 대한 신뢰가 어려움
→ 온-체인에서의 검증 과정이 필요
- 4 따라서, 추가적인 시간 및 비용이 요구됨



악의적인 노드에 대응 불가 or 무결성 및 신뢰성 문제 발생

이를 해결하기 위해 비용 및 효율성 문제 발생

*이외에도 DAPP에서 자체 제공하는 난수 생성 기능의 난수성 문제도 존재

딥러닝 기반의 온-체인 난수 생성기의 필요성

비편향성, 예측 불가능성, 조작 불가능성

블록체인을 위한 난수의 조건

+

현재 해결 방안들의 단점 보완

악의적 노드, 무결성 및 신뢰성, 비용 등의 문제 해결



블록체인 상에서의 난수 생성이라는 난제 해결을 위한

**딥러닝 기반의 온-체인 난수 생성기
(DL-ORNG)**

Deep Learning based On-chain RNG

딥러닝 기반의 온-체인 난수 생성기의 필요성

DL-ORNG

기존 방법들의 한계점 보완



온-체인

오프-체인 방식에서 필연적으로 발생하는 **오라클 문제 해결** 및 **비효율성 감소**

딥러닝 난수생성기

높은 난수성, **빠른** 속도, **경량화** 된 모델 (적은 용량)

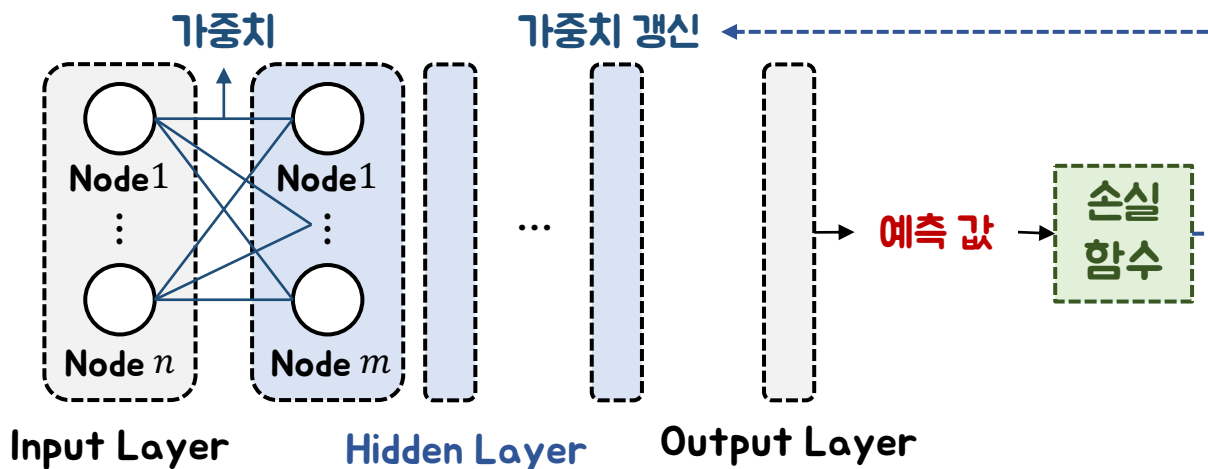
BLS 서명 적용

신뢰성 확보, **악의적 노드 문제 해결**

DL-ORNG를 위한 배경 지식 (딥러닝)

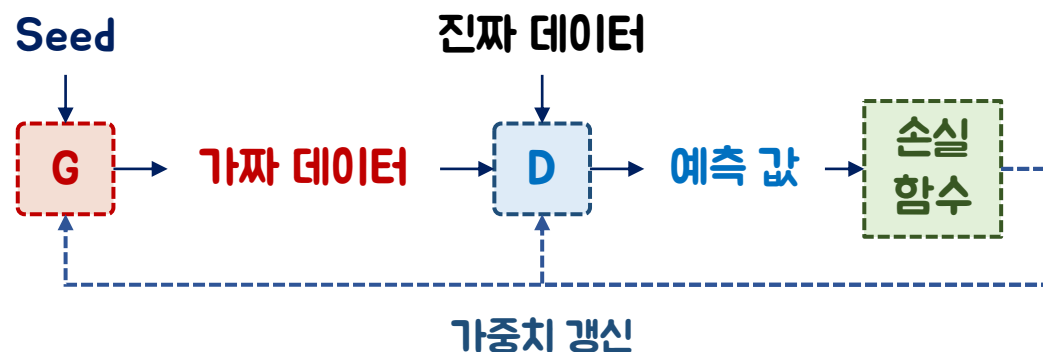
딥러닝

- 1 데이터에 대한 **특징을 학습**하여 다양한 예측 작업 가능
- 2 여러 개의 노드로 구성된 레이어가 쌓여서 구성 (가중치로 연결)
- 3 데이터는 딥러닝 모델에 입력, 모델은 그에 대한 **예측 값**을 출력
→ **예측 값과 실제 정답 간의 차이**를 계산 (손실)
- 4 손실을 줄이기 위해 모델의 **가중치가 갱신**되며 학습

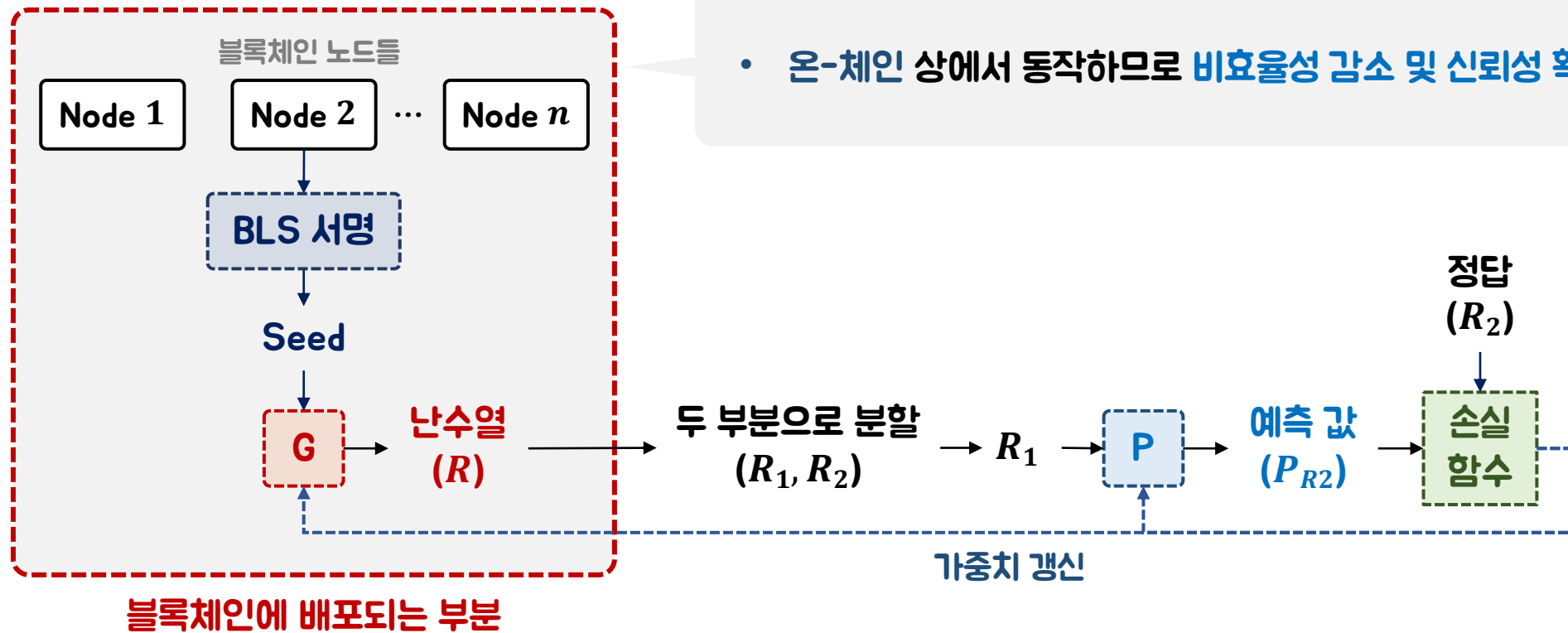


Generative Adversarial Network (GAN)

- 1 데이터를 **생성**해내는 딥러닝 모델
- 2 Generator 모델(G)과 Discriminator 모델 (D)이 결합
→ **서로 경쟁하며 학습**
- 3 Step 1. **G** : Seed로 부터 진짜 같은 가짜 데이터를 생성
Step 2. **D** : G가 생성한 가짜를 진짜 데이터와 구별
Step 3. **G** : D가 구별해내는지 보고 더 진짜처럼 만듦



DL-ORNG : 시스템 개요

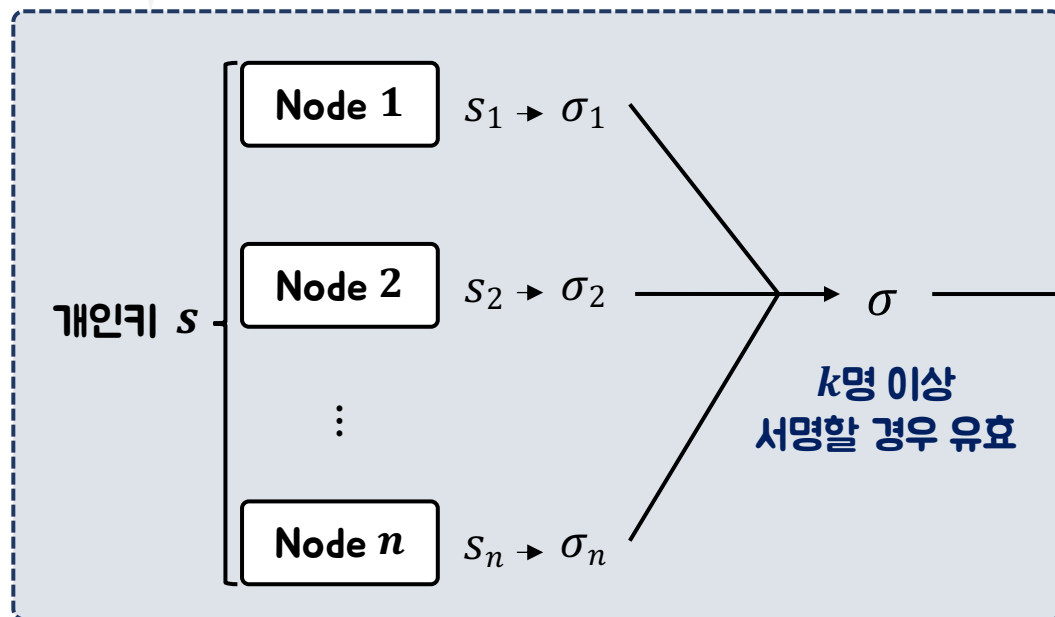


- BLS 서명 통해 온-체인 상에서 공정한 seed 생성
- 각 노드에는 학습이 완료된 G만 배포되므로 많은 메모리 및 연산 X
- 온-체인 상에서 동작하므로 비효율성 감소 및 신뢰성 확보

DL-ORNG : 난수 생성

BLS 서명

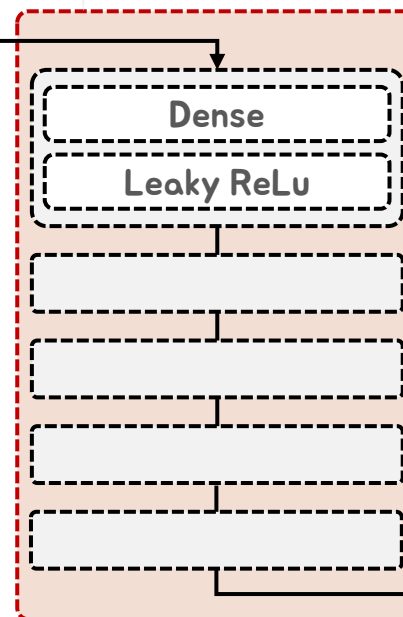
- 악의적 노드가 존재하더라도 **공정하게 seed 생성 가능**
→ **신뢰성** 확보
- 학습 시에는 BLS 서명 대신 Python의 난수 생성 함수 사용
→ 학습이 잘 이루어지는 랜덤 분포 사용 위해



BLS 서명

G

- RNN등과 같이 많은 연산이 필요한 레이어 사용 X
→ **저전력** 노드에서도 동작하도록 **경량화**



G

R

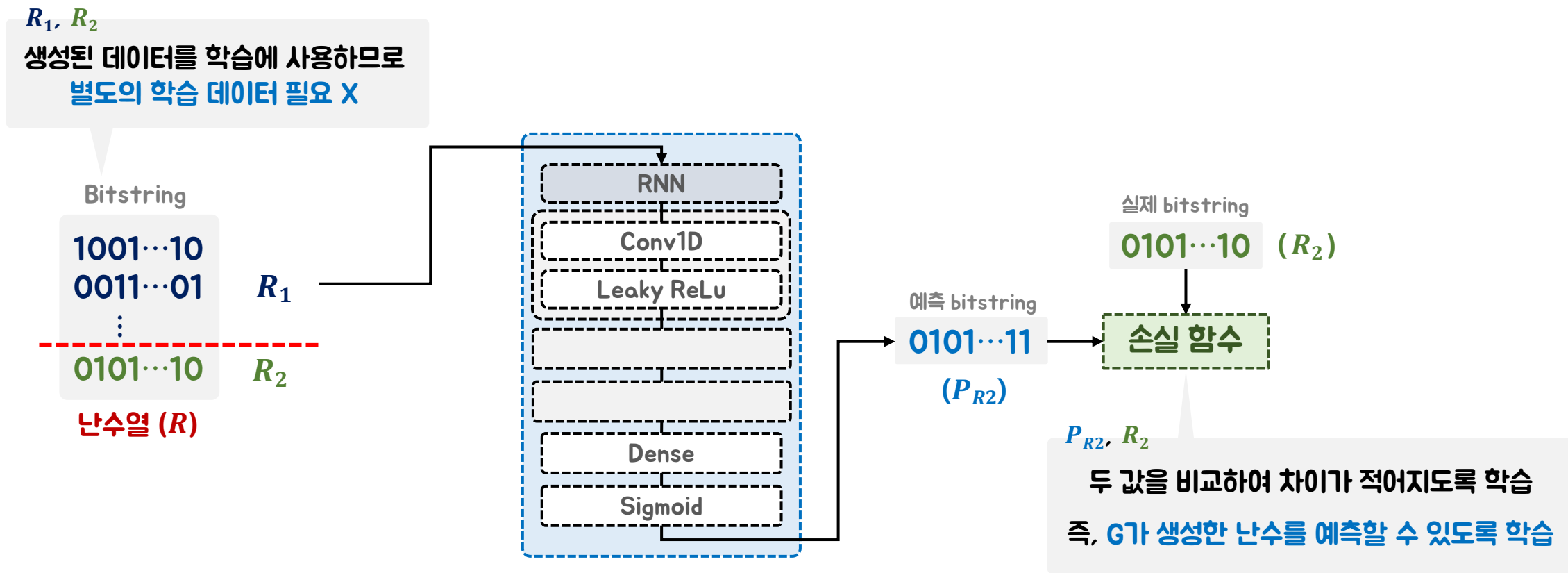
- **109920000-bit** 까지 난수성 확보

Bitstring

1001...10
0011...01
⋮
0101...10

난수열 (R)

DL-ORNG : 난수 예측



DL-ORNG : 블록체인 노드로의 배포



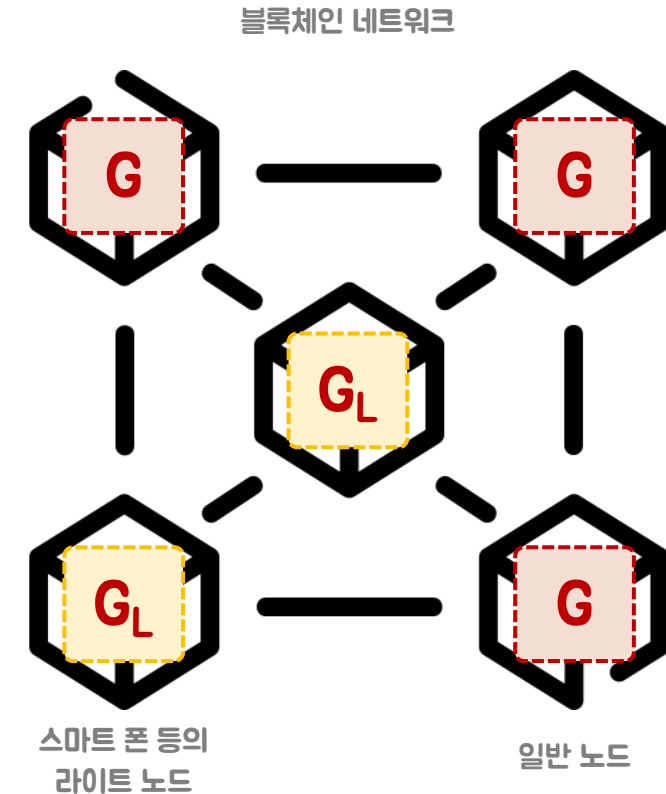
일반 딥러닝 모델



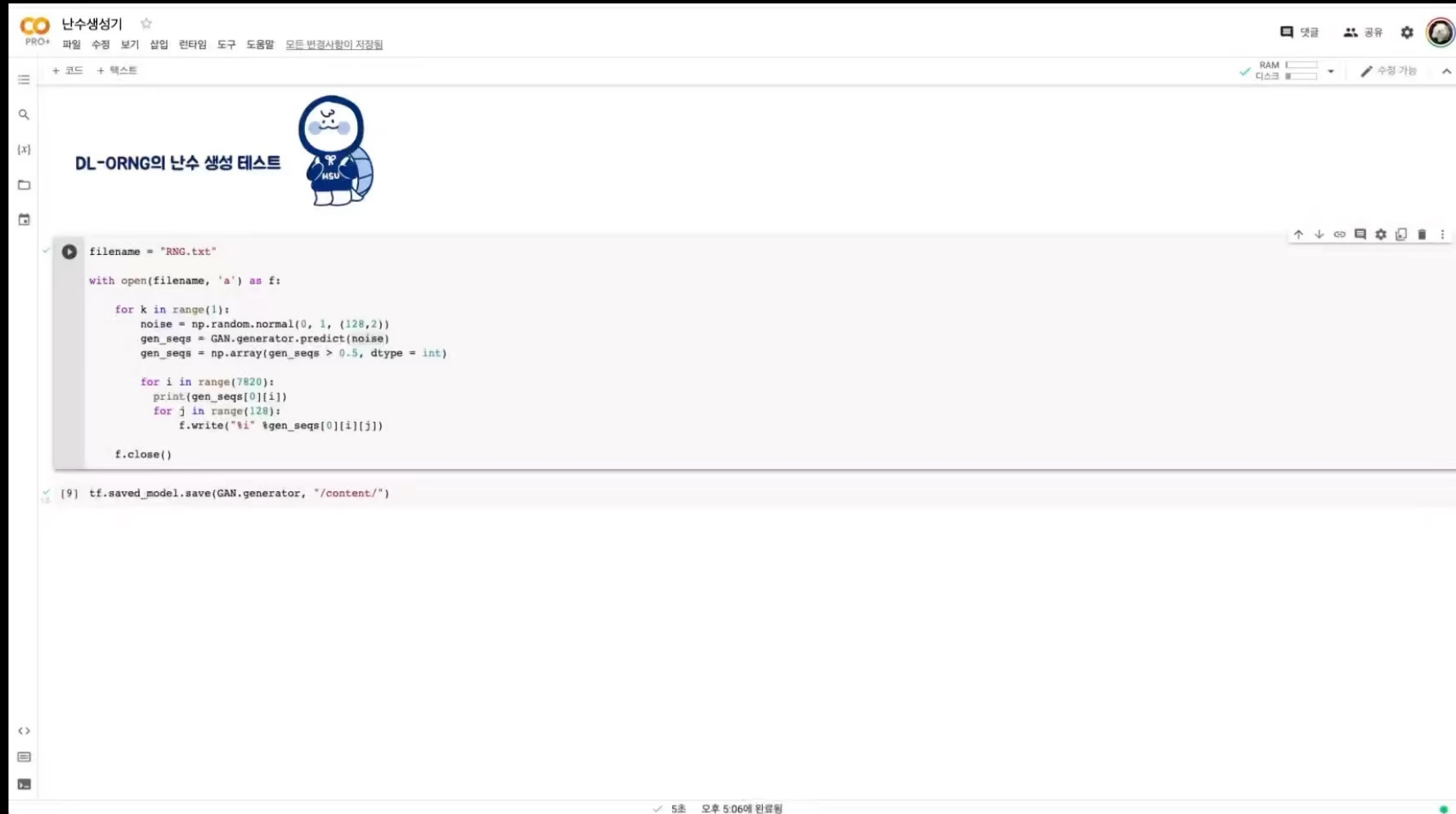
저전력 딥러닝 모델

(모바일 및 임베디드 장치 (ex: 스마트폰) 를 위해 일반 모델을 변환)

- 노드에 **학습이 완료된** 딥러닝 모델을 **배포**하여 난수 생성
- **난수성이 충분히 확보된 상태**로 배포되므로 **추가 배포 필요 X**
- 만약 다시 학습 시키고 싶다면?
딥러닝 서버에서 새롭게 학습시킨 후 노드에 다시 배포



DL-ORNG : 시연 영상



The screenshot shows a web-based code editor interface. At the top, there's a header with the text "난수생성기" (Random Number Generator) and "PRO+". Below the header, there's a navigation bar with icons for file, search, and other functions. The main area displays a Python script titled "DL-ORNG의 난수 생성 테스트" (DL-ORNG Random Number Generation Test). The script is as follows:

```
filename = "RNG.txt"

with open(filename, 'a') as f:

    for k in range(1):
        noise = np.random.normal(0, 1, (128,2))
        gen_seqs = GAN.generator.predict(noise)
        gen_seqs = np.array(gen_seqs > 0.5, dtype = int)

        for i in range(7820):
            print(gen_seqs[0][i])
            for j in range(128):
                f.write("%i" %gen_seqs[0][i][j])

    f.close()
```

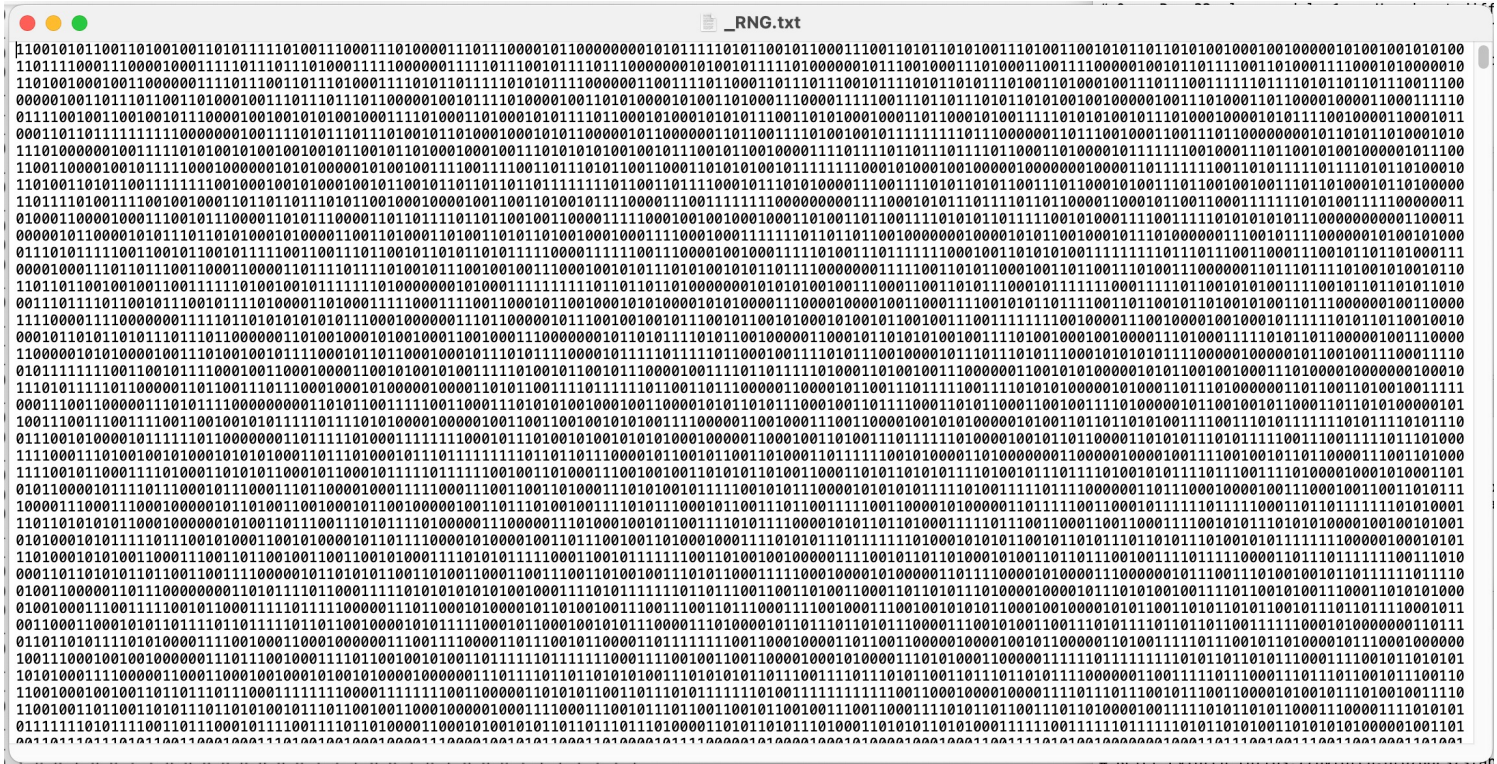
Below the script, there's a terminal output showing the command:

```
[9] tf.saved_model.save(GAN.generator, "/content/")
```

The interface also includes a sidebar on the left with icons for file, search, and other functions, and a status bar at the bottom showing the time and date.

DL-ORNG : 성능 분석

생성된 난수 시각화

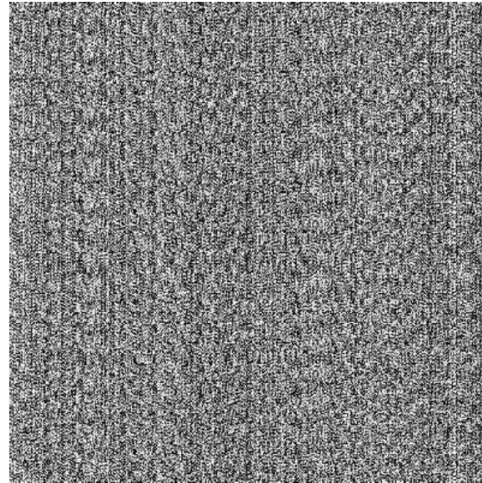


실제 생성된 난수 (Bitstring)

DL-ORNG : 성능 분석

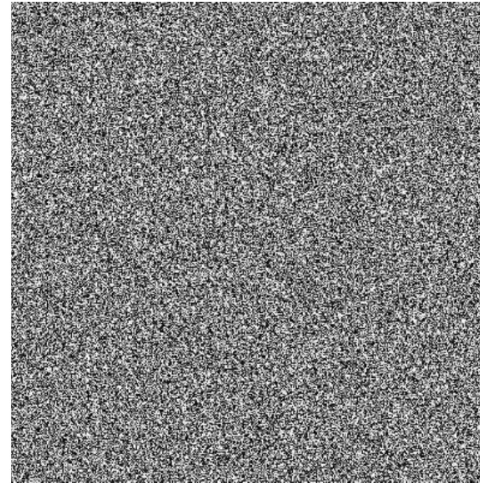
생성된 난수 시각화

학습 전



특정 패턴 존재
세로 줄무늬

학습 후



패턴 없음
고르게 분포

DL-ORNG : 성능 분석

NIST SP 800-22 통계적 테스트 결과

- 개별 테스트
: Frequency, Runs 등과 같이 각 항목
- 테스트 인스턴스
: 모든 테스트는 여러 번 반복 시행 (각 시행 = 테스트 인스턴스)

SP 800-22 결과 화면

generator is <data/1.pi> 테스트 인스턴스 (성공/전체)												
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
0	2	0	0	3	0	2	0	1	2	0.213309	10/10	Frequency 개별 테스트
0	1	0	3	0	1	1	1	2	1	0.534146	10/10	BlockFrequency
0	1	1	2	1	0	1	2	2	0	0.739918	10/10	CumulativeSums
1	1	1	1	0	1	2	2	1	0	0.911413	10/10	CumulativeSums
1	0	0	5	0	0	1	1	0	2	0.008879	10/10	Runs
1	1	0	1	2	1	0	1	2	1	0.911413	10/10	LongestRun
0	1	1	1	1	2	1	0	0	3	0.534146	10/10	Rank
2	3	1	2	1	1	0	0	0	0	0.350485	10/10	FFT
1	1	1	0	1	3	0	1	1	1	0.739918	10/10	NonOverlappingTemplate
0	0	1	1	1	1	4	1	1	0	0.213309	10/10	OverlappingTemplate
1	0	0	0	1	2	1	0	3	2	0.350485	9/10	Universal
0	1	1	1	0	3	0	0	4	0	0.035174	10/10	ApproximateEntropy
0	0	0	2	0	1	2	0	0	2	----	7/7	RandomExcursions
0	0	0	1	1	0	0	2	2	1	----	7/7	RandomExcursionsVariant
2	0	1	3	0	0	1	0	1	2	0.350485	10/10	Serial
1	1	1	0	1	2	0	0	1	3	0.534146	10/10	Serial
0	1	0	3	2	0	2	2	0	0	0.213309	10/10	LinearComplexity

난수성을 테스트 하기 위한 다양한 항목 존재
 빈도수, 특정 패턴 반복 여부 등

$F_I/\%$: 실패한 테스트 인스턴스 비율
 $F\%$: 실패한 개별 테스트 수의 비율

	$F_I/\%$	$F\%$
학습 전	98.8	98.9
학습 후	1.09	0.00

*10번의 테스트에 대한 평균이며, 자세한 분석 결과는 [1]에 언급

- 실패한 테스트 인스턴스는 평균 1.09개
- 모든 개별 테스트 통과
- 대부분의 PRNG보다 나은 성능 달성 [2]



높은 난수성 확보
 학습 후에는 대부분의 테스트 통과

DL-ORNG : 성능 분석

블록체인 네트워크에 과부하를 주지 않는 경량 모델

처리량	대상 기기	모델 크기
1.0 GB/s	임베디드 기기	166KB

- 다른 저전력 난수 생성기 (MPCG)에 비해 **6.25배 빠른 속도**
- 모바일 기기에서의 **사진 한 장보다 작은 용량**

DL-ORNG : 블록체인에서의 적용 가능성

- 1 딥러닝 기반의 난수 생성기 구현 (완료) [1]
- 2 해당 모델을 저전력 디바이스를 위한 모델로 변환 (완료) [1]
- 3 임베디드 프로세서 (Edge TPU) 상에서 동작 가능 확인 (완료) [1]
- 4 노드에서 딥러닝 모델을 실행할 수 있다면 실현 가능한 방법 (가능)
→ [3]에 따르면 각 노드는 본인의 로컬 모델을 가질 수 있으며, 학습 및 추론 가능 (이외에도 다수의 연구 존재)

» 따라서, 블록체인 네트워크에
실제로 적용 가능한 방안



DL-ORNG : 블록체인에서의 기대 효과

- 블록체인에서 **난수가 필요한 부분에 사용 가능**
 - 합의 알고리즘 (검증자 무작위 선출)
 - 스마트 컨트랙트
 - 키 생성
- 블록체인의 **안전성과 관련** 있는 부분들이므로 **신뢰성 확보 가능**



- **경량 디러닝 모델**
 - 저전력 기기에서 동작 가능
 - 적은 메모리 차지
- 따라서 **연산 및 메모리 과부하 X**
 - 블록체인의 확장성 저하 X

- BLS 서명을 통한 **공정하고 신뢰성 있는 난수 생성**
- 높은 난수성으로 인한 **안전성 확보**
- 오프-체인 사용 X
 - **오라클 문제 해결** (조작 방지 → 신뢰성 확보)
 - 비용 절감

결론

- **Defi 난제 해결 가능**
 - 스마트 컨트랙트 취약성 및 오라클 문제
- 전반적인 블록체인의 **안전성 및 확장성 보장 가능**

딥러닝 기반의 온-체인 난수생성기

**발표 들어주셔서
감사합니다.**

ZANMANG Boogies 