【서지사항】

【서류명】 특허출원서

【참조번호】 P20210705KR

【출원구분】 특허출원

【출원인】

【명칭】 광주과학기술원

【특허고객번호】 3-1998-099381-5

【대리인】

【명칭】 특허법인지담

【대리인번호】 9-2018-100261-1

【지정된변리사】 주한중

【포괄위임등록번호】 2021-020151-2

【발명의 국문명칭】 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템 및 방법

【발명의 영문명칭】 UNIVERSAL ZERO-KNOWLEDGE SUCCINCT NON-INTERACTIVE ARGUMENT OF KNOWLEDGE PROOF SYSTEM FOR STACK-BASED VIRTUAL MACHINE PROGRAM

【발명자】

【성명】 장재혁

【성명의 영문표기】 JANG Je Hyuk

【주민등록번호】 890921-1XXXXXX

【우편번호】 61005

【주소】 광주광역시 북구 첨단과기로 123 (오룡동,광주과학기술원 블록체인지능융합센터)

【발명자】

【성명】 이흥노

【성명의 영문표기】 LEE, Heung No

【주민등록번호】 661120-1XXXXXX

【우편번호】 61005

【주소】 광주광역시 북구 첨단과기로 123 (오룡동,광주과학기술원 전기전자컴퓨터공학부)

【출원언어】 국어

【심사청구】 청구

【이 발명을 지원한 국가연구개발사업】

【과제고유번호】 1711126352

【과제번호】 2020-0-00958-002

【부처명】 과학기술정보통신부

【과제관리(전문)기관명】 정보통신기획평가원

【연구사업명】 블록체인융합기술개발(R&D)

【연구과제명】 일반 연산을 검증하고 트랜잭션 검증량과 저장 공간을 줄여주는 유니버셜 영지식 증명 서킷 기반 가상머신 개발

【기여율】 1/1

【과제수행기관명】 (주)온더

【연구기간】 2021.01.01 ~ 2021.12.31

【취지】 위와 같이 특허청장에게 제출합니다.

대리인 특허법인지담 (서명 또는 인)

【수수료】

【출원료】	0	면	46,000 원
【가산출원료】	43	면	0 원
【우선권주장료】	0	건	0 원
【심사청구료】	13	항	715,000 원
【합계】	761,000 원		
【감면사유】	공공연구기관(50%감면)[1]		
【감면후 수수료】	380,500 원		

【발명의 설명】

【발명의 명칭】

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템 및 방법 {UNIVERSAL ZERO-KNOWLEDGE SUCCINCT NON-INTERACTIVE ARGUMENT OF KNOWLEDGE PROOF SYSTEM FOR STACK-BASED VIRTUAL MACHINE PROGRAM}

【기술분야】

【0001】 본 발명은 영 지식 증명 기술에 관한 것으로, 더욱 상세하게는 범용 스택 기반 가상 머신 프로그램을 위한 간결하고 비 상호적인 영 지식 증명 시스템 및 방법에 대한 것이다.

【발명의 배경이 되는 기술】

【0003】 영 지식 증명(ZKP, Zero-Knowledge Proof) 기술은 증명자가 검증자에게 민감 정보가 포함된 어떤 명제가 참임을 민감 정보의 공개 없이 증명하는데 사용되는 알고리즘의 모음이다. 자세히 설명하면, 증명자가 자신만이 알고 있는 민감 정보를 사용하여 주장하는 명제가 참임을 직접 증명하고, 그 과정의 올바름에 관한 영 지식 증거만을 생성 검증자에게 전달하고, 검증자는 영 지식 증거를 검증함으로써 민감 정보의 확보 없이 명제의 참 거짓을 판단할 수 있다. 영 지식 증명 기술의 활용 예로는 신분을 증명할 수 있는 개인 정보 데이터를 제시하는 것 대신에 자가 증명 과정의 올바름을 보여줌으로써 '그 사람만이 그것을 할 수 있다.' 고

하는 것을 인증하는 것이 있다. 영 지식 증명 기술의 또 다른 활용 예로는 스택 기반 가상 머신 (이하 스택 머신) 프로그램의 실행 결과의 올바름을 증명하는 경우가 있는데, 여기서 스택 머신 프로그램의 실행에 사용된 입력 데이터가 민감 정보일 수 있다.

【0004】최근 증명 과정에 상호작용이 필요 없고(Non-Interactive), 간결한(Succinct) 증명조건을 만들어 영지식증명을 효율적으로 수행하는 영 지식 스나크(ZK-SNARKs, Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) 기술이 주목을 받고 있다. 영 지식 스나크(ZK-SNARKs) 기술은 증명자와 검증자 간의 통신이 필요 없고, 영 지식 증거의 길이가 짧고, 증명 연산 량이 적으며, 검증 연산 량이 명제를 직접 실행하는 계산하는 연산 량보다 작다.

【0005】영 지식 스나크(ZK-SNARKs)는 암호학적 보안을 제공하기 위해 증명 알고리즘과 검증 알고리즘의 연산에 공통적으로 필요한 변수들을 레퍼런스 스트링(RS, Reference String)이라는 파일에서 불러와서 사용한다. 레퍼런스 스트링 (RS)의 변수들은 암호화 되어있으며, 신뢰 가능한 제3자가 신뢰설정(Trusted Setup) 절차를 거쳐 생성한다. 신뢰설정은 신뢰가능한 제3자의 필요와 신뢰 가능한 절차를 요구하므로 비용 및 시간이 크게 소요된다.

【0006】영 지식 스나크(zk-SNARKs)는 변수(RS)의 구조적 특징에 따라 명제 혹은 프로그램이 변경될 때마다 변수(RS)를 새로 신뢰설정(Trusted Setup)해야 하는 써킷 특정 영 지식 스나크(circuit-specific zk-SNARKs)와 명제 혹은 프로그램이 변경되어도 이전에 사용된 변수(RS)를 재사용할 수 있는 범용 영 지식 스나크

(Universal zk-SNARKs)로 나눌 수 있다. 일반적으로, 명제 및 프로그램이 고정되어 있다면 써킷 특정 영 지식 스나크가 범용 영 지식 스나크보다 훨씬 간결하다. 반면 써킷 특정 영 지식 스나크는 명제 및 프로그램이 바뀐다면 변수(RS)도 새롭게 생성되어야 하기 때문에, 증거의 길이가 크고 증명 및 검증이 느린 범용 영 지식 스나크(Universal zk-SNARKs)의 사용이 강제된다.

【선행기술문헌】

【특허문헌】

【0008】**(특허문헌 0001)** 1. 한국 등록특허공보 제10-1799517호 “인증 서버 및 방법”(공개일자: 2017년 04월 18일)

【발명의 내용】

【해결하고자 하는 과제】

【0009】본 발명은 영 지식 증명에서 스택 머신 프로그램의 실행 결과 증명에 사용 가능한 범용 영 지식 스나크 증명 시스템 및 방법을 제공한다.

【0010】본 발명은 명제가 스택 머신 프로그램인 경우에 한정하여 범용적이면서 간결한 범용 영 지식 스나크 증명 시스템 및 방법을 제공한다.

【0011】본 발명은 복호화 없이 암호화 상태에서 모듈을 통합하여 증거를 생성하고 검증할 수 있는 범용 영 지식 스나크 증명 시스템 및 방법을 제공한다.

【과제의 해결 수단】

【0013】 본 발명의 일 측면에 따르면, 스택 머신 프로그램을 위한 범용 영지식 스나크 증명 시스템을 제공한다.

【0014】 본 발명의 일 실시 예에 따른 스택 머신 프로그램을 위한 범용 영지식 스나크 증명 시스템은 사전에 정의된 연산 명령 코드를 써킷 다항식으로 변환하여 모듈을 생성하는 모듈 생성부, 하나 이상의 모듈을 통합하는 모듈 변환부 및 모듈의 연결 관계 정보를 적용하는 연결 관계 적용부를 포함할 수 있다.

【0016】 본 발명의 다른 일 측면에 따르면, 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 방법 및 이를 실행하는 컴퓨터 프로그램을 제공한다.

【0017】 본 발명의 일 실시 예에 따른 스택 머신 프로그램을 위한 범용 영지식 스나크 증명 방법 및 이를 실행하는 컴퓨터 프로그램은 사전에 정의된 연산 명령 코드를 써킷 다항식으로 변환하여 모듈을 생성하는 단계, 하나 이상의 모듈을 통합하는 단계 및 모듈의 연결 관계 정보를 적용하는 단계를 포함할 수 있다.

【발명의 효과】

【0019】 본 발명의 일 실시 예에 따르면, 스택 머신의 단순한 구조적 특징을 이용하여 스택 머신 프로그램의 연산 증명에 한정하여 범용적이면서 간결한 영 지

식 스나크로 블록 체인의 스마트 컨트랙트에 적용 될 경우 거래처리 속도를 높일 수 있다.

【0020】 또한, 본 발명의 일 실시 예에 따르면, 각 모듈의 선형성을 이용해 복호화 없이 암호화된 모듈을 변환할 수 있다.

【0021】 또한, 본 발명의 일 실시 예에 따르면, 새로운 스택 머신 프로그램의 모듈 간의 연결 관계 정보를 선형성을 이용해 복호화 없이 통합하여 적용할 수 있다.

【도면의 간단한 설명】

【0023】 도 1 및 도 2는 본 발명의 일 실시 예에 따른 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템을 설명하기 위한 도면들.

도 3 내지 도5는 본 발명의 일 실시 예에 따른 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템에 생성한 모듈의 예시들.

도 6내지 도 8은 본 발명의 일 실시 예에 따른 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템의 모듈 변환을 설명하기 위한 도면들.

도 9 및 도 10은 본 발명의 일 실시 예에 따른 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)의 연결 관계 정보 적용을 설명하기 위한 도면들.

도 11은 본 발명의 일 실시예에 따른 스택 머신 프로그램을 위한 범용 영 지

식 스나크 증명 방법을 도시한 도면.

【발명을 실시하기 위한 구체적인 내용】

【0024】본 발명은 다양한 변경을 가할 수 있고 여러 가지 실시 예를 가질 수 있는 바, 특정 실시 예들을 도면에 예시하고 이를 상세한 설명을 통해 상세히 설명하고자 한다. 그러나, 이는 본 발명을 특정한 실시 형태에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다. 본 발명을 설명함에 있어서, 관련된 공지 기술에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우 그 상세한 설명을 생략한다. 또한, 본 명세서 및 청구항에서 사용되는 단수 표현은, 달리 언급하지 않는 한 일반적으로 "하나 이상"을 의미하는 것으로 해석되어야 한다.

【0025】이하, 본 발명의 바람직한 실시 예를 첨부도면을 참조하여 상세히 설명하기로 하며, 첨부 도면을 참조하여 설명함에 있어, 동일하거나 대응하는 구성요소는 동일한 도면번호를 부여하고 이에 대한 중복되는 설명은 생략하기로 한다.

【0027】영 지식 스나크(zk-SNARKs)는 'zero-knowledge Succinct Non-interactive Argument of Knowledge'의 약자로, 기존의 영 지식 증명을 좀 더 간결하고(succinct) 비 상호적인 환경(non-interactive)에서 적용 가능하도록 변형한 기술이다.

【0028】영 지식 스나크(zk-SNARKs)는 변수(RS)의 구조적 특징에 따라 명제 혹은 프로그램이 변경될 때마다 변수(RS)를 새로 신뢰 설정(Trusted Setup)해야 하는 써킷 특정 영 지식 스나크(circuit-specific zk-SNARKs)와 명제 혹은 프로그램이 변경되어도 이전에 사용된 변수(RS)를 재사용할 수 있는 범용 영 지식 스나크(Universal zk-SNARKs)로 나눌 수 있다. 일반적으로, 명제 및 프로그램이 고정되어 있다면 써킷 특정 영 지식 스나크(circuit-specific zk-SNARKs)가 범용 영 지식 스나크(Universal zk-SNARKs)보다 훨씬 간결하다. 하지만 써킷 특정 영 지식 스나크(circuit-specific zk-SNARKs)는 명제 및 프로그램이 변경될 때마다 변수(RS)도 새롭게 생성되어야 하기 때문에, 증거의 길이가 크고 증명 및 검증이 느린 범용 영 지식 스나크(Universal zk-SNARKs)를 주로 사용하게 된다.

【0029】본 발명은 스택 머신을 위한 간결한 범용 영 지식 스나크(Universal zk-SNARKs)를 제공한다. 본 발명은 스택 머신에 한정되어 일반적인 범용 영 지식 스나크(Universal zk-SNARKs) 보다는 간결하여 블록체인 거래 처리 속도를 개선할 수 있다.

【0030】스택 머신은 스택 기반 가상 머신으로 컴퓨터 시스템을 가상화 하는 에뮬레이터이다. 스택 머신은 연산 명령들이 순차적으로 실행되는 단순한 구조이다. 이러한 스택 머신에서 실행할 수 있는 프로그램이 스마트 컨트랙트이다.

【0031】본 발명은 스마트 컨트랙트의 내용을 효과적으로 검증할 수 있다.

【0032】블록체인은 매번 모든 블록에 있는 스마트 컨트랙트 들의 검증을 진행한다. 이 때 스마트 컨트랙트의 내용 검증을 영 지식 스나크(zk-SNARKs)의 증거 검증으로 대체하면 익명성과 처리속도 면에서 많은 개선이 가능하다. 하지만 스마트 컨트랙트 프로그램은 사용자에 따라 변경될 수 있기 때문에 써킷 특정 영 지식 스나크(circuit-specific zk-SNARKs)은 적합하지 않다.

【0033】스마트 컨트랙트는 단순한 연산 명령 코드(opcode, Operation Code)들이 순차적으로 실행되는 단순한 구조다. 하지만 스택 머신에는 다양한 연산 명령 코드 조합을 가지는 수많은 스마트 컨트랙트 들이 존재할 수 있다. 이 모든 경우의 수 만큼 많은 스마트 컨트랙트를 다 변수(RS)로 셋업 하기에는 쉽지 않다.

【0034】도 1 및 도 2는 본 발명의 일 실시 예에 따른 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템을 설명하기 위한 도면들이다.

【0035】도 1을 참조하면 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 모듈 생성부(100), 모듈 변환부(200), 연결 관계 적용부(300), 증명부(400) 및 검증부(500)을 포함할 수 있다.

【0036】스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 스택 머신 프로그램의 변수(RS, Reference string)를 신뢰 셋업(Trusted Set-up)하여 셋업 파일을 생성한다.

【0037】셋업 파일은 비대화형 시스템이 보안성을 갖도록 하기 위해 증명자와 검증자가 함께 사용하는 참조 파일이다. 스택 머신 프로그램을 위한 범용 영 지

식 스나크 증명 시스템(10)은 스택 머신 프로그램의 정보 및 보안 파라미터 등의 변수(RS)를 암호화되어 셋업 파일에 저장한다.

【0038】 스택 머신은 연산 명령 코드(opcode, Operation Code)를 사전에 정의한 명령어 세트(instruction set)가 있다. 예를 들면 명령어 세트(Instruction set)는 “Instruction set:={ADD, MULT, EQUAL, AND, OR, NAND, …}”과 같은 형태로 연산 명령 코드(opcode)가 정의되어 있다.

【0039】 실제 스택 머신 프로그램은 연산의 수가 훨씬 많고 복잡하지만, 본 발명에서는 이해를 돕기 위해 2개의 AND 연산 명령과 1개의 MULT 연산 명령으로 이루어진 간단한 BITWISE-AND 연산 명령을 예시로 설명하도록 한다.

【0041】 본 발명에서는 스택 머신의 명령어 세트에 정의된 모든 연산 명령 코드(opcode)를 영 지식 증명에 사용하기 위한 써킷 다항식으로 변환하고, 이를 모듈로 칭한다.

【0043】 도 2는 종래의 영 지식 증명에서 사용하는 증명 수행의 예시이다.

【0044】 도 2의 예시와 같이 종래에는 각 모듈의 연산이 참임을 각각 증명하고 모든 연결 관계를 증명하여 스택 머신 프로그램을 참임을 증명하였다. 스택 머신 프로그램의 서킷 다항식은 모듈의 1차원적 순열로 모델링한다. 도 2와 같이 3개의 연산 명령 코드(opcode)에 대응하는 모듈을 사용하여 각각 증명하고, 각 모듈의

모든 연결 관계를 증명하므로 3개의 연산 명령 코드(opcode)에 대해 총 4번의 증명을 수행해야 한다. 종래에는 각 모듈의 연산과 연결 관계가 참임을 각각 증명하므로, 스택 머신 프로그램이 복잡할수록 증거의 크기가 커지고 검증 과정이 복잡하다.

【0045】 반면 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)에서는 스택 머신 프로그램에 사용된 모든 모듈의 연산과 연결관계가 참임을 한 번에 증명할 수 있다.

【0046】 본 발명은 스택 머신 프로그램의 명령어(instructions)의 배열을 프로그램 배열이라 정의하고, 연산 명령 배열은 값을 제외한 연산 명령 만을 배열한 정보로 정의한다.

【0048】 도 3 내지 도5는 본 발명의 일 실시 예에 따른 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템에 생성한 모듈의 예시들이다.

【0049】 도 3은 모듈 생성부(100)가 도 2의 예시에서 사용된 BITWISE-AND 연산 명령 코드(opcode)를 QAP 구조의 써킷 다항식으로 생성한 예시이다.

【0050】 도 3의 예시를 참조하면, 모듈 생성부(100)가 BITWISE-AND 연산 명령 코드 (opcode)의 QAP 써킷 다항식에 포함된 세 개(V, W, Y)의 행렬들의 각각의 열 (column)을 FFT 혹은 NTT를 사용하여 다항식으로 변환함으로써 모듈을 생성한다. 예를 들어, V 행렬의 3번째 열의 다항식 $v_3(X)$는 8개의 이산 데이터

$\{((w_8)^0,1),((w_8)^4,2)\} \cup \{((w_8)^k,0)\}^{7}_{k=1,k\neq 4}$ 를 보간하여 생성한다. 이 때 $(w_n)^k$ 는 써킷 다항식의 근이며 [수학식 1]과 같은 성질을 갖는 어떤 수이다.

【0051】 **【수학식 1】**

써킷 다항식의 근$(roots)=\{\omega_n^k\}_{k=0}^{n-1}$
$(\omega_n)^{n=1}, \sum_{k=0}^{n-1}(\omega_n)^k=0$

【0053】 올바르게 생성된 QAP 구조의 모듈은 [수학식 2]와 같은 특성을 갖는다. [수학식 2]의 *h(x)* 는 비밀 정보를 알고 있는 증명자만이 찾을 수 있다.

【0054】 **【수학식 2】**

$$\left(\sum_i c_i v_i(X)\right) \cdot \left(\sum_i c_i w_i(X)\right) - \left(\sum_i c_i y_i(X)\right) = t(X)h(X)$$

【0055】 모듈이 올바르기 위한 필요충분조건은 [수학식 2]를 만족하는 *h(X)* 가 존재하는 것이다. 여기서, $v_i(x), w_i(x), y_i(x)$ 는 각각 V, W, Y의 i 번째 열의 다항식이며, $c:=[1,a,a_1,a_1,a_2,b,b_1,b_2,c_1,c_2]$ 은 모듈에 사용된 와이어 변수들의 배열이며, c_i 는 c의 i번째 와이어 변수이며,

$t(X):=\prod_{(k=0)}^{(n-1)}(X-\omega_n)^k$ 이다.

【0056】 모듈 생성부(100)는 스택 머신의 명령어 세트에 정의된 모든 연산 명령 코드(opcode)를 영 지식 증명에 사용하기 위한 써킷 다항식으로 변환하여 모듈을 생성할 수 있다. 모듈 생성부(100)는 사전에 정의된 연산 명령 코드(opcode)를 써킷 다항식으로 변환하여 모듈을 생성할 수 있다. 즉, 변환된 써킷 다항식이 모듈이다.

【0057】 모듈 생성부(100)는 각 연산 명령 코드에 대응하는 각각의 써킷 다항식으로 개별 모듈로 생성할 수 있다.

【0058】 모듈 생성부(100)는 연산 명령 코드(opcode)를 써킷 다항식으로 변환할 때 다항식 보간법(Interpolation)을 사용한다. 예를 들면 모듈 생성부(100)는 다항식 보간법을 수행할 때 빠른 푸리엔 변환(FFT, Fast Fourier transform)의 활용하여 써킷 다항식을 변환하여 모듈을 생성할 수 있다. 모듈 생성부(100)는 실수 영역에서는 빠른 푸리에 변환(FFT) 방식을 활용하고, 정수 영역은 Number-theoretic transform (NTT)을 활용할 수 있다.

【0059】 모듈 생성부(100)는 각각의 연산 명령 코드(opcode)를 써킷 다항식으로 변환하고, 이를 암호화 하여 변수(RS)에 기록하고 셋업 파일을 생성할 수 있다.

【0060】 모듈 생성부(100)는 써킷 다항식을 암호화 할 때 부분적 동형 암호화(Partially homomorphic encryption)를 사용할 수 있다. 부분적 동형 암호화는

암호화 연산이 더하기 혹은 곱하기 연산 중 하나에 대하여 선형성을 갖는 암호화이다.

【0061】 모듈 생성부(100)는 스택 머신 프로그램의 모든 모듈을 변수(RS)에 기록할 수 있다. 모델 생성부(100)는 변수(RS)에 모든 모듈을 포함하므로 스택 머신 프로그램의 다양한 조합에도 셋업 파일을 새로 생성할 필요가 없다.

【0062】 도 4 는 모듈 생성부(100)가 생성하는 변수(RS) 셋업 파일의 예시이다.

【0063】 도 4 를 참조하면, 연산 기호 $[?]_G$와 $[?]_H$는 생성자가 각각 G와 H인 부분적 동형 암호 연산을 의미한다.

【0064】 도 4를 참조하면, 모듈 생성부(100)는 신뢰가능한 제3자에게 보안 파라미터 $\alpha,\beta,\gamma,\delta,x$ 를 설정하도록 할 수 있다.

【0065】 도 4를 참조하면, $v_{i,k}^{(u)}$, $w_{i,k}^{(u)}$, $v_{i,k}^{(u)}$, $a_{i,k}^{(u)}$, $z_{i,k}^{(u)}$ 는 각각 다항식 $v_i^{(u)}(X)$, $w_i^{(u)}(X)$, $y_i^{(u)}(X)$, $a_i^{(u)}(X)$, $z_i^{(u)}(X)$의 k차수 항의 계수이다. $v_i^{(u)}(X)$, $w_i^{(u)}(X)$, $y_i^{(u)}(X)$는 u번째 모듈의 써킷 행렬 V, W, Y의 i 번째 열의 다항식이다. $a_i^{(u)}(X)$와 $z_i^{(u)}(X)$는 각각 [수학식 3]과 [수학식 4] 와 같이 정의할 수 있다.

【0066】【수학식 3】

$$a_i^{(u)}(X) := \frac{\beta v_i^{(u)}(X) + \alpha w_i^{(u)}(X) y_i^{(u)}(X)}{\delta} .$$

【0067】【수학식 4】

$$z_i^{(u)}(X) := \frac{\beta v_i^{(u)}(X) + \alpha w_i^{(u)}(X) y_i^{(u)}(X)}{\gamma} .$$

【0069】모듈 생성부(100)는 연산 명령 코드(opcode)로 된 스택 머신 프로그램에서 프로그램 배열과 연산 명령 배열을 추출할 수 있다.

【0070】도 5의 예시를 참조하면, 모듈 생성부(100)는 2 개의 BITWISE-AND 연산 명령과 1개의 MULT 연산 명령을 사용하는 스택 머신 프로그램 예시 $c_5=(c_1 \otimes c_2)(c_3 \otimes c_4)\ with\ c_1=1,\ c_2=3,\ c_3=2,\ c_4=3,\ c_5=2$ 에서 프로그램 배열 ($c_1, c_2\ AND\ c_3, c_4\ AND\ MULT$)과 연산 명령 배열 (*AND AND MULT*)을 추출할 수 있다.

【0072】도 6내지 도 8은 본 발명의 일 실시 예에 따른 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템의 모듈 변환을 설명하기 위한 도면들이다.

【0073】종래의 영 지식 스나크 증명 시스템은 도 3의 예시의 스택 머신 컨트랙트를 증명하기 위해 두 개의 BITWISE-AND와 하나의 MULT 연산 명령 코드(opcode)의 모듈의 올바름을 각각 모두 증명하여야 한다.

【0075】도 6을 참조하면, 모듈 변환부(200)는 각 연산 명령 코드에 대응하여 생성된 각각의 모듈을 통합할 수 있다.

【0076】도 6의 예시를 자세히 설명하면, 모듈 변환부(200)는 모듈 번호 *u=1, 2, 3* 에 대하여 모든 모듈 $p^{(u)}(X):=(\sum_i c_i^{(u)} v_i^{(u)})(\sum_i c_i^{(u)} w_i^{(u)})-(\sum_i c_i^{(u)} y_i^{(u)})$ 들을 통합할 수 있다. 통합된 써킷 다항식은 [수학식 5]와 같다.

【0077】**【수학식 5】**

$$p(X):=\left(\sum_u\sum_i c_i^{(u)} v_i^{(u)}(Xw_n^{-(u)})\right)\cdot\left(\sum_u\sum_i c_i^{(u)} w_i^{(u)}(Xw_n^{-(u)})\right)-\left(\sum_u\sum_i c_i^{(u)} y_i^{(u)}(Xw_n^{-(u)})\right).$$

【0078】이를 통해 증명자는 *p(X)_=t(X)_h(X)*를 만족하는 *h(X)*를 알고 있음을 증명할 수 있다. 다시 말해, 모듈 번호 *u = 1, 2, 3* 에 대하여 모듈

$p^{(u)}(X):=(\sum_i c_i^{(u)} v_i^{(u)})(\sum_i c_i^{(u)} w_i^{(u)})-(\sum_i c_i^{(u)} y_i^{(u)})$ 가 주어졌을때, 증명자는 $p(X)=t(X)h(X)$를 만족하는 $h(X)$ 를 알고 있음을 증명한다.

【0079】 반면, 종래에는 u=1,2,3,에 대하여 $p^{(u)}(X)=t(X)h^{(u)}(X)$를 만족하는 $h^{(u)}(X)$를 모두 알고 있음을 각각 증명해야 한다.

【0081】 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 각 모듈을 통합하여 하나의 증거를 생성할 수 있다.

【0082】 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 각 연산 명령 코드에 대응하는 모듈들을 통합하고 이들 간의 연결 관계를 변수(RS)에서 참조하고 적용한 후 하나의 증거를 생성하여 영 지식 증명을 할 수 있다. 예를 들면 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 3개의 연산 명령 코드를 포함한 스택 머신 프로그램에서 3개의 모듈과 1개의 연결 관계 정보를 통합하여 하나의 증거를 생성하여 증명을 수행할 수 있다. 연결 관계 정보는 각 연산 명령 코드 간의 모든 연결 관계 정보를 포함할 수 있다.

【0083】 즉 실시 예에 따르면, 종래의 기술은 4번 증명을 하는데 반해, 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 1개의 증거를 이용해 1번의 검증을 수행하면 된다. 본 발명에서 설명하는 예시는 연산이 3개 이지만 실제 스택 머신 프로그램은 연산의 수가 훨씬 많기 때문에, 종래의 기술은 증거의

수도 많고 검증도 증거의 수만큼 수행해야 한다.

【0084】 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 연산 명령 코드에 대응하는 각각의 모듈과 각 연산 명령 사이의 연결 정보를 변수(RS)에서 참조하고 통합하여 1개의 증거를 생성하고 검증할 수 있다.

【0086】 도 7 및 도 8 은 본 발명의 일 실시 예에 따른 모듈 변환부(200)가 간단한 스택 머신 프로그램의 모듈을 이동하여 변환시키는 예시들이다.

【0087】 실시 예에 따르면, 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 각 모듈을 연결하여 하나의 증거로 통합하기 위해서는 모듈을 이동시키는 변환(shift) 과정이 필요하다.

【0088】 모듈 변환부(200)는 모듈을 이동시키는 변환 과정을 통해 하나 이상의 연산 명령 코드(opcode)의 각 모듈을 통합할 수 있다. 모듈 변환부(200)는 연상 명령 배열에 따라 연산 명령 코드에 대응되는 개별 모듈을 통합할 수 있다.

【0089】 각 모듈은 변수(RS)로 셋업 파일에 저장되어 있고, 그 값이 암호화되어 저장되어 있기 때문에 변환(shift) 과정을 수행하기 어렵다. 자세히 설명하면 셋업 파일에는 써킷 다항식인 모듈의 원형이 저장된 것이 아니고, 다항식에 비밀값을 집어넣어 계산한 결과를 암호화하여 저장되어 있다. 예를 들면, 다항식 *p(X)*에 어떤 비밀값 x를 이용하여 계산한 결과 p(x)를 암호화한 $[p(x)]_G$ 의 형태로 저장되어 있다. 모듈 $p(X):=p_2X^2+p_1X+p_0$일 때,

$[p(x)]_G = p_2[x^2]_G + p_1[x]_G + p_0$ 로 암호화되어 저장되는 것이다. 즉 모듈은 변수(RS)에 써킷 다항식 원본의 형태로 저장된 것이 아니라 암호화되어 저장되어 있고, 암호화된 출력 값은 공개되어 있지만 암호화 되기 전에 원본인 입력 값은 누구도 확인할 수가 없다. 따라서 종래의 기술은 암호화 되기 전의 입력 값을 알 수 없기 때문에 모듈을 수정 혹은 변환(Shift)할 수 없다.

【0090】 하지만 도 7 을 참조하면, 모듈 변환부(200)는 시프팅 특성(Shifting Property)을 이용해 모듈을 변환(shift)하여 통합할 수 있다. 예를 들면 모듈 변환부(200)는 암호화된 $[p(x)]_G$를 $[p(X-x^{(u)})]_G$와 같이 변환할 수 있다.

【0091】 모델 생성부(100)가 빠른 푸리에 변환(FFT)을 활용해 서킷 다항식을 변환하였기 때문에, 모듈 변환부(200)는 시프팅 특성(Shifting Property)을 이용해 모듈을 변환(shift)할 수 있다.

【0092】 모듈 변환부(200)는 시프팅 특성(Shifting Property)을 적용하여 [수학식 6]의 예와 같이 암호화된 모듈을 복호화 하지 않고 변환(shift)할 수 있다.

【0093】 **【수학식 6】**

$$p(X) := p_2X^2 + p_1X + p_0$$
$$p(X\omega_n^{-1}) = \omega_n^{-2}p_2X^2 + \omega_n^{-1}p_1X + \omega_n^{0}p_0$$
$$[p(x\omega_n^{-1})]_G = \omega_n^{-2}p_2[x^2]_G + \omega_n^{-1}p_1[x^1]_G + \omega_n^{0}p_0[x^0]_G$$

【0095】도 8의 예시를 참조하면, 모듈 변환부(200)는 변수(RS)에 기록된 모듈들을 통합하여 하나의 증거로 생성하기 위해 시프팅 특성(Shifting Property)을 이용한 변환(shift) 과정을 수행할 수 있다. 예를 들면 모듈 변환부(200)는 모듈 $p^{(2)}(X)$ 를 $p^{(2)}(X(\omega_n)^{-u_2})=(\omega_n)^{-2u_2}p_2^{(2)}X^2+(\omega_n)^{-u_2}p_1^{(2)}X^1+(\omega_n)^0p_0^{(2)}$ 로 변환(shift)할 수 있고, 모듈 $p^{(3)}(X)$ 를 $p^{(3)}(X(\omega_n)^{-u_3})=(\omega_n)^{-2u_3}p_2^{(3)}X^2+(\omega_n)^{-u_3}p_1^{(3)}X^1+(\omega_n)^0p_0^{(3)}$ 로 변환할 수 있다. 이때 각 모듈의 변환에 사용되는 근 ω_n^{-k} 에서 양의 정수 k는 스택 머신 프로그램의 연산 명령 배열 순서에 대응하도록 결정한다.

【0097】도 9 및 도 10은 본 발명의 일 실시 예에 따른 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)의 연결 관계 정보 적용을 설명하기 위한 도면들이다.

【0098】연결 관계 적용부(300)는 통합된 모듈에 연결 관계 정보를 적용할 수 있다.

【0099】연결 관계 적용부(300)는 연산 명령 코드(opcode)로 된 스택 머신 프로그램에서 모듈 생성부(100)가 추출한 프로그램 배열과 연산 명령 배열을 이용한다. 다시 도 6의 예시를 참조하면, 모듈 생성부(100)가 2 개의 BITWISE-AND 연산 명령과 1개의 MULT 연산 명령을 사용하는 스택 머신 프로그램 예시 $c_5=(c_1 \otimes c_2)(c_3 \otimes c_4)\ with\ c_1=1,\ c_2=3,\ c_3=2,\ c_4=3,\ c_5=2$ 에서 추출한 프로그램 배열 ($c_1,c_2\ AND\ c_3,c_4\ AND\ MULT$)과 연산 명령 배열 (*AND AND MULT*)을 이용할 수 있다.

【0100】연결 관계 적용부(300)는 프로그램 배열을 참조하여 각 모듈의 입력 와이어와 출력 와이어 간의 연결 관계 정보를 [수학식 7]에 정의된 함수의 형태 $\rho(u,i)$ 로 생성할 수 있다.

【0101】**【수학식 7】**

$$\rho: I_U \times I_m \rightarrow I_{U \times I_m,}$$
$$I_{U:=} 1,2,\cdots,U$$
$$I_{m:=} 1,2,\cdots,m$$

【*0102*】U : 사용된 모듈의 개수

【*0103*】m : 모듈 와이어 변수의 개수의 최대 값

【0105】 만약 와이어 $c_i^{(u)}$ 가 어떤 모듈의 입력 와이어이고 다른 모듈의 출력 와이어 $c_i^{(k)}$ 와 연결되어 있다면 $\rho(u,i)=(k,j)$ 이며, 그렇지 않다면 $\rho(u,i)=(u,i)$이다.

【0107】 도 9의 예시를 참조하여 $\rho(u,i)$를 자세히 설명하면, 세 개의 모듈이 사용된 스택 머신 프로그램의 예시에서, 1번 모듈의 출력 와이어인 $c_3^{(1)}$ 과 2번 모듈의 출력 와이어인 $c_3^{(2)}$ 가 3번 모듈의 입력 와이어들인 $c_1^{(3)}$ 과 $c_2^{(3)}$ 에 각각 연결되었으므로 $\rho(3,1)=(1,1)$이다. 따라서, $\rho(3,2)=(2,1)$, 그리고 $(u,i)\neq(3,1)$와 $(u,i)\neq(3,2)$인 모든 (u,i)에 대하여 $\rho(u,i)=(u,i)$이다.

【0108】 연결 관계 적용부(300)는 기존 통합 모듈의 다항식에 연결 관계 정보 함수 $\rho(u,i)$를 적용하여 새로운 통합 모듈을 생성한다.

【0109】 연결 관계 적용부(300)가 생성하는 새로운 통합 모듈의 와이어 배열은 [수학식 8]과 같다.

【0110】【수학식 8】

새로운 통합 모듈의 와이어 배열$c:=[c_i^{(u)}]_{(u,i)\in img(\rho)}$.

【0111】 $c_i^{(u)}$: : 기존 통합 모듈의 와이어 배열

【0112】 $img(\rho)$ 함수 ρ 의 치역(image)

【0114】연결 관계 적용부(300)가 생성하는 새로운 통합 모듈의 다항식은 [수학식 9]와 같다. 모든 $(u,i)\in dom(\rho)$에 대하여, $v_i^{(u)}(X)$와 $w_i^{(u)}(X), y_i^{(u)}(X)$를 기존 통합 모듈 변환(shifting)된 다항식이라 할 때, 새로운 통합 모듈의 다항식은 모든 $(k,j)\in dom(\rho)$에 대하여 [수학식 9]와 같다.

【0115】【수학식 9】

$$v_j^{(k)}(X)\leftarrow\Sigma_{(u,i)\in\rho^{-1}(\{(k,j)\})}v_u^{(i)}(X),$$
$$w_j^{(k)}(X)\leftarrow\Sigma_{(u,i)\in\rho^{-1}(\{(k,j)\})}w_u^{(i)}(X),$$
$$y_j^{(k)}(X)\leftarrow\Sigma_{(u,i)\in\rho^{-1}(\{(k,j)\})}y_u^{(i)}(X)$$

【0116】 $dom(\rho)$: 함수 ρ의 정의역(domain)

【0118】 도 10은, 도9의 예시로 사용된 스택 머신 프로그램과 연결 관계 정보 함수 ρ대하여, 연결 관계 적용부(300)가 기존 통합 모듈에 연결 관계를 적용하여 새로운 통합 모듈을 생성하는 예시이다.

【0119】 도 10의 예시를 참조하면, 연결 관계 적용부(300)에 의해 생성된 새로운 통합 모듈의 다항식과 와이어의 개수가 기존 통합 모듈의 다항식과 와이어의 개수보다 더 적을 수 있다. 이는 연결 관계 적용부(300)가 ρ를 참조하여, 기존 통합 모듈에서 서로 연결관계가 있는 다수의 다항식들을 모두 더함으로써 새로운 통합 모듈의 하나의 다항식을 생성할 수 있기 때문이다.

【0121】 증명부(400)는 각 모듈과 연결 관계 정보를 모두 통합하여 한 번에 증명할 수 있는 하나의 증거를 생성할 수 있다.

【0122】 대부분의 영 지식 스나크(ZK-SNARKs)는 암호화 방식으로 타원 곡선 암호(ECC, Elliptic Curve Cryptography) 알고리즘을 이용한다. 타원 곡선 암호(ECC, Elliptic Curve Cryptography) 알고리즘은 선형성을 가진다.

【0123】 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 푸리에 변환을 이용한 다항식 보간법을 사용하므로 모듈 변환과 연결 관계 정보 적

용 또한 선형 연산일 수 있다.

【0124】타원 곡선 암호화(ECC)는 선형성을 가짐으로 모듈 변환도 선형 연산이고, 연결 관계 정보 적용 또한 선형 연산이므로, 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 선형성을 이용해 복호화 없이 모듈 변환과 연결 관계 정보 적용이 가능하다.

【0125】스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 타원 곡선 암호화(ECC, Elliptic Curve Cryptography)를 이용해 복호화 없이 모듈을 변환(shift)하여 통합하고, 연결 관계가 있는 다항식을 더하는 방식으로 적용할 수 있다.

【0126】증명부(400)는 영 지식 증명을 위한 증거를 생성할 수 있다. 증명부(400)는 각 모듈 변환과 연결 관계 정보를 적용하여 하나의 영 지식 증거를 생성할 수 있다. 자세히 설명하면 증명부(400)는 스택 머신 프로그램과 변수가 암호화된 셋업 파일, 비공개의 스택 머신 프로그램의 입력 값과 공개된 스택 머신 프로그램의 출력 값을 이용해 증명하고, 영 지식 증거를 생성할 수 있다.

【0127】증명부(400)는 먼저 비공개 보안 파라미터 무작위 정수 r, s를 생성한 후 참임을 증명하는 3개의 증거 파일 $[A]_G,[B]_G,[C]_G$를 생성한다. 3개의 증거 파일은 하나의 증거 셋(SET)이다.

【0128】 【수학식 10】

$$[A]_G := [\alpha]_G + \sum_{(u=0)}^{(U-1)} \sum_{(i=0)}^{(m-1)} \sum_{(k=0)}^{(n-1)} c_i^{(u)} v_{i,k}^{(u)} \omega_n^{-uk} [x^k]_G + r[\delta]_G,$$
$$[B]_H := [\beta]_H + \sum_{(u=0)}^{(U-1)} \sum_{(i=0)}^{(m-1)} \sum_{(k=0)}^{(n-1)} c_i^{(u)} w_{i,k}^{(u)} \omega_n^{-uk} [x^k]_G + s[\delta]_H,$$
$$[C]_G := \sum_{(u=0)}^{(U-3)} \sum_{(i=0)}^{(m-1)} \sum_{(k=0)}^{(n-1)} c_i^{(u)} \omega_n^{-uk} [a_{(i,k)}^{(u)} x^k]_G$$
$$+ \sum_{k=0}^{n-2} h_k \left[\frac{x^k t(x)}{\delta} \right]_G + s[A]_G + r[B]_G - rs[\delta]_G$$

【0130】 검증부(500)는 생성된 영 지식 증거를 검증한다. 자세히 설명하면 검증부(500)는 스택 머신 프로그램과 변수(RS)가 암호화된 셋업 파일, 공개된 스택 머신 프로그램의 출력 값을 이용해 영 지식 증거가 참인지 거짓인지 검증할 수 있다.

【0131】 검증부(500)는 먼저 생성된 증거의 데이터들이 모두 타원곡선상의 점이 맞는지 확인한 후, 증거를 사용하여 $LHS = [A]_G \cdot [B]_H$를 계산한다.

【0132】 검증부(500)는 증거와 변수(RS)를 사용하여 [수학식 10]의 *RHS*를 산출한다.

【0133】 【수학식 11】

$$RHS := [\alpha]_G \cdot [\beta]_H + [C]_G \cdot [\delta]_H$$
$$+ \left(\sum_{u=U-2}^{U-1} \sum_{i=0}^{m-1} \sum_{k=0}^{n-1} c_i^{(u)} \omega_n^{-uk} \left[z_{i,k}^{(u)} x^k \right]_G \right) \cdot [\gamma]_H.$$

【0134】검증부(400)는 *LHS=RHS* 가 성립하면 영 지식 증거를 참으로 판단하고, 등호가 성립하지 않으면 영 지식 증거를 거짓으로 판단할 수 있다.

【0136】도 11은 본 발명의 일 실시예에 따른 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 방법을 도시한 도면이다. 이하 설명하는 과정은 각 단계에서 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템을 구성하는 각 기능부가 수행하는 과정이나, 본 발명의 간결하고 명확한 설명을 위해 각 단계의 주체를 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템으로 통칭하도록 한다.

【0137】도11를 참조하면, S1101 단계에서 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 각각의 연산 명령 코드를 모듈로 생성한다. 예를 들면 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 다항식 보간법을 이용하여 연산 명령 코드별로 써킷 다항식으로 변환하여 모듈로 생성할 수 있다.

【0138】S1102 단계에서 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 각 모듈을 통합할 수 있다. 예를 들면 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 개별 모듈을 변환(shift)하여 통합할 수 있다.

【0139】S1103 단계에서 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 각 모듈의 연결 관계 정보를 적용할 수 있다. 예를 들면 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 각각의 연산 명령 코드에 대응되는 개별 모듈 통합하고 연결 관계 정보를 적용하여 최적화된 새로운 통합 모듈을 생성 할 수 있다.

【0140】S1104 단계에서 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 각 모듈 변환과 연결 관계 정보를 적용하여 하나의 영 지식 증거를 생성한다. 자세히 설명하면 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 스택 머신 프로그램과 변수가 암호화된 셋업 파일, 비공개의 스택 머신 프로그램의 입력 값과 공개된 스택 머신 프로그램의 출력 값을 이용해 증명하고, 영 지식 증거를 생성할 수 있다.

【0141】S1105 단계에서 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 영 지식 증거를 검증한다. 자세히 설명하면 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템(10)은 스택 머신 프로그램과 변수가 암호화된 셋업 파일, 공개된 스택 머신 프로그램의 출력 값, 영 지식 증거를 영 지식 증거가 참인지 거짓인지 검증할 수 있다.

【0143】상술한 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 방법은 컴퓨터가 읽을 수 있는 매체 상에 컴퓨터가 읽을 수 있는 코드로 구현될 수 있

다. 상기 컴퓨터로 읽을 수 있는 기록 매체는, 예를 들어 이동형 기록 매체(CD, DVD, 블루레이 디스크, USB 저장 장치, 이동식 하드 디스크)이거나, 고정식 기록 매체(ROM, RAM, 컴퓨터 구비형 하드 디스크)일 수 있다. 상기 컴퓨터로 읽을 수 있는 기록 매체에 기록된 상기 컴퓨터 프로그램은 인터넷 등의 네트워크를 통하여 다른 컴퓨팅 장치에 전송되어 상기 다른 컴퓨팅 장치에 설치될 수 있고, 이로써 상기 다른 컴퓨팅 장치에서 사용될 수 있다.

【0144】이상에서, 본 발명의 실시 예를 구성하는 모든 구성 요소들이 하나로 결합되거나 결합되어 동작하는 것으로 설명되었다고 해서, 본 발명이 반드시 이러한 실시 예에 한정되는 것은 아니다. 즉, 본 발명의 목적 범위안에서라면, 그 모든 구성요소들이 하나 이상으로 선택적으로 결합하여 동작할 수도 있다.

【0145】도면에서 동작들이 특정한 순서로 도시되어 있지만, 반드시 동작들이 도시된 특정한 순서로 또는 순차적 순서로 실행되어야만 하거나 또는 모든 도시된 동작들이 실행되어야만 원하는 결과를 얻을 수 있는 것으로 이해되어서는 안 된다. 특정 상황에서는, 멀티태스킹 및 병렬 처리가 유리할 수도 있다. 더욱이, 위에 설명한 실시 예 들에서 다양한 구성들의 분리는 그러한 분리가 반드시 필요한 것으로 이해되어서는 안 되고, 설명된 프로그램 컴포넌트들 및 시스템들은 일반적으로 단일 소프트웨어 제품으로 함께 통합되거나 다수의 소프트웨어 제품으로 패키지 될 수 있음을 이해하여야 한다.

【0146】이제까지 본 발명에 대하여 그 실시 예들을 중심으로 살펴보았다. 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본

질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 개시된 실시 예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 본 발명의 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

【부호의 설명】

【0148】10: 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템

100: 모듈 생성부

200: 모듈 변환부

300: 연결 관계 적용부

400: 증명부

500: 검증부

【청구범위】

【청구항 1】

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템에 있어서,

사전에 정의된 연산 명령 코드를 써킷 다항식으로 변환하여 모듈을 생성하는 모듈 생성부;

하나 이상의 상기 모듈을 통합하는 모듈 변환부; 및

상기 모듈의 연결 관계 정보를 적용하는 연결 관계 적용부를 포함하는

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템.

【청구항 2】

제1항에 있어서,

상기 모듈 생성부는

상기 연산 명령 코드에 대응하는 각각의 써킷 다항식으로 개별 모듈을 생성하는

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템.

【청구항 3】

제2 항에 있어서,

상기 써킷 다항식을 암호화하여 셋업 파일을 생성하는

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템.

【청구항 4】

제1항에 있어서,

상기 모듈 변환부는

연산 명령 배열에 따라 상기 연산 명령 코드에 대응되는 개별 모듈을 통합하는

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템.

【청구항 5】

제1항에 있어서,

상기 연결 관계 적용부는

프로그램 배열을 참조하여 각 모듈의 입력 와이어와 출력 와이어 간의 연결 관계 정보를 적용하는

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템.

【청구항 6】

제1 항에 있어서,

영 지식 증명을 위한 증거를 생성하는 증명부; 및

상기 증거를 검증하는 검증부를 더 포함하는

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템.

【청구항 7】

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 시스템이 수행하는 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 방법에 있어서,

사전에 정의된 연산 명령 코드를 써킷 다항식으로 변환하여 모듈을 생성하는 단계;

하나 이상의 상기 모듈을 통합하는 단계; 및

상기 모듈의 연결 관계 정보를 적용하는 단계를 포함하는

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 방법.

【청구항 8】

제7항에 있어서,

상기 사전에 정의된 연산 명령 코드를 써킷 다항식으로 변환하여 모듈을 생성하는 단계는

상기 연산 명령 코드를 개별로 각각 상기 써킷 다항식으로 변환하는

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 방법.

【청구항 9】

제8항에 있어서,

상기 써킷 다항식을 암호화하여 셋업 파일을 생성하는

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 방법.

【청구항 10】

제7항에 있어서,

상기 하나 이상의 상기 모듈을 통합하는 단계는

연산 명령 배열에 따라 상기 연산 명령 코드에 대응되는 개별 모듈을 통합하는

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 방법.

【청구항 11】

제7항에 있어서,

상기 모듈의 연결 관계 정보를 적용하는 단계는

프로그램 배열을 참조하여 각 모듈의 입력 와이어와 출력 와이어 간의 연결

관계 정보를 적용하는

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 방법.

【청구항 12】

제7항에 있어서,

영 지식 증명을 위한 증거를 생성하는 단계; 및

상기 증거를 검증하는 단계를 더 포함하는

스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 방법.

【청구항 13】

제7항 내지 제12항의 스택 머신 프로그램을 위한 범용 영 지식 스나크 증명 방법 중 어느 하나를 실행하는 컴퓨터가 판독 가능한 기록매체에 기록된 컴퓨터 프로그램.

【요약서】

【요약】

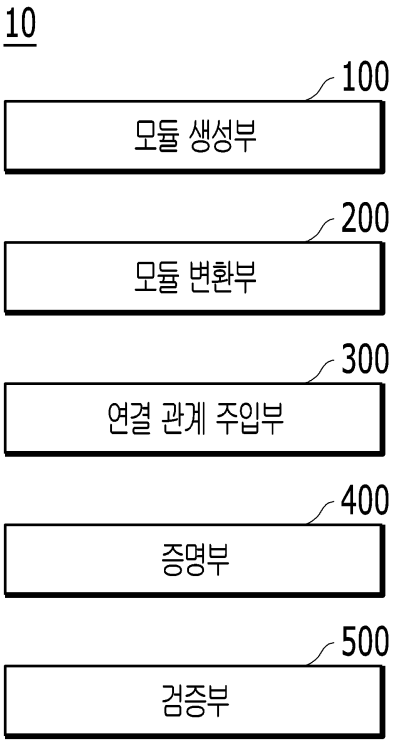
본 발명은 영 지식 증명 기술에 관한 것으로, 더욱 상세하게는 범용 스택 머신 프로그램을 위한 간결하고 비 상호적인 영 지식 증명 시스템 및 방법에 대한 것이다. 본 발명의 일 실시 예에 따르면, 스택 머신의 단순한 구조적 특징을 이용하여 스택 머신 프로그램의 연산 증명에 한정하여 범용적이면서 간결한 영 지식 스나크로 블록 체인의 스마트 컨트랙트에 적용 될 경우 거래처리 속도를 높일 수 있다.

【대표도】

도 1

【도면】

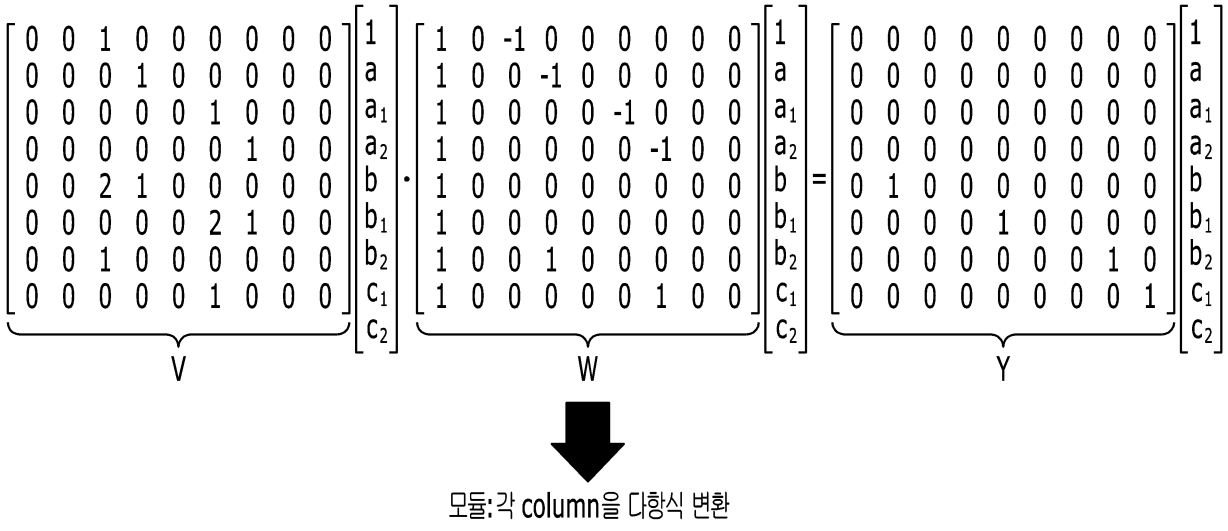
【도 1】



【도 2】

$$※p^{(u)}(X):=\left(\sum_{i\in I_u} v_i^{(u)}(X)\right)\left(\sum_{i\in I_u} w_i^{(u)}(X)\right)-\sum_{i\in I_u} y_i^{(u)}(X)$$

【도 3】



【도 4】

$$\sigma_G := \left(\begin{array}{l} [\alpha]_G, [\beta]_G, [\delta]_G, \left\{ [\omega^{i(n-1)} x^i]_G \right\}_{i=0}^{n-1}, \\ \left\{ \left\{ \left\{ \left[v_{i,k}^{(u)} x^{\bar{i}} \right]_G \right\}_{i=0}^{n-1} \right\}_{i=0}^{m_u-1} \right\}_{u=0}^{U-1}, \\ \left\{ \left\{ \left\{ \left[w_{i,k}^{(u)} x^{\bar{i}} \right]_G \right\}_{i=0}^{n-1} \right\}_{i=0}^{m_u-1} \right\}_{u=0}^{U-1}, \\ \left\{ \left\{ \left\{ \left[z_{i,k}^{(u)} x^{\bar{k}} \right]_G \right\}_{k=0}^{n-1} \right\}_{i=0}^{m_u-1} \right\}_{u=U-2}^{U-1}, \\ \left\{ \left\{ \left\{ \left[a_{i,k}^{(u)} x^{\bar{k}} \right]_G \right\}_{k=0}^{n-1} \right\}_{i=0}^{m_u-1} \right\}_{u=0}^{U-3}, \\ \left\{ \left[\frac{x^i t(x)}{\delta} \right]_G \right\}_{i=0}^{n-2} \end{array} \right),$$

$$\sigma_G := \left([\beta]_H, [\gamma]_H, [\delta]_H, \left\{ [x^i]_H \right\}_{i=0}^{n-1} \right),$$

$$※ z_i^{(u)}(x) := \frac{\beta v_i^{(u)}(x) + \alpha w_i^{(u)}(x) + y_i^{(u)}(x)}{\gamma} = \sum_{k=0}^{n-1} z_{i,k}^{(u)} x^k$$

$$※ a_i^{(u)}(x) := \frac{\beta v_i^{(u)}(x) + \alpha w_i^{(u)}(x) + y_i^{(u)}(x)}{\delta} = \sum_{k=0}^{n-1} a_{i,k}^{(u)} x^k$$

【도 5】

스택 머신 프로그램	$c_5=(c_1 \otimes c_2)(c_3 \otimes c_4)$ with $c_1=1$, $c_2=3$, $c_3=2$, $c_4=3$, $c_5=2$
프로그램 배열	($c_1 c_2$ AND $c_3 c_4$ AND MULT)
연산 명령 배열	(AND AND MULT)
비공개 입력	(c_1 c_2 c_3 c_4 c_6 c_7), 여기서 $c_6=c_1 \otimes c_2$, $c_7=c_3 \otimes c_4$
공개 출력	(c_5)

【도 6】

$$※\ p^{(u)}(X) \coloneqq \left(\sum_{i\in I_u} v_i^{(u)}(X)\right)\left(\sum_{i\in I_u} w_i^{(u)}(X)\right) - \sum_{i\in I_u} y_i^{(u)}(X)$$

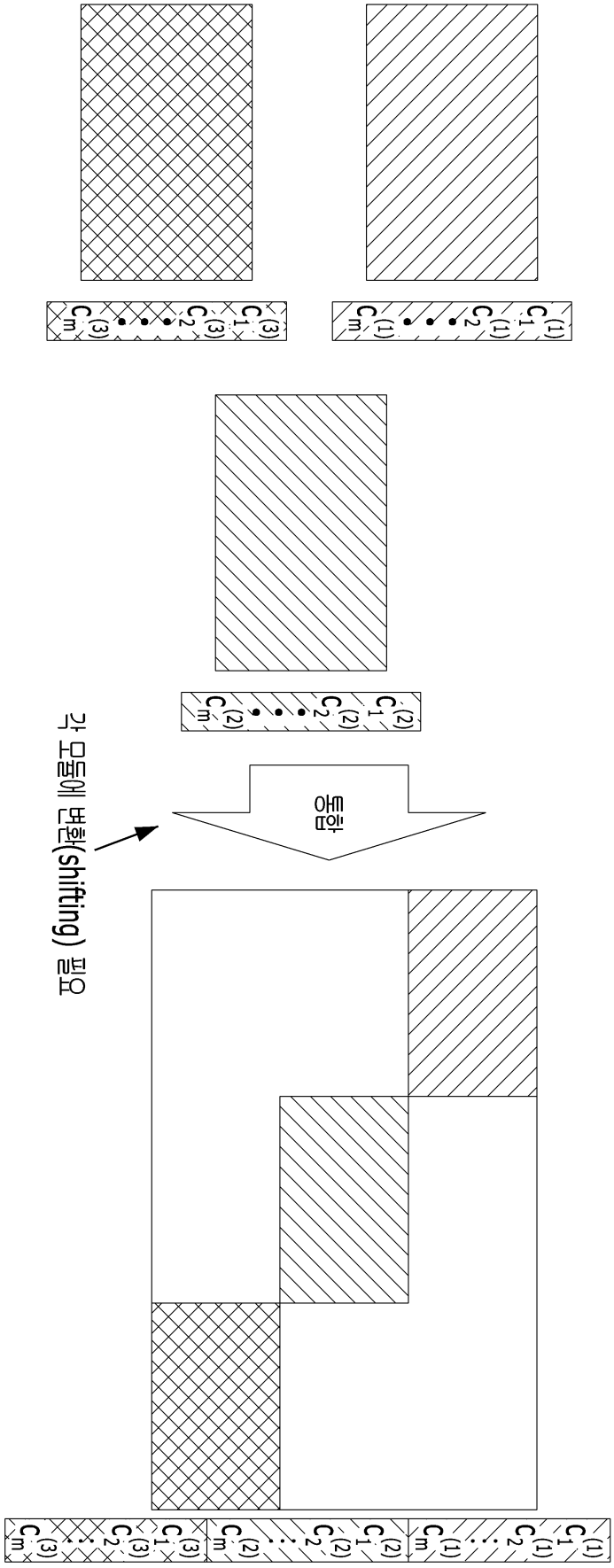
$$\begin{cases} p^{(1)}(X) = h^{(1)}(X)\,t^{(1)}(X) \text{ for AND(1)} \\ p^{(2)}(X) = h^{(2)}(X)\,t^{(2)}(X) \text{ for AND(2)} \\ p^{(3)}(X) = h^{(3)}(X)\,t^{(3)}(X) \text{ for AND(3)} \end{cases}$$

통합

$$p(X) = h(X)t(X) \quad \text{for all modules}$$

$$※\ p^{(u)}(X) \coloneqq \left(\sum_u \sum_i c_i^{(u)} v_i^{(u)}(X\omega_n^{-u_i})\right)\cdot\left(\sum_u \sum_i c_i^{(u)} w_i^{(u)}(X\omega_n^{-u_i})\right) - \left(\sum_u \sum_i c_i^{(u)} y_i^{(u)}(X\omega_n^{-u_i})\right)$$

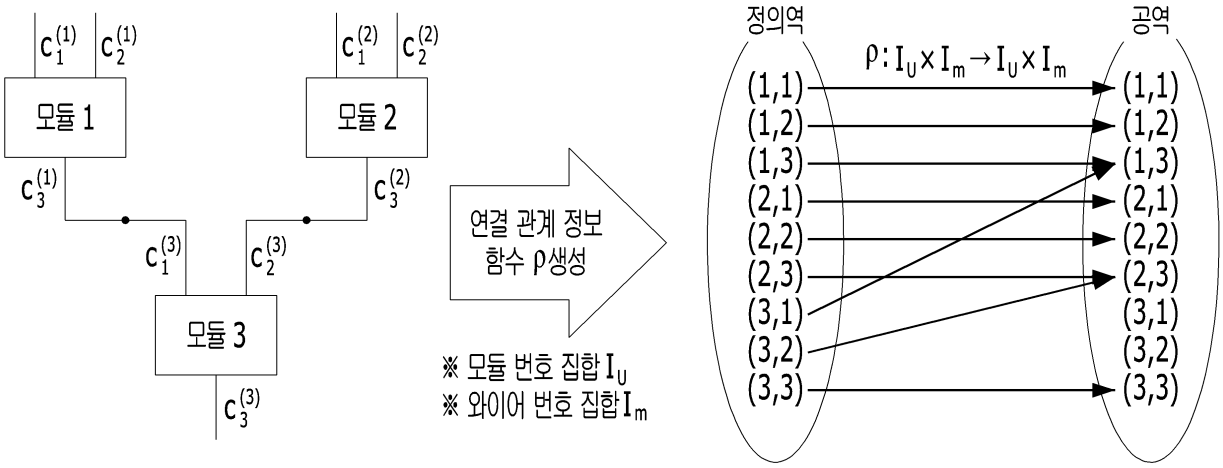
【도 7】


각 모듈에 변화(shifting) 필요
합성

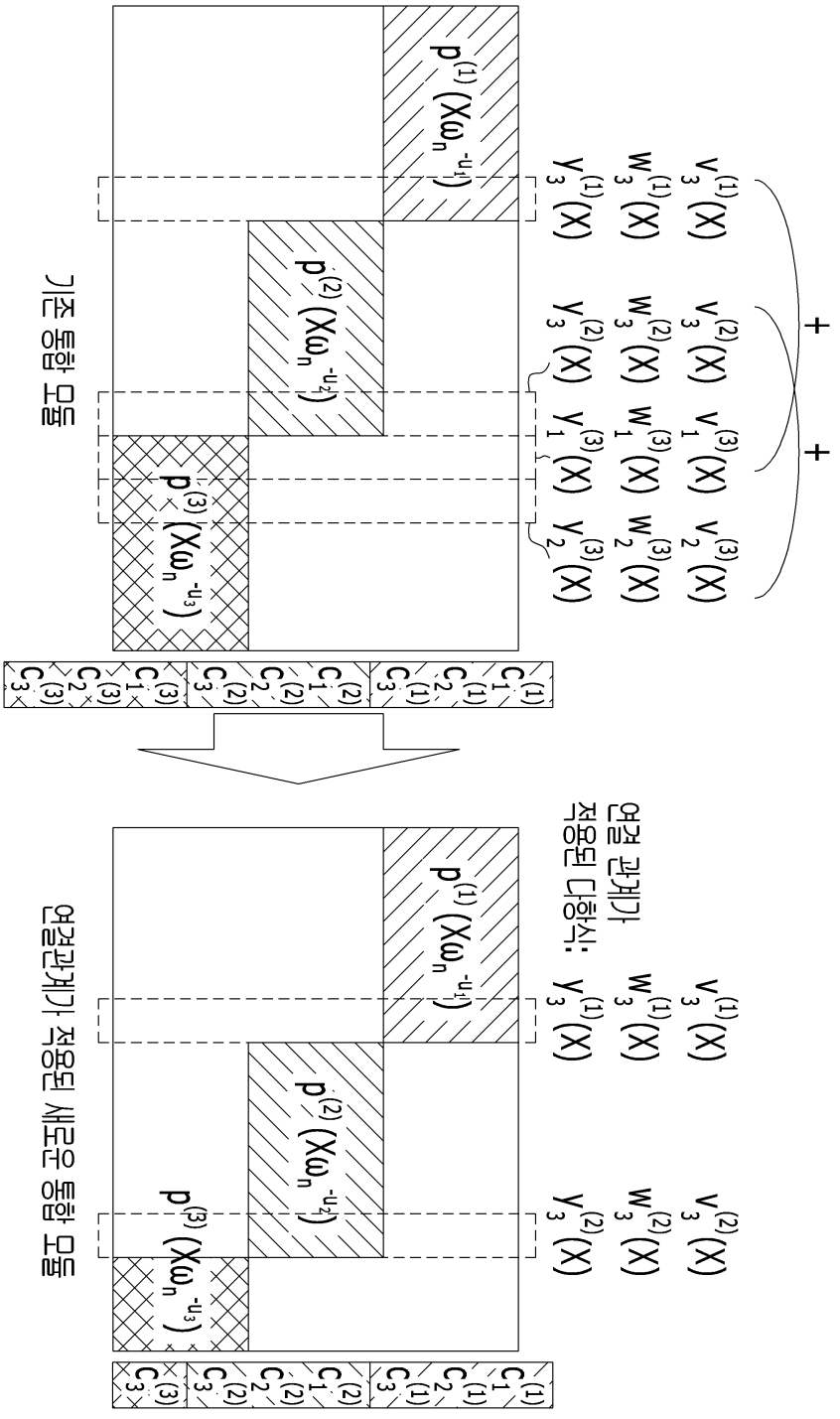
【도 8】

RS에 기록된
ADD(2) 모듈
$p^{(2)}(X)$
$p^{(2)}(X(\omega_n)^{-u_2})$
$=(\omega_n)^{-2u_2}\,p_2^{(2)}X^2+(\omega_n)^{-u_2}\,p_1^{(2)}X^1+(\omega_n)^0\,p_0^{(2)}$
Shifting
새로운 프로그램에
적용 될 변환된
ADD(2) 모듈
$p^{(2)}(X(\omega_n)^{-u2})$
RS에 기록된
MULT(3) 모듈
$p^{(3)}(X)$
$p^{(3)}(X(\omega_n)^{-u_3})$
$=(\omega_n)^{-2u_3}\,p_2^{(3)}X^2+(\omega_n)^{-u_3}\,p_1^{(3)}X^1+(\omega_n)^0\,p_0^{(3)}$
Shifting
새로운 프로그램에
적용 될 변환된
MULT(3) 모듈
$p^{(3)}(X(\omega_n)^{-u3})$

【도 9】


$c_1^{(1)}$
$c_2^{(1)}$
모듈 1
$c_3^{(1)}$
$c_1^{(2)}$
$c_2^{(2)}$
모듈 2
$c_3^{(2)}$
$c_1^{(3)}$
$c_2^{(3)}$
모듈 3
$c_3^{(3)}$
연결 관계 정보
함수 ρ생성
※ 모듈 번호 집합 I_U
※ 와이어 번호 집합 I_m
정의역
공역
$\rho: I_U \times I_m \rightarrow I_U \times I_m$
(1,1)
(1,2)
(1,3)
(2,1)
(2,2)
(2,3)
(3,1)
(3,2)
(3,3)
(1,1)
(1,2)
(1,3)
(2,1)
(2,2)
(2,3)
(3,1)
(3,2)
(3,3)

【도 10】



【도 11】