https://doi.org/10.33778/kcsa.2020.20.2.069

비정형 빅데이터를 이용한 난수생성용 블록체인 오라클★

정 승 욱*

요 약

블록체인 2.0은 프로그래밍 가능한 스마트계약을 사용하여 다양한 DApp(Distributed Application) 개발을 지원한다. 하지만 스마트계약이 동작하는 환경은 블록 높이, 블록 해쉬, 트랜잭션 해쉬 등 결정된 데이터만 접근할 수 있어서 블록체인 기반 복권, 배팅, 키 생성 등 난수를 필요로 하는 응용은 블록체인 외부에서 데이터를 가져 올 수 있는 오라클 서비스를 이용해야 한다. 본 논문에서는 난수 생성 오라클 서비스를 개발하였다. 또한 난수 생성을 위해 비정형 빅데이터를 entropy source로 사용하였다. 이렇게 생성된 난수를 NIST SP800-22 난수 테스트하여 난수로 사용할 수 있음을 확인하였다. 또한, 물리적 세계를 entropy source로 사용하는 기존의 진성난수 생성기에 비해서 비용측면에서 유리함을 설명한다.

Blockchain Oracle for Random Number Generator using Irregular Big Data

Seung Wook Jung*

ABSTRACT

Blockchain 2.0 supports programmable smart contract for the various distributed application. However, the environment of running smart contract is limited in the blockchain, so the smart contract only get the deterministic information, such as block height, block hash, and so on. Therefore, some applications, which requires random information, such as lottery or batting, should use oracle service that supply the information outside of blockchain. This paper develops a random number generator oracle service. The random number generator oracle service use irregular big data as entropy source. This paper tests the randomness of bits sequence generated from oracle service using NIST SP800-22. This paper also describes the advantages of irregular big data in our model in perspective of cost comparing hardware entropy source.

Key words : Blockchain, Random Number, Blockchain Oracle, Big Data

접수일(2020년 4월 23일), 1차 수정일(2020년 6월 16일), 게재확정일(2020년 6월 26일)

★ 본 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2019-0-00411, 블록체인의 개인 콘텐츠 추적과 완전소멸 수정을 위한 잊힐 권리 문제 해결)

* 건양대학교/사이버보안공학과(corresponding author)

1. 서 론

퍼블릭 블록체인은 신뢰할 수 있는 제3자 (Trust Third Party : TTP) 없이 네트워크에 참여한 사람들에게 신뢰를 제공해주는 신뢰머신으로 많은 주목을 받고 있다. 따라서 다양한 분야[1][2][3]에서 블록체인을 이용하고 있다.

이더리움 같은 블록체인 2.0은 각 노드가 동일한 상태, 즉 월드 상태(world state)를 가지는 프로그래밍 가능한 월드 컴퓨터이다[4]. 블록체인 월드 컴퓨터에서 동작하는 프로그램을 주로 스마트계약[5]이라고 부른다. 본 논문에서는 스마트계약을 법률적인 계약을 의미하지 않고 블록체인에서 동작하는 프로그램을 의미한다.

하지만 이더리움의 경우 Ethereum Virtual Machine (EVM)[4]에서 동작하는 스마트계약은 블록의 높이, 블록 해쉬 등 결정된 정보 (Deterministic Information)만 획득할 수 있어 비결정적인 정보 (Non-Deterministic information), 예를 들어, 난수 등을 사용하지 못하는 문제가 있다. 그 이유는 스마트계약을 연산하는 블록체인 노드마다 서로 다른 난수를 사용하게 되면 동일한 월드 상태를 가지지 않게 되기 때문이다[6].

하지만 난수를 필요로 하는 경우도 있다. 예를 들어, 블록체인 기반 복권[7]이나 배팅 등이 있을 수 있다.

EVM과 블록체인 자체적으로 구할 수 없는 정보를 제공하는 서비스를 블록체인 오라클(Oracle)이라고 한다. 즉 블록체인 오라클 서비스를 통하여 블록체인 외부의 정보를 획득하는 것이다. 따라서, 난수도 블록체인 오라클 서비스를 통하여 공급할 수 있게 된다.

난수생성용 블록체인 오라클 서비스는 온라인으로 난수를 제공해주는 서비스이다. 즉 로컬머신에서 난수를 생성하는 것이 아니다. 따라서 난수 오라클 서비스는 온라인에 있는 어떤 난수라도 사용할 수 있을 것이다.

본 논문에서는 비정형 빅데이터를 entropy source, 즉 진성난수생성기(TRNG: True Random Number Generator)로 사용하였다. 해당 진성난수가 bias되어 있을 수 있어 de-skewing[8]을 하여 보정한 값을 난수로 사용하고자 한다. 좀 더 구체적으로 본 논문에서 인스타그램[9]을 entropy source로 이용한다. 인스타그램의 이미지 중 임의의 위치에서 정보를 읽어서 이를 de-skewing 한후에 난수 pool(random number pool)을 넣고 요청이 들어오면 앞쪽에서부터 필요한 만큼 난수를 제공해준다.

이렇게 생성한 난수가 난수의 통계적 특성이 있는지는 NIST SP800-22[10]의 난수 테스트 15개를 돌려서 확인하였다. 본 논문은 블록체인 환경에서 비정형 빅데이터를 난수의 entropy source로 쓸 수 있는지 실용적 관점에서 살펴보는 것이다.

본 논문이 기여하는 첫 번째는 온라인상에 있는 빅데이터를 entropy source, 즉 TRNG로 사용할 수 있음을 보이는 것이다. 본 논문에서는 인스타그램에서 받은 사진 이미지의 임의의 위치에 있는 비트 열들을 받아서 난수성 테스트를 통하여 실용적 관점에서 진성난수로 사용할 수 있음을 보인다. 따라서, 진성난수 생성기는 대기의 화이트 노이즈 등 물리적인 세계에서 구한다는 관점, 즉 하드웨어적으로 구현해야 한다는 관점을 뒤집어 소프트웨어적으로도 진성난수를 구할 수 있음을 보인다. 따라서, 고가의 난수 생성 하드웨어를 구매할 필요가 없다.

하드웨어 진성난수 생성기는 성능적으로는 문제가 없지만 고가의 장비를 설치하고 관리 운영해야 하는 비용 및 편리성 면에서 본 논문의 진성난수 생성기와 차이를 보인다.

기존의 소프트웨어적 난수 생성기는 1)시스템 시간 2)키 입력 간의 시간 간격 3) 입출력 버퍼의 내용 4) 사용자 입력 5) 운영체제의 여러 가지 값 등이 있다. 하지만 기존의 소프트웨어적 난수 생성기는 주로 로컬에서 사용하며, 서비스로 제공하기에 충분한 entropy를 획득하기 어려운 성능상의 문제가 있다.

두 번째로는 블록체인을 비정형 빅데이터에 기반한 진성 난수 생성기의 응용으로 선택하고 진성난수 생성용 블록체인 오라클 구조를 밝히고 실질적으로 구현하여 해당 구조를 바탕으로 인스타그램 뿐 아니라 다양한 빅데이터를 entropy source로 사용하는 블록체인 오라클 서비스구축을 위한 단초를 제공한다.

본 논문의 구성은 2장에서 전체 시스템 구조, 블록체인 오라클 및 난수 생성 알고리즘을 설명하고 3장

에서는 난수 테스트 결과 및 보안 분석을 살펴보고 4장에서 관련 연구를 살펴보고 5장에서 향후 계획을 포함한 논의를 한다. 마지막으로 6장에서 결언을 맺는다.

2. 블록체인을 위한 비정형 빅데이터를 이용한 난수생성 오라클

2.1 전체 시스템 구성도 및 오라클

본 논문에서는 이더리움 기반으로 네 개의 node로 구성된 PoA(Proof-of-Authority)[11] 기반 private network을 구성하였다.

난수를 생성하기 위한 기본적인 시스템 구성도는 Fig. 1과 같다.

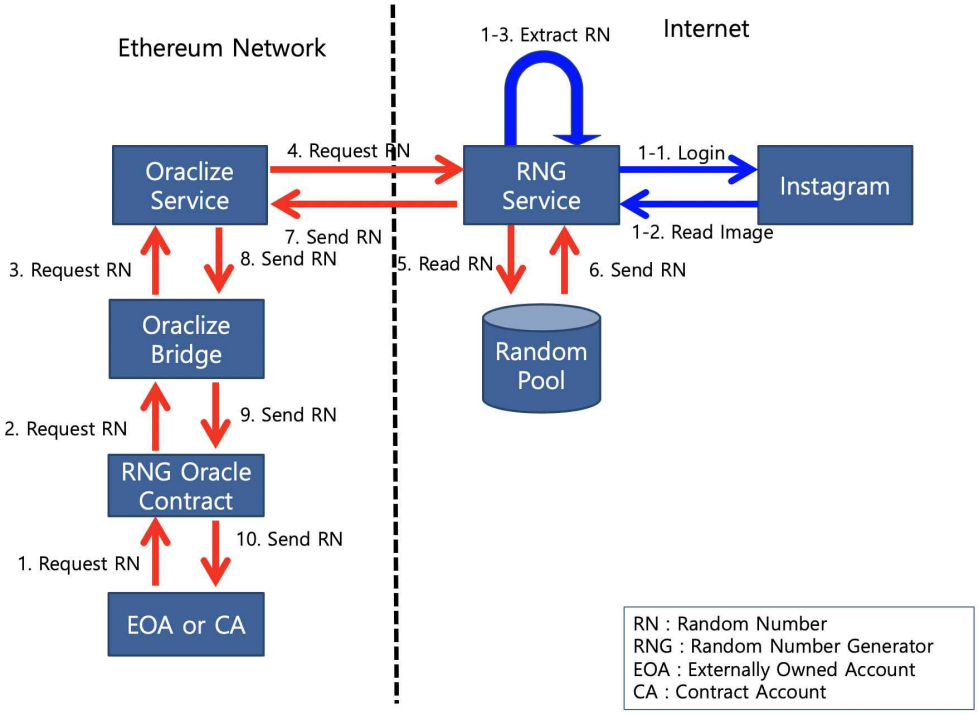

Fig. 1 블록체인 오라클을 위한 난수생성기의 전체적 구조

EOA(Externally Owned Account)나 CA(Contract Account)에서 난수를 요청하면 RNG(Random Number Generator) Oracle Contract에 난수를 요청한다.

본 논문에서는 간단히 난수생성을 테스트하기 위하여 Oraclize[12]를 활용하였다. Oraclize가 특별한 이유는 스마트계약에 제공되는 모든 데이터에 대해 신뢰성을 입증하기 때문이다[13]. Oraclize 서비스는 TLSNotary[14]를 통하여 진위 증명을 하게 된다. TLSNotary는 3개의 엔티티, 즉 감시대상(RNG Oracle Contract), 감시자(아마존 가상 시스템 인스턴스), 서버(Oraclize 서버)로 구성된다. 서버와 감시대상은 TLS를 통하여 대칭키를 받게 되고 감시자와 서버는 메시지 인증을 위한 MAC 키를 가지게 된다. 감시대상은 MAC키를 가지고 있지 않아서 변조를 할 수 없으며 감시자는 MAC키를 가지고 있어서 변조 여부를 확인할 수 있다. 즉 Oraclize는 감시대상이 값을 변경하지 않았다는 것을 확인할 수 있다. Oraclize의 TLSNotary에 대해서 좀 더 상세한 설명은 [13]와 [14] 에 나와 있다.

Oraclize는 개발자가 쉽게 오라클을 만들 수 있도록 정의된 Contract (usingOraclize contract)를 상속 받으면 쉽게 오라클을 만들 수 있다.

본 논문에서 usingOraclize를 상속받아 RNG Contract를 이더리움 공식 언어인 solidity 0.5.0 버전으로 Fig. 2와 같이 만들었다.

Fig. 2에서 일반적인 내용은 생략하고 오라클 부분만 설명을 한다. Private blockchain이나 자체 테스트넷인 경우 Oraclize를 사용하기 위해서 Oraclize Bridge[15]를 생성해야 한다. Fig. 1의 Oraclize Bridge가 이에 해당한다. Fig. 2 의 코드에서 OAR 변수에 bridge 주소가 들어간다. Oraclize Bridge를 통하여 Oraclize Service에 접근한다. Oraclize Service를 통하여 실질적으로 가져오는 난수는 RNG Service에서 받아오며 해당 URL 주소는 Fig. 2의 코드 중 getRand함수 내부에 있는 oraclize_query에 들어 있는 주소이다.

```
pragma solidity ^0.5.0;
import "github.com/oraclize/ethereum-api/oraclizeAPI.sol";

contract RNGContract is usingOraclize {
    string public resultCode;
    constructor() public payable {
        OAR=OraclizeAddrResolverI(0xF4de356135716Fbb066373ae
9EBcab91dfe614b5);
    }
    function __callback(bytes32 myid, string memory result) publi
c {
        resultCode = result;
    }
    function getRand() public payable {
 oraclize_query("URL","json(http://14.63.199.16:18080/random).res
ultCode");
    }
}
```

Fig. 2 랜수생성 스마트계약

블록체인 오라클은 즉시 읽기, 게시-구독, 요청-응답의 세 가지 방법으로 서비스할 수 있다. 본 논문에서는 즉기 읽기 방식을 사용하였다. 즉시 읽기 방식은 즉각적으로 필요한 정보를 조회하고 그 결과를 요청하는 스마트계약 저장소에 저장하는 방식이다. 예를들어, 배팅이나 복권 같은 경우 누구나 결과를 확인할 필요가 있을 때 사용한다. 즉 본 논문의 난수생성용 블록체인 오라클에 난수를 요청한 노드와 다른 노드에서 PoW(Proof of Work)나 PoS(Proof of Stack) 등으로 합의할 때 해당 스마트계약의 저장소에 난수가 저장되어 있어 동일한 연산이 가능하여 어떤 노드나 최종 연산 결과 동일한 EVM의 상태머신(State Machine)을 가지게 되어 합의가 가능하게 된다. 또한, 전술한 TLSNotary를 통하여 스마트계약의 저장소에 저장된 해당 난수가 변경되어 않았음을 확인할 수 있게 된다.

또 다른 방식으로 게시-구독 형태가 있다. 이 경우 오라클은 정기적 혹은 작은 변화가 예상되는 데이터를 브로드캐스트하는 방식이다. 마지막으로 요청-응답은 데이터 공간이 너무 커서 스마트계약의 저장소에 저장할 수 없으며, 사용자는 한 번에 전체 데이터 중 일부만 필요한 경우에 사용한다[6]. 난수를 암호학적 키 값으로 사용하는 경우 요청-응답 방식을 사용할 수 있다. 응용에 따라서 오라클 서비스 방법을 선택하면 된다.

2.2 난수 생성 알고리즘

난수 생성 알고리즘의 의사코드는 Fig. 3과 같다.

```
1. choose randomly account from account pool
2. login into Instagram
3. choose randomly a picture from pictures shown when login
4. choose randomly a position from middle of choosen picture
5. read 256 bits from the position
6. de-skewing the 256 byte using SHA-256
7. store the de-skewed random number into random pool
```

Fig. 3 난수생성 알고리즘

본 논문에서는 5개의 계정 중에서 랜덤하게 선택하여 인스타그램에 로그인한다. 인스타그램은 개인화된 페이지를 보여주기 때문에 계정이 다르면 보여지는 사진도 달라진다. 또한 같은 계정이라도 접속할 때마다 새로 등록된 사진을 제공해 준다. 접속 시 제공되는 그림은 24개이며 그 중 하나를 랜덤하게 선택한다. 인스타그램 계정은 본 논문의 난수생성용 블록체인 오라클 서비스가 가진 것으로 노드나 사용자가 인스타그램 계정을 가지고 있을 필요가 없다. 즉 스마트계약이 난수생성용 블록체인 오라클에 난수를 요청하여 받고, 스마트계약의 저장소에 난수가 저장되고, 다른 노드가 합의할 때는 저장된 난수를 이용하여 스마트계약을 동작하게 되어 동일한 연산 결과를 가지고 동일한 EVM 상태머신을 가지게 된다. 선택된 그림의 중간 위치(m)를 찾은 다음 랜덤하게 앞쪽 또는 뒤쪽을 선택하고 1~1000까지 랜덤한 값 (x)을 구한다. 만약 앞쪽이면 m-x 바이트부터 256 바이트를 읽고 만약 뒤쪽이면 m+x 바이트부터 256 바이트를 읽는다.

참고로 여기서 랜덤 값은 java의 java.SecureRandom 함수를 사용하여 얻었다. 비정형 빅데이터로부터 읽어들인 값은 bias되어 있을 수 있다. 따라서 해당 값을 de-skewing한다. de-skewing하는 다양한 방법[8]이 있지만 본 논문에서는 SHA-256 해쉬함수[16]로 해시를 하여 얻어진 값을 난수로 저장한다. 이렇게 생성된 난수를 30MByte가 될 때까지 지속적으로 생성하여 난수 pool에 저장하고 난수 요청이 들어오면 난수 pool에서 읽어서 제공해 준다.

트위터같은 텍스트 빅데이터 소스는 터미널문자만을 포함하여 엔트로피가 낮을 수 밖에 없다. youtube 동영상의 경우 스트림으로 제공되어 임의의 위치에서 난수를 가져오기 위해서 상당시간을 대기해야 하여 성능적인 문제가 있다. 이러한 이유로 이미지를 제공해주는 인스타그램을 빅데이터 소스로 선택하였다. 하지만 다른 빅데이터 소스가 난수의 entropy source로 사용할 수 없다는 것을 의미하지는 않는다.

3. 실험 결과

3.1 통계적 난수성 테스트

난수 생성기로 생성된 난수들이 난수인지 확인이 필요하다. 생성된 난수가 가지는 통계적 테스트를 통하여 난수 생성기가 맞는 지 확인하는 방법은 무한의 방법이 있을 수 있다. 본 논문에서는 NIST SP800-22에 제시된 15가지 통계적 테스트를 통하여 본 논문에서 사용한 난수 생성기가 난수 생성을 하는지 확인하였다. 난수 테스트는 python으로 NIST SP800-22 구현 [17]을 이용하였다.

인스타그램에서 난수를 획득하면 한 줄에 하나씩 before_de-skewing 파일에 저장하고 저장한 난수를 de-skewing하고 한줄에 하나씩 de-skewing 난수를 after_de-skewing 파일에 저장하였다. before_de-skewing 파일의 저장 위치와 after_de-skewing 파일의 저장 위치를 [17] 구현의 입력 파라미터로 넣으면 자동으로 난수성을 테스트한다.

Table 1은 NIST SP800-22 난수 테스트 결과이다. 보는 바와 같이 de-skewing 전에는 10개의 테스트가 실패하였으며 de-skewing 후에는 모든 테스트를 성공적으로 통과하였다. 이를 통하여 알 수 있는 것은 적어도 인스타그램에서 획득한 데이터를 바로 난수로 사용할 수 없고 de-skewing을 반드시 실시해야 한다는 것을 알 수 있다.

Table 1. NIST SP 800-22 난수성 테스트 결과

	Test Name	before de-skewing	after de-skewing(P value)
1	monobit test	fail	pass(0.449215772544)
2	frequency within block test	fail	pass(0.485390546496)
3	runs test	fail	pass(0.266392442826)
4	longest run ones in a block test	pass	pass(0.859770205989)
5	binary matrix rank test	pass	pass(0.943146298074)
6	dft test	fail	pass(0.668975782859)
7	non overlapping template matching test	pass	pass(0.999983501611)
8	overlapping template matching test	fail	pass(0.718484907467)
9	maurers universal test	pass	pass(0.999342585761)
10	linear complexity test	pass	pass(0.188138795829)
11	serial test	fail	pass(0.468697429596)
12	approximate entropy test	fail	pass(0.468401577766)
13	cumulative sums test	fail	pass(0.362981615981)
14	random excursion test	pass	pass(0.35931869258)
15	random excursion variant test	fail	pass(0.013070220507)

SP800-22는 수학적으로 난수성을 분석하여 난수인지 아닌지를 판별하는 것으로 각 15개의 테스트는 각각의 수학적 의미를 가진다. 각 난수테스트에서 나

온 P값을 Table 1.의 괄호에 표시하였다. .

3.2 보안 분석

공격자는 인스타그램 계정을 모르고 따라서 자신이 인스타그램에 접속했을 때 보는 이미지와 RNG Service가 보는 이미지가 다르다. 또한, 접속시 보이는 이미지 중 랜덤하게 하나를 선택하고 또 랜덤하게 이미지의 임의의 위치를 선택하여 256 바이트를 읽어서 SHA256 해시 후 난수 pool에 저장한다. 최종적으로 특정 사용자가 특정 시점에 요청하여 응답으로 받은 난수는 난수 pool에 들어간 시점(이미지, 이미지 위치)을 전혀 알 수 없기 때문에 공격자는 어떤 이미지의 어떤 위치에서 데이터를 읽었는지 파악할 수 없기 때문에 공격자는 어떤 난수를 사용자가 얻게 될지 전혀 예측할 수 없다. 즉, de-skewing 전의 데이터를 전혀 예측할 수 없기 때문에 de-skewing 후의 난수도 예측할 수 없다.

3.2 성능 분석

인스타그램에 접속하고 entropy를 얻는 것은 시간이 소요되는 작업이다. 1회당 30분간 entropy를 획득하고 실험을 3회를 실시하여 초당 entropy의 획득 비트 수를 얻었다. Table 2는 이를 보여 준다.

Table 2. 인스타그램 난수획득 속도

회	속도
1회	606bits/sec
2회	600bits/sec
3회	625bits/sec

이더리움은 현재 20 TPS(Transaction Per Second) 정도이다[18]. 2018년 3월 기준 이더리움 트랜잭션의 60%가 송금이며 40%정도가 스마트계약 호출 트랜잭션이다[19]. 또한, 대부분의 스마트계약 호출 트랙잭션(80%이상)이 소수의 스마트계약에 집중되어 있다[20]. 가장 많이 호출되는 이더리움 스마트계약는 ERC20 Token 관련이다[20]. 즉 토큰 획득, 전송 등이다. 따라서, 배팅이나 복권과 같이 난수를 필요로 하는 스마트계약의 트랜잭션은 미미할 것이다. 난수를 필요로 하는 트랜잭션을 10% (> 8% = 40% * 20%) 정도의 가정하면 초당 2번 정도의 난수 요청이 있을 것이다. 최저의 속도(600bits/sec)를 고려했을 때 256bits를 두 번은 제공할 수 있다. 따라서, 현재는 문제가 없으나 이더리움이 PoS로 넘어가면 속도가 상당히 빨라 질 것이다. 이를 대응하기 위해서는 더 많은 인스타그램 계정을 이용하던지 아니면 다른 빅데이터 소스 (youtube 등)도 추가해야 할 것이다. 또는 256bits를 제공하는 것이 아니라 32bits 정도의 seed를 제공하는 것도 방법이다.

4. 관련 연구

블록체인의 난수에 관한 기존 연구는 주로 난수를 생성할 때 발생되는 fairness 문제를 중심으로 연구가 되어 있고 생성한 난수 그 자체가 얼마만큼 안전한 난수인지에 대한 연구는 아직 없다. 본 논문에서 중요한 점은 안전한 난수를 생성하는 것이며 그 응용으로써 블록체인 오라클을 선택한 것이다. 따라서, 본 논문에서 제안하는 비정형 데이터, 인스타그램을 이용한 안전한 난수는 어떤 응용에서도 사용할 수 있다.

기존의 블록체인에서 난수를 얻는 방식은 1) block hash나 타임스탬프를 활용하는 방식 2) 블록체인 오라클을 이용하는 방식 3) 난수를 해쉬하여 commitment로 제출하고 이 후에 난수를 공개하는 방식 4)RANDAO[21] 방식으로 나뉜다 [22]. 전술한 바와 같이 [22] 논문에서는 각 방식의 장단점을 fairness 측면에서 분석하였으나, 생성된 난수 자체의 안전성을 검토한 것은 아니다. 블록체인 오라클을 이용하는 방식에서는 오라클 서비스를 신뢰해야만 하는 것이 단점이라고 분석하고 있다. 하지만 2장 1절에서 밝힌 것처럼 Oraclize 서비스는 TLSNotary를 이용하여 난수 생성 Oracle Contract가 마음대로 난수를 변경하거나 조작하지 못함을 설명하였다. [22] 논문에서는 random.org[23]와 같은 난수 서비스를 이용할 수 있다고 밝히고 있다. random.org는 대기 중의 백색 잡음을 이용한 진성 난수 생성 서비스이다. 즉 random.org는 물리적인 세계에서 난수를 만들어 사용하는 방식이며 본 논문은 비정형 빅데이터에서 진성 난수를 만드는 방식이 차이점이다. 두 방식 모두 NIST SP800-22를 통과할 것이므로 난수성 측면에

서 우열을 가릴 수는 없다. 다만 random.org와 같은 물리적 세계에서 난수를 가져오려면 특별한 장치가 필요하며 이에는 비용이 발생할 것이다.

5. 향후 계획 및 토론

본 논문에서는 인스타그램을 entropy source로 이용하였다. 인스타그램만 사용하는 경우 일정 시간 이지나야 로그인되며 빠른 속도로 접근할 때 일시적으로 차단되는 특성이 있었다. 따라서, 아주 빠르게 난수를 생성할 수 없는 문제점이 있으며 만약 블록체인 오라클에 매우 빠른 속도로 난수 요청이 들어오면 난수 pool이 소진될 문제점이 있다. 이 문제를 해결하기 위해서 향후에는 페이스북, 유튜브 등 다양한 비정형 빅데이터를 동시에 사용하여 좀 더 빠르게 난수를 획득하는 방안을 적용 해 볼 계획이다.

6. 결 론

본 논문은 블록체인의 스마트계약이 직접 구할 수 없는 난수를 블록체인 오라클 서비스로 제공하는 난수생성용 블록체인 오라클 서비스를 구축하였다. 또한 본 논문은 실용적으로 사용할 수 있는 비정형 빅데이터를 entropy source로 사용하였다. 좀 더 구체적으로는 인스타그램의 사진을 entropy source로 사용하였다.

인스타그램의 개인화 맞춤 이미지 제공 특성에 따라서 본 논문에서 구축한 오라클 서비스가 접근하는 이미지와 공격자가 접근한 이미지가 같은 시간에 접근하여도 서로 다르다. 또한, 오라클 서비스가 획득한 이미지의 특정 부분을 de-skewing하여 난수 pool에 저장하고 있다가 난수 생성 요청이 들어오면 제공하는 방식을 사용하여 사용자가 요청한 시점에 제공되는 난수가 어떤 시간에 어떤 이미지의 어떤 위치의 값을 de-skewing했는지 전혀 예측할 수 없다. 따라서, 비정형 빅데이터를 entropy source로 실용적으로 이용할 수 있다는 것을 본 논문에서 보여 준다.

마지막으로 NIST SP800-22 통계적 난수 테스트를 이용하여 온라인으로부터 획득한 비정형 빅데이터를 바로 사용할 수 없고 de-skewing을 해야만 한다는 사실도 본 논문은 보여준다.

참고문헌

[1] 신동진, 박창섭, "Hyperledger Fabric을 이용한 중첩형 무한 해시체인 기반의 클라이언트 인증기법", 융합보안논문지, 제18권제4호., pp. 3-10, 2018.

[2] 지승원, 이원기, 고태광, 박소희, 오구연, 김종민, 김동민, "블록체인 기반의 SCADA 시스템 보안", 융합보안논문지, 제19권제5호., pp. 55-61, 2019.

[3] 지승원, 이원기, 고태광, 박소희, 오구연, 김종민, 김동민, "블록체인 기반의 SCADA 시스템 보안", 융합보안논문지, 제19권제5호., pp. 55-61, 2019.

[4] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger.Ethereum Project," Yellow Paper, 2014.

[5] N. Szabo, "Smart Contract," http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html, 1994.

[6] A. M. Antonopoulos and G. Wood, "Mastering Ethereum," O'Reilly, Dec. 2018.

[7] Quanta, www.quanta.im

[8] Katz, J., Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. "Handbook of applied cryptography," CRC press, 1996.

[9] Instagram, https://www.instagram.com/

[10] R. Andrew, et al. "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Booz-Allen and Hamilton Inc Mclean Va, 2001.

[11] POA, https://en.wikipedia.org/wiki/Proof_of_authority

[12] Oraclize, http://provable.xyz/

[13] Prusty, Narayan. "Building Blockchain Projects," Packt Publishing Ltd, 2017. pp203-208.

[14] TLSNotary, https://tlsnotary.org/TLSNotary.pdf

[15] Oraclize bridge, https://github.com/provable-things/ethereum-bridge

[16] Pub, NIST FIPS. "180-2." Secure Hash Standard, National Institute of Standards and Technology, US Department of Commerce, 2004.

[17] A python implementation of the SP800-22 Rev 1a PRNG test suite https://github.com/dj-on-github/sp800_22_tests

[18] Who Scales It Best? Blockchains' TPS Analysis, https://hackernoon.com/who-scales-it-best-blockchains-tps-analysis-pv39g25mg

[19] Ranking Ethereum Smart Contractshttps://medium.com/@vikati/ranking-ethereum-smart-contracts-a27e6f622ac6

[20] Ethereum Smart-Contracts: Most of them are rarely used !, https://hackernoon.com/ethereum-smart-contracts-most-of-them-are-rarely-used-f45749730d3e

[21] randao.org, "RANDAO : A DAO working as RNG of Ethereum," Feb. 2019, https://github.com/randao/randao.

[22] Chatterjee, Krishnendu, Amir Kafshdar Goharshady, and Arash Pourdamghani. "Probabilistic Smart Contracts: Secure Randomness on the Blockchain." arXiv preprint arXiv:1902.07986 (2019).

[23] Randomness and Integrity Services, "True random number services," https://www.random.org

〔저자소개〕

정승욱 (Seung Wook Jung)
2005년 12월 : University of Siegen, 전자정보공학박사
현재 : 건양대학교 사이버보안공학과 교수
관심분야 : 개인정보보호, 블록체인, 네트워크 보안, 암호학 등
email :swjung@konyang.ac.kr