ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 20-04-02

j.inst.Korean.electr.electron.eng.Vol.24,No.4,931~941,December 2020
http://dx.doi.org/10.7471/ikeee.2020.24.4.931
11

경량 작업증명시스템을 이용한 스마트 홈 접근제어 연구

A Study on a Smart Home Access Control using Lightweight Proof of Work

김 대 엽*★

DaeYoub Kim*★

Abstract

As natural language processing technology using machine learning develops, a Smart Home Network Service (SHNS) is drawing attention again. However, it is difficult to apply a standardized authentication scheme for SHNS because of the diversity of components and the variability of users. Blockchain is proposed for data authentication in a distributed environment. But there is a limit to applying it to SHNS due to the computational overhead required when implementing a proof-of-work system. In this paper, a lightweight work proof system is proposed. The proposed lightweight proof-of-work system is proposed to manage block generation by controlling the work authority of the device. In addition, this paper proposes an access control scheme for SHNS.

요 약

기계학습을 이용한 자연어처리 기술이 발전하면서 SHNS (Smart Home Network Service)가 다시 주목받고 있다. 그러나 SHNS는 구성 기기의 다양성과 사용자의 가변성 등으로 인하여 표준화된 인증 시스템 적용이 어렵다. 블록체인은 분산 환경에서 데이터 인증을 위한 기술로 제안되고 있지만, 작업증명시스템 구현 시 요구되는 계산 오버헤드 때문에 SHNS에 적용하는데 한계가 있다. 본 논문에서는 경량화된 작업증명시스템을 제안하였다. 제안하는 경량화된 작업증명시스템은 기기의 작업 권한을 제어함으로써 블록 생성을 관리하도록 제안되었다. 또한 본 논문에서는 이를 기반으로 SHNS의 접근통제 방안을 제안한다.

Key words : SHNS, Blockchain, PoW, Access Control, Right Management

Ⅰ. 서론

스마트 폰, 스마트 와치와 같은 다양한 모바일 기기의 보급과 함께 머신러닝/딥러닝을 사용한 자연어 처리 기술이 획기적으로 개선되면서 그동안 정적인 기술로 인식되어 사용자 접근성이 낮았던 스마트 홈 네트워크 서비스(Smart Home Network Service, SHNS)에 대한 관심이 다시 높아지고 있다. 그러나 기기 간 통신 및 음성을 이용한 사용자 인터페이스 구현과 같은 단편적 기술의 발전 외에도 SHNS의 보급에는 여전히 해결해야 될 몇 가지 문제들이 남아있다. 그 중에서도 SHNS 구현에 가

* Dept. of Information Security, Suwon University
★ Corresponding author
E-mail : daeyoub69@suwon.ac.kr, Tel : +82-31-229-8352
※ Acknowledgment
The paper was supported by The research grant of the University of Suwon in 2019.
Manuscript received Nov. 17, 2020; revised Dec. 14, 2020; accepted Dec. 15, 2020.

장 큰 장애물 중 하나는 SHNS 구성에 사용되는 기기들의 다양성이다. 이 기기들은 네트워크, 데이터 저장 공간, 계산 처리 능력과 같은 기기 성능에 있어서 그 편차가 매우 크다. 또한, 일반적으로 가정에서 사용되는 기기들이 서로 다른 다양한 제조사의 제품인 경우가 많다. 이와 같은 SHNS 구성 기기의 다양성으로 인하여 SHNS 구현 시 통일된 기기 규격을 일률적으로 적용하기 어렵다.

또 다른 제약사항은 SHNS를 이용하는 가정(Domain)의 구성원들이 갖는 특성, 즉 다양성 및 가변성이다. 일반적으로 SHNS를 이용하는 도메인 기기에 접근하는 구성원이 가변적일 수 있다. 구성원 가변성은 SHNS Usage Case를 매우 복잡하게 만드는 요인이 된다. 그러므로 이와 같은 구성원의 다양성 및 가변성을 고려하여 SHNS를 안전하게 구현하기 위해서는 기본적으로 구성원의 식별 및 인증 절차, 그리고 접근통제와 같은 보안 요구사항이 반드시 충족되어야 한다. 또한 이와 같은 보안 기술들을 구현 할 때, 기기의 특성을 고려하여 최소 사양의 기기에서도 구현이 가능해야 하며, 특정 제조사의 기술에 의존하지 않는 일반적인 기술로 구현이 가능해야 한다.

효율적이고 안전한 데이터 인증 기능을 필요로 하는 다양한 서비스에서 블록체인 기술이 주목을 받고 있다. 블록체인이 주목 받는 이유 중 하나는 중앙 관리식 인증 시스템 구축을 요구하지 않으면서도 데이터의 위/변조 시도를 효과적으로 탐지할 수 있기 때문이다. 이를 위하여 블록체인은 데이터 셋을 포함하는 블록들을 생성 순서에 따라 체인구조로 연결한 후, 블록체인 기반 서비스에서 운영/관리되는 블록체인이 여러 개 존재할 경우, 블록체인들 중에서 체인의 길이가 가장 긴 블록체인을 유효한 체인으로 간주한다. 블록체인에 연결된 블록들은 데이터 인증 정보를 활용한 연결 구조를 갖기 때문에, 블록체인에 포함된 하나의 블록을 변경하기 위해서는 해당 체인에 포함되어 있는 다수의 블록들을 수정해야 한다. 이와 같은 블록 수정 작업은 수정자에게 많은 계산을 요구하며, 다수의 블록들이 수정되기 때문에 수정 발생 여부가 쉽게 탐지될 수 있다. 이와 같은 블록체인의 특성을 기반으로 블록체인의 블록에 저장된 데이터의 위/변조 시도를 방지한다.

그러나 일반적으로 분산된 시스템을 운영하기 위해서는 작업에 참여하는 노드나 시스템의 권한을 적절하게 관리하는 것이 필요하다. 블록체인 시스템의 경우, 가장 긴 체인을 유효한 체인으로 간주하기 위해서는 블록의 생성 및 연결 작업 권한을 적절하게 제어할 수 있어야 한다. 만약 누구나 쉽게 블록을 생성하고 연결할 수 있다면 체인의 길이를 임의로 연장하려는 시도를 제어하기 어렵기 때문이다. 블록의 생성 및 연결 권한을 관리하기 위하여 블록체인에 적용된 기술 중 하나가 작업증명시스템(Proof of Work, PoW)이다. 작업증명시스템은 특정 작업에 참여하길 원하는 구성원이 자신의 권한/능력을 스스로 증명함으로써 작업에 유효하게 참여하는 일련의 기술적 과정을 의미한다[1].

블록체인 시스템에서 활용되는 작업증명시스템 중 하나는 제시된 특정 조건을 만족하는 값을 단방향 해시 값으로 갖는 데이터를 찾는 것이다. 일반적으로 단방향 해시 함수의 특성 상 주어진 특정 값을 해시 값으로 갖는 데이터를 유추하는 것은 매우 힘들고 이와 같은 데이터를 찾기 위해서는 많은 양의 시스템 자원을 필요로 한다. 그러므로 블록체인 시스템은 참여하는 기기들의 시스템 자원의 한계를 고려하여 해시 함수의 출력 데이터의 특정 조건을 선택하며, 참여자들은 주어진 조건을 만족하는 해시 값을 갖는 데이터를 찾음으로써 블록생성에 참여한다. 그러나 이와 같은 작업증명시스템은 여전히 많은 시스템 자원을 요구한다[2]. 그러므로 블록체인 기술을 SHNS에 적용하기 위해서는 이와 같은 시스템 자원 문제를 해결해야 한다.

본 논문에서는 블록체인 시스템을 SHNS의 콘텐츠/리소스 접근제어 서비스에 적용하기 위한 경량 작업증명시스템의 개념과 양방향 체인을 이용한 발급 권한관리를 제안하고, 홈 네트워크 시스템에 적용하기 위한 서비스 구성을 제안한다. 또한 제안된 경량 작업증명시스템 기반의 홈 네트워크 서비스의 안전성을 평가 한다.

II. 작업증명시스템

1. 작업증명시스템

작업증명시스템은 서비스 참여자에게 참여자의 시스템 자원 소비가 필요한 특정 작업을 요구함으로써 서비스 오남용 등과 같은 불법적인 행위를 단념하도록 하는 수단이다. 작업증명시스템의 개념은

C. Dwork와 M. Naor에 의해 처음 제안되었으며, M. Jakobsson와 A. Juels가 그 체계를 수립했다[3]. 작업증명시스템은 참여자의 시스템 자원(CPU, Memory, Network) 소비를 요구한다. 가장 일반적인 작업증명시스템은 비트코인 시스템에서 활용하고 있는 HaSHNScaSHNS가 있다[4]. HaSHNScaSHNS는 주어진 퍼즐의 솔루션을 찾기 위해서 많은 계산이 요구되지만 주어진 솔루션을 검증하는 작업은 매우 쉬운 특징을 갖는다. 그러므로 블록을 생성하고 블록체인에 연결하기 위해서 참여자는 시스템의 자원을 이용하여 주어진 퍼즐의 솔루션을 찾아야 한다. 일반적으로 퍼즐의 난이도는 계산량과 밀접한 관계가 있으며, 블록체인을 적용한 서비스의 특성에 따라 조절할 수 있으나 일반적으로 작업증명시스템의 제안 목적을 만족시키기 위하여 많은 계산을 요구한다[5].

2. 경량화 된 작업증명 시스템

본 절에서는 스마트 홈 네트워크 서비스의 접근제어를 블록체인을 활용하여 구현하기 위해 해시함수를 이용한 경량화 된 작업증명시스템을 제안한다. 본 논문에서 제안하는 작업증명시스템은 컴퓨터의 리소스 소모를 요구할 뿐만 아니라 작업 수행 권한을 갖고 있는 사용자만이 해당 작업을 수행할 수 있는 인증 기능도 함께 제공한다.

가. 해시 체인

일반적으로 단방향 해시함수 H는 입력 데이터 x가 주어졌을 때 함수의 결과 값인 $H(x)$를 쉽게 계산할 수 있다. 특히, $y=H(x)$라 할 때, y를 x의 해시 값이라 한다. 그러나 y 값이 랜덤하게 주어졌을 때, $y=H(x)$를 만족하는 해시 함수의 입력 데이터 x를 추정하기 어렵다. 여기서 '추정하기 어렵다'는 의미는 x를 추정하기 위해서 많은 양의 계산이 요구되며, 이는 x를 추정하여 획득할 수 있는 이득보다 높은 계산 비용을 지불해야 된다는 의미이다. 해시함수 H와 입력 데이터 x가 주어졌을 때, x에 대한 해시함수 계산을 반복적으로 수행한 결과들의 집합 $S(x)$를 해시 체인(Sequence Hash Chain)이라 한다. 즉, $S(x)$는 다음과 같이 정의된다.

$$S(x)=\{y_i|1\le i, y_1=H(x), y_i=H(y_{i-1})\} \quad (1)$$

여기서, y_i는 $H^i(x)$으로 표기하기도 한다. $H^i(x)$는 x에 대한 해시 값 계산을 i번 수행한 결과를 의미한다. 예를 들어 $H^2(x)$는 $H(H(x))$를 의미하고, $H^3(x)$는 $H(H(H(x)))$를 의미한다. 해시 함수 H가 n비트 해시 값을 출력한다고 가정할 때, 집합 $S(x)$의 크기는 최대 $N=2^n$을 넘을 수 없으며 집합의 크기는 해시 함수의 종류와 입력 데이터 x에 의하여 결정된다. 집합 $S(x)$의 최대 크기를 $L(x)$라고 하면, $H(x)=H^{L(x)+1}(x)$를 만족한다. 또한 집합 $S(x)$의 원소들은 생성 순서에 따라 구분되는 순차 집합(Sequential Set)이다. 단방향 해시함수의 특성에 따라 $S(x)$의 i 번째 원소인 y_i가 주어졌을 때, $i+1$번째 원소인 y_{i+1}을 쉽게 계산할 수 있다.

나. 역순서 해시 체인

역순서 해시 체인(Reverse Sequence Hash Chain)은 n개의 원소로 구성된 순차 집합 $S(x)$의 원소의 순서를 역순으로 구성한 순차 집합 $R(x,n)$을 의미한다. 즉, $n\le L(x)$이라 가정하면, $R(x,n)$는 다음과 같이 정의한다:

$$R(x,n)=\{r_i|r_i=y_{n-i+1}, 1\le i\le n\} \quad (2)$$

역순서 해시 체인의 원소는 그 순서에 따라서 $i-1$번째 원소 r_{i-1}과 i번째 원소 r_i 사이에 다음과 같은 특성이 있다:

$$\begin{aligned} H(r_i) &= H(y_{n-i+1})=H(y_{(n-i+2)-1}) \\ &= y_{n-i+2}=y_{n-(i-1)+1} \\ &= r_{i-1} \end{aligned} \quad (3)$$

그러므로 단방향 해시 함수의 특성에 따라 r_{i-1} 값이 주어지더라도 r_i 값을 예측하기 어렵다. 그러나 x와 n의 값이 주어지면 $R(x,n)=\{r_i\}$의 모든 원소들을 순서에 상관없이 쉽게 계산할 수 있다.

이와 같은 역순서 해시 체인의 성질을 이용하여 본 논문에서는 역순서 해시 체인을 작업증명 시스템으로 활용한다. 즉, i 번째 작업을 수행할 수 있는 권한을 갖고 있음을 증명하기 위한 작업증명 데이터로 r_i를 제시할 것을 요청한다.

역순서 해시 체인을 작업증명시스템으로 활용할 때, $i-1$번째 작업이 완료된 후, $i-1$번째 작업증명 데이터 r_{i-1}이 공개 되어도 역순서 해시체인의 특성에 따라 주어진 r_{i-1}를 이용하여 $r_{i-1}=H(r_i)$를 만

족하는 i번째 작업증명 데이터 r_i를 유추하기 어렵다. 오직 (x,n)을 소유한 사용자만이 i번째 작업 수행을 위하여 r_i를 생성할 수 있기 때문에 해당 사용자는 r_i를 제시함으로써 작업 수행 권한을 정당하게 소유하고 있음을 증명할 수 있다.

또한, 생성된 r_i의 검증은 이미 작업이 수행되어 공개된 r_{i-1}을 활용하여 누구나 쉽게 검증할 수 있다.

III. BC을 활용한 SHNS 접근 제어 구성

1. SHNS 접근 제어 권한

효율적인 SHNS 접근 제어를 위하여 SHNS를 구성하는 기기의 권한을 다음과 같이 구분 한다:

(1) 소유자 권한(Owner) : SHNS 내에서 유일한 권한으로, 작업증명 데이터 생성을 위한 랜덤 기초 데이터 x_0와 $l(x_0)$를 생성한 후, 안전하게 관리한다. 이 때, $l(x_0) \le L(x_0)$를 만족한다. 소유자는 SHNS 내의 기기에 관리자 권한을 부여한다.

소유자는 k 번째 관리자에게 작업증명 데이터 생성을 위한 랜덤 기초 데이터 x_k와 $l(x_k)$를 생성하여 할당한다. 이 때, $l(x_k) \le L(x_k)$를 만족한다. 소유자는 SHNS 접근 제어를 위한 유효한 블록체인을 생성하며, 블록체인의 관리를 위한 권한 블록과 검증 블록을 생성 한다.

(2) 관리자 권한(Manager) : 관리자는 SHNS 내에서 사용자 권한 기기들의 권한 데이터를 생성한다. k번째 관리자는 소유자로부터 할당 받은 x_k와 $l(x_k)$를 사용하여 사용자 권한 데이터를 생성한다.

(3) 사용자 권한(User) : SHNS 내에서 콘텐츠 및 리소스를 공유 한다:

2. 권한 데이터(Transaction)

SHNS의 접근제어를 위한 블록체인의 블록은 권한 데이터 리스트(트랜젝션 데이터 리스트, *TDL*)의 분산 인증을 제공한다. i번째 블록의 *TDL*은 SHNS의 소유자나 관리자에 의해서 생성된 권한 데이터(*TD*)의 리스트로 구성된다. *TD*는 SHNS를 구성하는 기기들의 식별 및 인증 데이터, SHNS를 구성하는 기기들의 리소스에 접근할 수 있는 기능 권한 데이터, 그리고 SHNS에서 공유되는 콘텐츠의 콘텐츠 접근 권한 데이터 등을 의미한다. *TD*는 SHNS의 소유자에 의해서 블록에 저장되고 블록체인에 연결된다. 블록체인에 연결된 블록에 저장된 *TD*는 소유자에 의해 인증된 것으로 간주되며, *TD*를 저장한 블록은 SHNS의 모든 기기들에 의해서 검증될 수 있다.

SHNS의 k번째 관리자에 의해서 생성된 i번째 *TD*의 구조는 다음과 같다:

(1) 일련번호(sn)는 SHNS의 관리자에 의해서 생성된 *TD*의 생성순번을 의미한다. $sn=1$인 *TD*는 원시 권한 데이터로, 해당 *TD*를 생성한 관리자의 인증 데이터를 저장한다.

(2) 트랜젝션 구분자($type$)는 *TD*의 종류를 설명하는 구분자이다. $type$은 $data$에 저장된 정보의 종류에 따라서 기기 식별/인증 데이터, 기기 권한 데이터, 기기 권한 폐지 데이터, 그리고 기기 접근 권한 데이터로 구분된다.

(3) 관리자 식별자(uid)는 해당 *TD*를 생성한 SHNS 관리자의 유일한 식별자이다.

(4) 데이터($data$)는 *TD*에 저장되는 식별/인증 또는 권한 데이터를 의미한다.

- 기기 식별, 인증 데이터 : SHNS의 구성원으로 등록하려는 기기의 식별 데이터를 의미한다.
- 기기 권한 데이터 : SHNS의 구성원이 공유하는 콘텐츠 또는 리소스에 접근하기 위해 충족되어야 하는 접근 요구 사항을 의미한다.
- 기기 권한 폐지 데이터 : SHNS의 구성원으로 자격을 상실한 기기의 식별자를 의미한다.
- 기기 접근 권한 데이터 : SHNS에서 공유되는 콘텐츠 또는 리소스에 접근할 수 있는 권한을 증명하는 데이터를 의미한다.

(5) *TD* 데이터 인증 정보(tac)는 데이터 인증을 위해서 해당 *TD*가 정당한 권한을 가진 관리자에 의해 생성되었고, 그 내용이 위/변조 되지 않았음을 검증하기 위해서 사용된다. 메시지 $m=(i,type,uid,data)$에 대한 tac는 다음과 같이 생성되다.

$$tac = H(key, m) = H(H(x_k, i), m) \tag{4}$$

만약 $sn \ge l(x_k)$이면, 관리자는 소유자에게 새로운 x_k의 할당을 요청해야 한다. 이 경우, 소유자는 해당 관리자의 권한이 새롭게 정의되었음을 공지하는 *TD*를 생성해야 한다.

3. 접근 제어 블록(Block)

가. 블록 구성

SHNS 접근제어에서 활용되는 블록체인을 구성하는 블록은 블록 헤더(BH), 블록인증정보(BA), 권한 데이터 리스트(TDL)으로 구성된다.

i번째 블록의 블록 헤더(BH_i)의 구조는 다음과 같다:

(1) 일련번호(SN_i) : i번째 블록의 생성 순번을 의미한다. 원시 블록의 일련번호는 1번이며, 이 후 생성된 블록은 블록체인에 연결된 순서에 따라 순차적으로 1씩 증가된 값이 할당된다.

(2) 작업증명 데이터(POW_i) : 일련번호가 i인 블록의 작업증명 데이터는 $R(x_0,l(x_0))$의 i 번째 원소인 r_i를 의미한다.

$$POW_i = r_i = \begin{cases} H^{l(x_0)} & ,i=1 \\ H^{l(x_0)-i}(x_0) & ,i>1 \end{cases} \quad (5)$$

(3) 블록 연결 정보(PH_i) : 일련번호가 i인 블록을 검증된 블록체인의 일련번호가 $i-1$인 블록과 연결시키기 위한 정보를 의미한다. 블록연결정보가 $H(BH_{i-1})$와 같이 계산되면, 일련번호가 $i-1$인 블록의 블록헤더 정보의 해시 값을 의미한다.

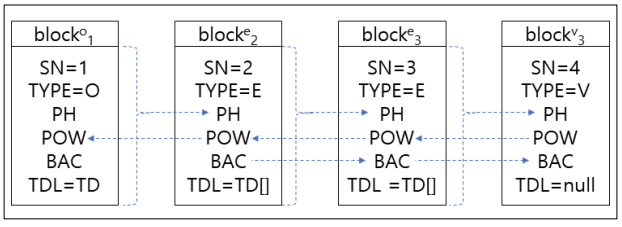

Fig. 1. Block Verification using Two Way Links.
그림 1. 양방향 링크를 이용한 블록 검증

(4) 권한 데이터 인증 정보(TDA_i) : 일련번호가 i인 블록에 저장된 권한 데이터 셋의 검증을 위한 정보를 의미한다. 일반적으로 머클 해시 트리를 이용하여 권한 데이터 셋을 인증할 경우, 머클 해시 트리의 루트 노드 값을 의미한다. 그러나 이 경우, TDA_i값 생성 및 검증을 위해 해시 값 계산을 반복적으로 수행해야 한다. 성능 개선을 위해서 권한 데이터 리스트 전체를 해시하는 방안을 이용한다. 소유자는 TDA_i를 이용하여 i번째 블록에 저장된 권한 데이터 리스트를 인증한다.

블록 인증 정보(BAC_i)는 일련번호가 i인 블록이 위/변조 되지 않았음을 증명하기 위해 사용된다. i번째 블록의 BA는 다음과 같이 계산된다.

$$BAC_i = H(key, BH) = H(r_{i+1}, BH_i) \quad (6)$$

i번째 블록의 인증정보는 $i+1$번째 블록의 $POW_{i+1} = r_{i+1}$이 공개되면 누구나 i번째 블록을 검증할 수 있다.

나. 블록 종류

블록체인을 활용한 효율적인 SHNS 접근 제어를 위하여 다음과 같은 블록을 정의한다:

(1) 원시 블록($block_1^o$) : 원시 블록은 SHNS에서 관리되는 블록체인의 초기 생성 블록을 의미한다. SHNS에서 관리되는 블록체인은 소유자에 의해 생성/관리되는 $(x_0,l(x_0))$에 의하여 구분되며, 원시블록의 $SN=1$이고, POW은 $R(x_0,l(x_0))$의 첫 번째 원소인 $r_1 = H^{l(x_0)}(x_0)$이다. PH는 0값을 갖는다. BA 생성에 사용되는 HMAC 키도 r_1이다. 원시블록에 저장되는 TDL은 1개의 TD만을 포함하며, 해당 TD의 $data$에는 소유자의 권한인증정보인 r_1와 $l(x_0)$가 저장된다.

(2) 권한 블록($block_i^e$) : 관리자들이 생성한 권한 데이터들의 목록인 TDL을 인증하고 SHNS 내에서 공유하는 블록을 의미한다. 권한블록의 $SN=i>1$이다. TDL에 포함되는 개별 TD는 SHNS의 관리자에 의해서 생성되고, SHNS의 소유자에 의해서 검증된 후, TDL에 포함된다. 그러므로 사용자는 TDL에 포함된 개별 TD를 인증할 필요가 없다.

(3) 검증 블록($block_j^v$) : 권한 폐기 TD로 생성된 TDL을 포함한다. 만약 권한 폐기 TD가 없다면 TDL을 포함하지 않는 빈 블록일 수 있다. 검증 블록은 블록체인에 포함된 개별 블록의 검증을 위하여 소유자가 주기적으로 또는 사용자의 요청에 의해 생성한다.

다. 블록 검증

블록체인의 i번째 블록 검증을 위하여 사용자는 별도의 등록과정(Enrollment Process)를 통해 SHNS의 구성원으로 등록되고, 이 과정에서 관리자에 의해서 소유자의 권한인증정보($H^{l(x_0)}(x_0)$)가 안전하게 공유되었다고 가정한다.

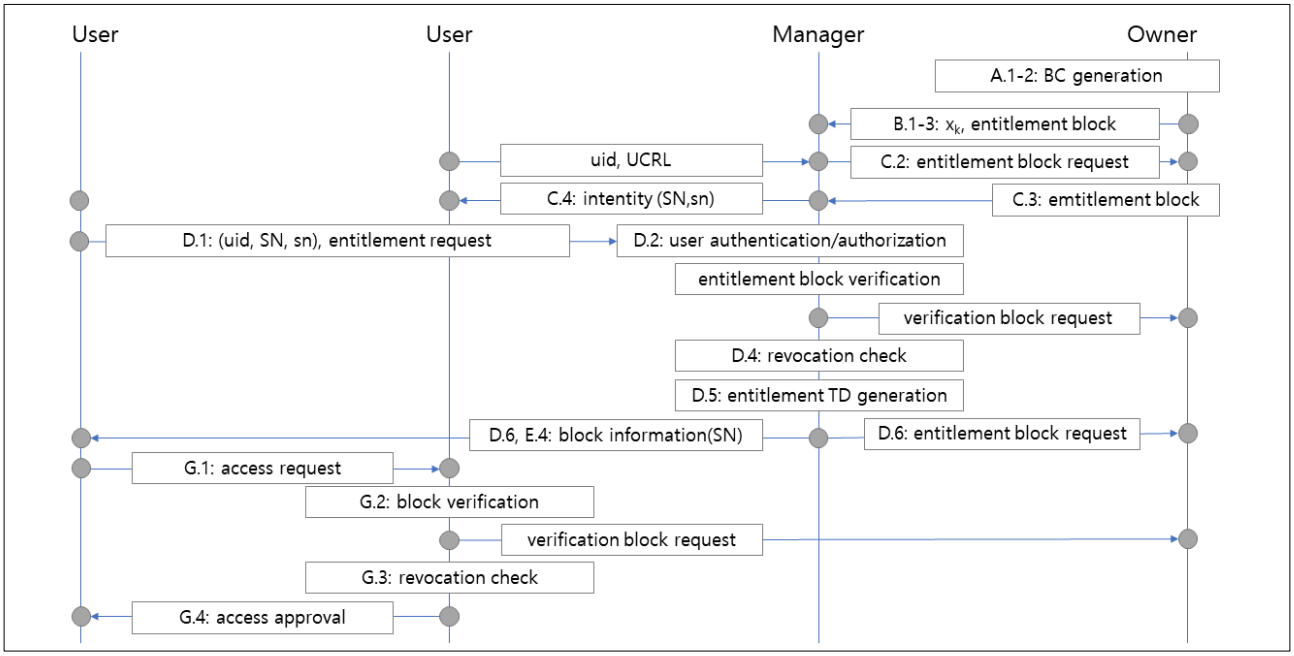

Fig. 2. Access Control Procedure.
그림 2. 접근통제 절차

그림 1은 블록 검증 절차의 예이다. 제안하는 기법은 일반 블록체인과 달리 양방향 링크를 통해 블록의 발급 권한 인증과 블록 인증을 수행한다. 블록체인의 i번째 블록 $block_i^*$을 검증하는 절차는 다음과 같다.

(1) 만약 $i=1$이면, 해당 블록은 원시 블록 $block_1^o$이다. 이 경우, 사용자는 등록과정에서 관리자로부터 전달 받은 소유권자의 권한인증정보를 이용하여 POW_1과 BA를 검증한다.

(2) 만약 $i>1$이면, 검증자는 $block_i^*$이 블록체인에 정상적으로 연결되어 있는지를 확인한다. 이를 위하여 검증자는 $SN=i-1$인 $block_{i-1}^*$을 획득하여 $block_i^*$의 PH를 검증한다. 검증자는 $block_i^*$이 소유자에 의해 정상적으로 생성된 것인지를 확인한다. 이를 위하여 검증자는 $block_i^*$의 $POW_i=r_i$를 검증한다. 만약 $H(POW_i)=H(r_i)$가 $block_{i-1}^*$의 $POW_{i-1}=r_{i-1}$와 같다면, $block_i^*$의 POW_i가 검증된 것으로 간주한다. 검증자는 $block_i^*$의 데이터를 인증하기 위하여 $block_i^*$의 BAC를 검증한다. 이를 위하여 검증자는 $SN=i+1$인 $block_{i+1}^*$을 획득한다. 만약 $block_{i+1}^*$이 블록체인에 존재하지 않는다면, 검증자는 소유권자에게 $block_{i+1}^v$ 생성을 요청한다. $block_{i+1}^*$의 POW_{i+1}를 이용하여 데이터 검증을 위한 HMAC 키를 생성하고, $block_i^*$의 BAC를 검증한다.

Ⅳ. BC을 활용한 SHNS 접근제어 절차

1. BC 생성

소유권자는 SHNS 접근제어를 위한 블록체인을 생성한다.

(A.1) 소유권자는 자신의 권한인증정보 x_0와 $l(x_0)$를 생성한다.

(A.2) x_0와 $l(x_0)$을 이용하여 원시 블록 $block_1^o$을 생성한 후, $block_1^o$를 SHNS 내에 공개 한다.

2. 관리자 지정

(B.1) Ⅳ-3-다 절에서 가정한 것처럼 등록과정을 통해 기기의 식별 및 인증절차가 완료된 후, 소유자는 해당 기기를 관리자로 지정할 수 있다. 이를 위하여 소유자는 권한인증데이터 x_k와 $l(x_k)$를 생성하고, 해당 기기에게 안전하게 전달한다.

해당 기기는 x_k를 안전하게 관리해야 하며, 해당 기기가 관리자로써 TD를 생성할 수 있는 권한 한계는 $l(x_k)$로 지정된다. 즉, 해당 기기는 TD를 최대 $l(x_k)$개 생성할 수 있으며, 이후에 $(x_k, l(x_k))$는 폐기된다.

(B.2) 소유자는 해당 기기를 관리자로 지정한 TD

를 생성하고, 생성된 *TD*를 포함하는 $block_i^e$을 생성하여 블록체인에 연결하고, 해당 블록을 SHNS 내에 공개한다.

(B.3) $block_i^e$이 정상적으로 SHNS 내에 공개되면, 관리자 기기로 지정된 기기에게 $block_i^e$의 *SN*과 *TD*의 *sn*을 통보한다.

3. 사용자 지정

IV-3-다 절에서 가정한 것처럼 등록과정을 통해 기기의 식별 및 인증절차가 완료된 후, 관리자는 해당 기기를 SHNS 내의 사용자로 지정할 수 있다.

(C.1) 관리자는 해당 기기를 사용자로 지정한 *TD*를 생성한다. *TD*의 *data*는 등록과정에서 식별/인증된 해당 기기의 *uid*와 사용자의 콘텐츠/자원 목록 데이터(User Content/Resource List, UCRL)를 포함한다. 사용자의 콘텐츠/자원 데이터는 해당 사용자가 SHNS 내에 공유할 콘텐츠/자원 목록으로 접근 통제를 위한 접근요구사항이 명시되어 있다.

(C.2) 관리자는 생성한 *TD*의 인증 및 공개를 위해서 소유자에게 블록 생성을 의뢰한다.

(C.3) 소유자가 해당 *TD*를 포함한 $block_i^e$를 생성하면, 관리자는 해당 블록의 *SN*과 생성한 *TD*의 *sn*을 사용자 기기에게 전달한다.

(C.4) 사용자 기기는 *uid*와 함께 수신된 (*SN*, *sn*)을 식별 데이터로 활용한다.

단, 등록 절차 완료 후 콘텐츠/자원 목록 데이터의 수정이 발생한 경우, 등록 절차를 새로 수행해야 한다.

4. 접근권한 데이터 발부

SHNS의 사용자는 SHNS 내에서 공유되는 콘텐츠 및 기기의 리소스에 접근하기 위하여 관리자에게 접근권한 데이터 발부를 요청할 수 있다.

(D.1) 사용자 기기는 C.4 단계에서 수신한 식별 데이터 (*SN*, *sn*)와 접근하려는 콘텐츠/자원 정보를 관리자에게 전송하다.

(D.2) 관리자는 전송된 (*SN*, *sn*)을 이용하여 블록체인에서 대응되는 블록 $block_i^e$와 *TD*를 획득한다. 만약 $block_i^e$가 블록체인의 가장 최근 블록이라면, 관리자는 소유자에게 $block_{i+1}^v$생성을 요청한다.

(D.3) 관리자는 $block_i^e$와 *TD*를 인증하여 사용자가 정당한 SHNS의 구성원임을 확인한다.

(D.4) 관리자는 최근 검증블록의 *TDL*을 확인한다. 만약 사용자의 정보가 해당 *TDL*에 포함되어 있다면, 사용자의 요청을 폐기한다.

(D.5) 사용자가 접근권한을 요청한 콘텐츠 또는 리소스의 접근 정책에 따라 접근권한 데이터를 생성한 *TD*로 패키징한다. 이와 같이 생성된 *TD*는 해당 관리자가 인증한 것으로 간주된다.

(D.6) 관리자는 생성한 *TD*를 Pool에 저장한 후, 소유자에게 이를 통보한다.

5. 권한 블록 생성

소유자는 주기적으로 또는 관리자나 사용자의 요청에 의하여 블록을 생성하고 블록체인에 연결한다.

(E.1) Pool에 저장되어 있는 *TD* 중 권한 폐기 *TD*를 제외한 *TD*를 읽어 온 후, *TD*를 생성한 관리자의 정보를 이용하여 해당 *TD*의 *ac*를 검증한다. 만약 *ac*가 유효하지 않다면, 해당 *TD*를 폐기한다.

(E.2) Pool에서 읽어온 *TD*들을 이용하여 *TDL*을 구성한 후, *TDL*을 위한 $block_i^e$를 생성한다.

(E.3) 생성된 $block_i^e$를 공개하고, 관리자들에게 블록 정보를 공표한다.

(E.4) 관리자는 *TD* 생성을 요청한 사용자에게 해당 블록의 정보를 전달한다.

6. 검증 블록 생성

소유자는 주기적으로 또는 관리자/사용자의 요청에 의해서 검증 블록을 생성한다.

(F.1) 최근에 발행간 검증 블록의 *TDL*을 읽어온다.

(F.2) Pool에 저장되어 있는 *TD* 중 권한 폐기 *TD*를 읽어 온 후, 이 정보를 이용 F.1 단계에서 확보한 *TDL*에 추가하여 새로운 *TDL*을 생성한다. 만약 SHNS 내에 공개한 권 폐지 정보가 없다면, *TDL*은 NULL 값을 갖는다.

(F.3) 생성한 *TDL*을 위한 $block_i^v$를 생성한다.

7. 접근 권한 검증

사용자가 콘텐츠 또는 리소스에 접근하기 위해서 해당 콘텐츠 또는 리소스의 소유권자에게 사용자가 발급 받은 접근권한을 제시하고, 접근 허용을 요청한다.

(G.1) 사용자는 E.4 단계에서 관리자로부터 전송

받은 블록 정보를 활용하여, 콘텐츠 또는 리소스의 소유권자에게 접근 허용을 요청한다.

(G.2) 콘텐츠 또는 리소스의 소유권자는 블록 정보를 활용하여 해당 블록 ($block_i^e$)을 획득한 후, $block_i^e$을 검증한다. 단, $block_i^e$이 SHNS 블록체인의 최신 블록인 경우, SHNS 소유자에게 $block_{i+1}^v$생성을 요청한다.

(G.3) 최근 검증 블록의 *TDL*에 사용자의 정보가 포함되어 있는지 확인한다. 만약 사용자 정보가 권한 폐지 목록에 포함되어 있다면, 사용자의 요청을 폐기한다.

(G.4) $block_i^e$이 유효한 블록으로 인증된 경우, 콘텐츠 또는 리소스의 소유권자는 사용자의 접근권한이 명시된 *TD*가 유효한 것으로 간주하고, 사용자에게 접근을 허용한다.

Ⅴ. 분석

1. 안전성 분석

가. 발급 권한 증명

본 논문에서 제안하는 접근통제 기법을 활용할 때, 블록체인의 블록과 각 블록의 *TD*는 정당한 권한을 소유한 구성원만이 생성할 수 있다. 이를 증명하기 위하여 본 논문에서는 정당한 발급 권한을 소유한 구성원에게만 발급된 권한 인증 정보 x_k와 이를 이용하여 계산된 $POW_i = r_i$를 이용하여 생성 권한을 증명한다. 만약 i 번째 블록(또는 *TD*)이 생성되면, 정당한 생성 권한자가 해당 블록(또는 *TD*)을 생성했는지 여부를 확인하기 위하여 사용자들은 $i-1$ 번째 블록(또는 *TD*)를 획득한 후, $POW_{i-1} = H(POW_i)$을 만족하는지를 확인함으로써 POW_i를 검증할 수 있다. 단방향 해시함수의 특성에 의하여 공개된 POW_{i-1}을 이용하여 POW_i를 예측할 수 없기 때문에 x_k를 소유한 구성원만이 i번째 블록(또는 *TD*)를 생성할 수 있다.

나. 접근권한 위/변조 탐지

불법적으로 SHNS의 콘텐츠 또는 리소스에 접근하기 위해서는 권한 인증 정보를 소유하지 않은 공격자가 블록체인에 포함된 블록을 위조 하거나 블록의 *TDL*을 변조하는 경우를 고려할 수 있다. 블록의 위조는 소유자가 아닌 공격자가 새로운 블록을 생성하여 마치 소유자가 생성한 것처럼 SHNS 내에 공개해야 한다. 공격자가 블록을 위조할 수 있는 방법은 다음 같다.

(1) 생성한 블록($block_i^e$)을 기존의 블록체인의 끝 점에 삽입하는 경우 : 공격자가 $block_i^e$을 기존 블록체인의 끝 점에 삽입하기 위해서는 $block_{i-1}^e$에 연결해야 한다. 이를 위해서 공격자는 유효한 POW_i를 생성할 수 있어야 한다. 그러나 앞서 설명한 것처럼 소유자의 권한 인증 정보 x_0를 획득하지 못한 상태에서 공개된 POW_{i-1}을 이용하여 POW_i를 생성하기 어렵기 때문에 유효한 $block_i^e$을 생성할 수 없다. 또한 유효하지 않은 POW_i 값을 이용하여 $block_i^e$을 생성할 경우, 소유자가 i번째 블록을 정상적으로 생성하게 되면 공격자가 생성한 유효하지 않은 $block_i^e$는 체인에서 자동적으로 폐기 처리된다.

(2) 생성한 블록을 기존 블록체인의 중간 점에 삽입하는 경우 : 공격자가 생성한 블록($block_i^e$)을 기존 블록체인의 중간에 삽입하기 위해서는 기존 블록체인의 $block_{i+1}^e$에 저장된 블록 연결 정보(PH_{i+1})를 수정하거나 생성한 $block_{i+1}^e$의 BH_i가 PH_{i+1}에 대응하는 $PH_{i+1} = H(BH_i)$를 만족하도록 구성해야 한다. 그러나 단방향 해시 함수의 특성 상 주어진 PH_{i+1}을 해시 값으로 하는 해시 함수의 입력 값 BH_i를 임의로 생성하기 어렵다. 또한 PH_{i+1}을 수정하기 위해서는 기존블록체인의 모든 $block_{k>i+1}^*$을 수정해야만 한다. 이 경우, SHNS 내에는 두 개의 블록체인이 동시에 존재하게 되며, 이와 같은 상황은 SHNS의 블록체인을 관리하는 소유자에 의해 발견된다.

(3) 생성한 블록을 블록체인의 시작점으로 사용하는 경우 : 공격자는 원시 블록을 생성하고, 이 원시 블록을 기점으로 하는 새로운 블록체인을 생성하여 이 블록체인이 마치 유효한 블록체인인 것처럼 주장한다. 그러나 서비스 운영을 위해 본 논문에서는 사용자 등록 과정에서 소유자의 권한인증 정보를 검증할 수 있는 데이터를 제공하기 때문에, 공격자가 생성한 원시블록이 사용자가 제공 받은 소유자의 권한인증정보를 검증할 수 있는 데이터와 다를 경우, 해당 원시블록은 유효하지 않은 것으로 간주될 것이다. 그러므로 공격자가 생성한 블록체인은 유효하지 않은 것으로 판명된다.

다. 자격증명 복사 방지

또 다른 공격 시나리오는 공격자가 기존 블록체인의 모든 블록들의 POW_i를 복사하여 위조된 블록에 재사용하는 것이다. 이 경우, 기존 블록체인과 같은 길이의 위조된 블록체인이 SHNS 내에 존재하게 된다. 그러나 소유자가 새로 생성한 블록은 기존 블록체인에 연결될 것이며, 기존 블록체인의 길이가 위조된 블록체인의 길이보다 길게 되면 사용자들은 길이가 긴 기존 블록체인을 유효한 것으로 간주하기 때문에 위조된 블록체인은 자동으로 폐기된다.

그러나 기존 블록체인의 최신 블록이 검증블록인 경우, 새로운 블록이 연결되기 전까지 공격자의 위조 블록과 기존블록의 유효성을 검증할 수 없다는 문제점이 있다. 그러므로 향후 추가 연구를 통해 이와 같은 문제점을 보완할 필요가 있다.

2. 성능 분석

표 1은 k번째 권한 데이터(TD_k) 및 i번째 블록($block_i^*$) 생성 및 검증을 위한 계산 오버헤드를 나타낸다. 여기서 $block_i^e$은 2^m개의 TD들을 포함하고 있다고 가정한다.

TD_k의 생성과 검증 시, 관리자와 소유자는 메시지 인증 코드 값(tac_j)을 생성/검증한다. 이를 위하여 HMAC에서 사용하는 1회용 키 생성과 메시지 해시를 위하여 모두 2번의 해시 값 계산을 필요로 한다. 단, 재사용공격(Replay Attack)을 방지하기 위해서 관리자와 소유자는 발행된 TD의 sn을 관리해야 한다.

$block_i^*$의 생성을 위해서 소유자는 PH_i, TDA_i, POW_i 그리고 BAC_i를 생성해야 한다. 이를 위하여 각각 최대 1번, $2^{m+1}-1$번, $l(x_0)-i$번 그리고 $l(x_0)-i$번의 해시 값 계산이 필요하다. 그러나 다음과 같은 방법들을 적용하여 계산 오버헤드를 개선할 수 있다.

(1) 만약 소유자가 TDA_i 값으로 머클 해시 트리를 이용하지 않고 권한 데이터 리스트 전체를 해시하는 방안을 이용한다면, 1번의 해시 값 계산으로 충분하다.

(2) BAC_i 생성을 위하여 $POW_{i+1}=r_{i+1}$을 계산해야 한다. 이를 위하여 $l(x_0)-i-1$번의 해시 값 계산이 필요하다. 그러나 이와 같이 계산된 POW_{i+1}을 소유자가 저장한 후 $block_{i+1}^*$ 생성 시 재사용할 수 있다. 이 경우, POW_{i+1} 생성을 위하여 추가로 해시 값들을 계산할 필요가 없기 때문에, 실제 POW_i와 BAC_i를 생성하기 위한 계산 오버헤드를 $l(x_0)-i+1$로 간주해도 일반성을 잃지 않는다.

$block_i^*$의 검증을 위해서 사용자는 PH_i ,POW_i 그리고 BA_i를 검증해야 한다. 이를 위하여 최대 $i+2$번의 해시 값 계산이 필요하다. 그러나 사용자가 $block_j^*$ ($j<i$)를 검증한 후, PH_j를 저장하고 있다면, $i-j+2$번의 해시 값 계산으로 충분하다. 특히, SH에 오직 한 개의 블록체인만 운영되고 있다고 가정하면, $block_{i-1}^*$의 검증이 완료된 상태이므로 3번의

Table 1. Performace Evaluation(Computation Overheads).
표 1. 성능 분석(계산 오버헤드)

Operation			Hash Operations (hp)		Operator
			Worst Case	Improved Case	
TD_j	Generation	tac_j	2		Manager
	Verification	tac_j	2		Owner
$block_i^*$	Generation	PH_i	1		Owner
		TDA_i	$2^{m+1}-1$	1	
		POW_i	$l(x_0)-i$	$l(x_0)-i+1$	
		BAC_i	$l(x_0)-i$		
	Verification	PH_i	$1 \le hp \le i$		User
		POW_i	1		
		BAC_i	1		

해시 값 계산으로 $block_i^*$를 검증할 수 있다.

블록을 생성하기 위한 *POW* 생성은 일반적으로 시스템의 리소스 소비를 필요로 한다. 일예로 비트코인에 연결된 656522번째 블록(2020-11-12 09:55)의 *POW*는 '0x00000000000000000000b8851fe de0783 fdc849f378b5fc359c63caf15c61cfec'이다[6]. 이는 해시 값이 '0x000000000000000000'으로 시작되는 퍼즐의 솔루션을 찾는 것이다. 이 경우, 최대 2^{72}번의 해시 값 계산이 요구되며, 이는 한 대의 시스템으로 계산할 경우, 1,192,516,788,603 시간이 소요되는 작업이다(Intel Core 2 1.83 GHz processor under Windows Vista in 32-bit mode)[7]. 본 논문에서는 역순서 해시 체인을 사용하여 POW_i를 생성한다. POW_i를 생성할 때 발생하는 계산 오버헤드는 $l(x_0)$의 값에 의존한다. SHNS의 특성 상 홈 내부에서 발행되는 블록의 숫자는 매우 제한적이다. 그러므로 $l(x_0)=100{,}000$으로 설정해도 충분하며, 그 이상이 필요한 경우, x_0를 갱신하면 된다. $l(x_0)=100{,}000$인 경우에도 [7]과 같은 시스템 환경에서 POW_i 계산에 소요되는 시간은 1초 이하이므로 SHNS에 적용하는데 문제가 없다.

특히, TD_k와 $block_i^*$의 인증을 위해서 전자서명을 사용하지 않고 HMAC를 사용함으로써 TD_k와 $block_i^*$의 인증을 위한 오버헤드를 줄였다. [7]에 의하면, SHA-512는 99MiB/s의 성능을 보이는 반면에, RSA 2048 전자서명은 6.05ms, 검증은 0.16ms가 소요된다. 그러므로 전자서명 대신에 HMAC을 사용함으로써 TD_k와 $block_i^*$의 인증 정보 생성의 효율성을 최적화하였다.

Ⅵ. 결론

블록체인은 분산 환경에서 데이터 인증을 위하여 효율적으로 위/변조를 방지하는 기술로 주목을 받고 있다. 일반적으로 블록체인은 체인에 연결된 블록의 수를 기반으로 가장 긴 블록체인을 유효한 것으로 간주한다. 그러므로 블록체인에 연결되는 블록 생성을 통제할 필요가 있다. 이와 같이 블록 생성을 통제하는 기술 중 하나가 작업증명시스템이다. 일반적으로 작업증명시스템은 일정 수준 이상의 시스템 리소스를 소비하기 때문에 다양한 성능/기능의 기기들로 구성된 SHNS에 적용하기 어렵다는 문제점이 있었다.

본 논문에서는 경량화된 작업증명시스템을 제안하고, 제안된 작업증명 시스템을 블록체인에 적용하여 SHNS의 접근 제어를 구현할 수 있는 방안을 제안하였다. 본 논문의 제안은 다음과 같은 의미를 갖는다. 첫 번째, 제안된 기법은 계산 오버헤드를 획기적으로 줄여서 SHNS의 기기들에 적용할 수 있도록 하였다. 이를 위하여 작업증명 시스템으로 역순서 해시 체인을 활용함으로써 블록체인 시스템의 퍼즐 솔루션을 계산하기 위한 계산 오버헤드를 줄였고, 데이터 인증을 위하여 전자서명 대신 HMAC을 적용함으로써 계산 오버헤드를 개선하다. 또한 인증서 관리를 위한 추가 시스템 운영을 필요로 하지 않는다.

두 번째, 제안된 작업증명시스템은 접근 통제를 위한 비밀 정보 운영 시스템의 역할을 병행할 수 있게 함으로 별도의 구성원 인증 시스템을 구현할 필요가 없다.

References

[1] L. Debin and C. Jean. "Proof of Work can work," *The 5th WorkSHNSop on the Economics of Information Security (WEIS 2006), Robinson College, University of Cambridge, England,* pp. 26-28, 2006.

[2] S. Fan, S. Ghaemi, H. Khazaei, and P. Musilek, "Performance Evaluation of Blockchain Systems: A Systematic Survey," *IEEE Access*, vol.8, 2020. DOI: 10.1109/ACCESS.2020.3006078

[3] D. Cynthia and N, Moni, "Pricing via Processing, Or, Combatting Junk Mail," Advances in Cryptology ≪CRYPTO'92: Lecture Notes in Computer Science No. 740≫ (Springer), pp.139-147.

[4] A. Back, "HaSHNScaSHNS-Amortizable Publicly Auditable Cost-Functions," 1st Aug. 2002. [Online]. Available: http://www.haSHNScaSHNS.org/papers/amortIz able.pdf. [Accessed: 29-Sept- 2020]

[5] H. Aljassas and S. Sasi, "Performance Evaluation of Proof-of-Work and Collatz Conjecture Consensus Algorithms," *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, Riyadh, Kingdom of Saudi Arabia,

2019.
[6] The 656522th block of Bitcoin blockchain. [Online]. Available: https://www.blockchain.com/btc/block/000000000000000000b8851fede0783fdc849f378b5fc359c63caf15c61cfec [Accessed: 17-Nov-2020]
[7] Crypto++ 5.6.0 Benchmarks. [Online]. Available: https://www.cryptopp.com/benchmarks.html [Accessed: 17-Nov-2020]

BIOGRAPHY

DaeYoub Kim (Member)

1994 : BS degree in Math., Korea University.
1997 : MS degree in Math., Korea University.
2000 : PhD degree in Math., Korea University.
2000~2002 : Research Engineer, SECUI
2002~2012 : Senior Researcher and Project Manager, Samsung Electronics.
2012~ : Professor, Dept. of Information Security, Suwon Univ.