



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0096335
(43) 공개일자 2025년06월27일

(51) 국제특허분류(Int. Cl.)

G06T 1/00 (2006.01) G06F 21/16 (2018.01)
G06Q 20/36 (2012.01) G06T 7/10 (2021.01)
G06T 7/73 (2017.01)

(52) CPC특허분류

G06T 1/0021 (2013.01)
G06F 21/16 (2023.08)

(21) 출원번호 10-2023-0187498

(22) 출원일자 2023년12월20일

심사청구일자 2023년12월20일

(71) 출원인

서강대학교산학협력단

서울특별시 마포구 백범로 35 (신수동, 서강대학교)

(72) 발명자

박수용

서울특별시 마포구 독막로18길 5, 101동 1001호
(상수동, 상수두산위브아파트)

한윤경

경기도 구리시 응달말로52번길 10, 101동 1404호
(인창동, 정은스카이빌)

서중원

서울특별시 서대문구 수색로8길 11, 201호 (북가좌동, 한영빌라)

(74) 대리인

김중선, 이형석

전체 청구항 수 : 총 8 항

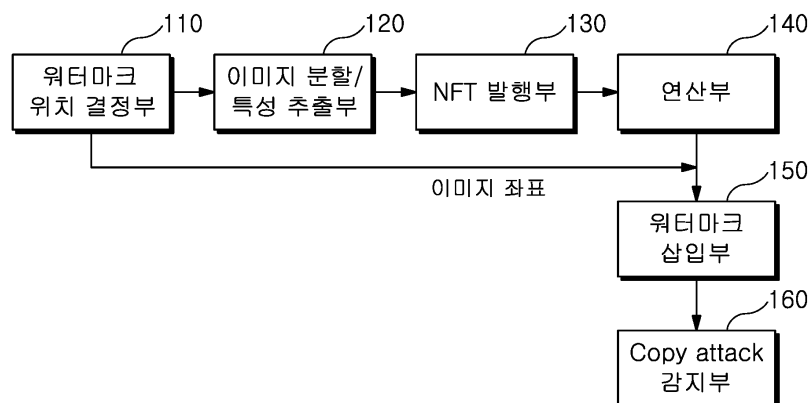
(54) 발명의 명칭 **아이템 위변조 방지를 위한 NFT 기반의 워터마킹 장치 및 방법**

(57) 요약

본 발명은 가성 양성률 및 가성 음성률이 존재하지 않는 Copy Attack을 막을 수 있는 NFT 기반의 워터마킹 장치 및 방법에 관한 것이다. 본 발명의 워터마킹 장치는, 워터마크 이미지 좌표를 결정하는 워터마크 위치 결정부, 이미지를 분할하고, 분할된 이미지에서 각각 이미지 특성을 추출하여 해싱한 후, 각 해시 값을 리스트 형태로 저장하는 이미지 분할 및 특성 추출부, 상기 이미지의 저작권 관련 정보를 해싱하고, 해시 값을 NFT ID로 설정하여 NFT를 발행하는 NFT 발행부, 상기 NFT ID와 이미지 해시 값(ImageHash)을 XOR 연산하여 워터마크 값을 계산하는 연산부, XOR 연산에 의해 계산된 워터마크 값을 상기 워터마크 위치 결정부에 의해 정해진 좌표에 삽입하는 워터마크 삽입부 및 상기 워터마크 값과 이미지 해시 값을 XOR 연산하여 Copy attack을 감지하는 Copy attack 감지부를 포함하여 구성된다.

대표도 - 도1

100



(52) CPC특허분류

G06Q 20/36 (2013.01)

G06T 7/10 (2021.01)

G06T 7/73 (2017.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	1375027550
과제번호	00219237
부처명	문화체육관광부
과제관리(전문)기관명	한국콘텐츠진흥원
연구사업명	소프트웨어저작권연구개발
연구과제명	메타버스에서 저작권 보호 및 이용 활성화 기술 개발
기 여 율	1/3
과제수행기관명	서강대학교산학협력단
연구기간	2023.04.01 ~ 2023.12.31

이 발명을 지원한 국가연구개발사업

과제고유번호	1711179357
과제번호	00156318
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보통신방송혁신인재양성
연구과제명	메타버스 융합대학원(서강대학교)
기 여 율	1/3
과제수행기관명	서강대학교산학협력단
연구기간	2023.01.01 ~ 2023.12.31

이 발명을 지원한 국가연구개발사업

과제고유번호	1711198535
과제번호	00259099
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보통신방송혁신인재양성
연구과제명	웹 3.0을 위한 오픈 플랫폼 기술 개발
기 여 율	1/3
과제수행기관명	서강대학교산학협력단
연구기간	2023.07.01 ~ 2023.12.31

명세서

청구범위

청구항 1

워터마크 이미지 좌표를 결정하는 워터마크 위치 결정부;

이미지를 분할하고, 분할된 이미지에서 각각 이미지 특성을 추출하여 해싱한 후, 각 해시 값을 리스트 형태로 저장하는 이미지 분할 및 특성 추출부;

상기 이미지의 저작권 관련 정보를 해싱하고, 해시 값을 NFT ID로 설정하여 NFT를 발행하는 NFT 발행부;

상기 NFT ID와 이미지 해시 값(ImageHash)을 XOR 연산하여 워터마크 값을 계산하는 연산부;

XOR 연산에 의해 계산된 워터마크 값을 상기 워터마크 위치 결정부에 의해 정해진 좌표에 삽입하는 워터마크 삽입부; 및

상기 워터마크 값과 이미지 해시 값을 XOR 연산하여 Copy attack를 감지하는 Copy attack 감지부를 포함하여 구성되는 것을 특징으로 하는, NFT 기반의 워터마킹 장치.

청구항 2

제1항에 있어서,

상기 워터마크 위치 결정부는,

개인 키를 해싱하는 해싱부;

상기 해싱부에 의해 해싱된 해시 값을 분할하는 분할부;

상기 분할부에 의해 분할된 16진수 해시 값인 제1 값과 제2 값을 각각 10진수로 변환하는 변환부; 및

상기 변환된 10진수인 제1-1 값과 제2-1값을 각각 이미지 크기로 모듈러 연산하여 워터마크 좌표를 제공하는 좌표 제공부를 포함하는, NFT 기반의 워터마킹 장치.

청구항 3

제1항에 있어서,

상기 저작권 관련 정보는,

이미지 생성 날짜, 작가 이름, 저장 위치 정보 중 적어도 하나를 포함하는, NFT 기반의 워터마킹 장치.

청구항 4

제1항에 있어서,

상기 연산부는,

XOR 연산하여 계산된 워터마크 값과 이미지 해시 값을 다시 XOR 연산하면 기 발급된 NFT ID를 계산하는, NFT 기반의 워터마킹 장치.

청구항 5

제1항에 있어서,

상기 Copy attack 감지부는,

원본 이미지와 copy attack 이미지의 NFT ID를 비교하여 copy attack 여부를 감지하는, NFT 기반의 워터마킹 장치.

청구항 6

개인키 해싱을 통해 워터마크가 삽입될 이미지 좌표를 결정하는 제1 단계;

이미지를 분할하고, 분할된 각 이미지 영역의 특성을 추출하는 제2 단계;

상기 각 이미지 영역의 저작권 관련 정보를 해싱하고, 해시 값을 NFT ID로 설정하여 NFT를 발행하는 제3 단계;

NFT ID와 이미지 해시 값을 XOR 연산하여 워터마크 값을 계산하는 제4 단계;

상기 워터마크 값을 상기 이미지 좌표에 삽입하는 제5 단계; 및

상기 워터마크 값과 이미지 해시 값을 XOR 연산하여 Copy attack을 감지하는 제6 단계를 포함하여 수행하는 것을 특징으로 하는, NFT 기반의 워터마킹 방법.

청구항 7

제6항에 있어서,

상기 제1 단계는,

개인 키를 해싱하는 단계;

상기 해싱된 해시 값을 X 좌표 값 계산용 제1 값과 Y 좌표 값 계산용 제2 값으로 분할하는 단계;

상기 분할된 제1 값과 제2 값을 10진수인 제1-1 값과 제2-1값으로 변환하는 단계; 및

상기 제1-1 값과 제2-1값을 각각 이미지 크기로 모듈러 연산하여 상기 이미지 좌표를 계산하는 단계를 포함하여 수행하는, NFT 기반의 워터마킹 방법.

청구항 8

제6항에 있어서,

상기 제6 단계의 Copy attack 감지는,

NFT ID의 계산 결과를 기초로 하여 수행하는, NFT 기반의 워터마킹 방법.

발명의 설명

기술 분야

[0001] 개시된 발명은 가성 양성물 및 가성 음성물이 존재하지 않는 Copy Attack을 막을 수 있는 NFT 기반의 워터마킹 장치 및 방법에 관한 것이다.

배경 기술

[0002] 메타버스는 사용자가 아바타를 생성하여 다양한 활동을 수행할 수 있는 가상 환경으로, 새로운 형태의 콘텐츠 제작 및 판매 방식 등을 제공할 수 있어 실제 세계와 가상 세계의 연계를 통해 새로운 경제적, 사회적 가치를 창출하고 있다. 이에 메타버스 내의 디지털 콘텐츠는 지속적으로 증가하고 있다.

[0003] 그러나 메타버스 내에서 이러한 디지털 콘텐츠의 증가와 함께 저작권 침해 사례들이 증가하고 있으며, 이에 메타버스 내 저작권 보호의 필요성이 커지고 있다.

[0004] 디지털 저작권 보호 기술(DRM)은 디지털 콘텐츠의 무단 사용과 배포를 방지하기 위해 개발된 기술이며, 워터마킹 기술은 DRM의 구성 요소 중 하나이다. 이러한 디지털 워터마킹 기술은 암호화, 라이선스 관리, 디지털 워터마킹, 디지털 서명 등 다양한 보안 메커니즘을 통해 콘텐츠 보호에 중요한 역할을 제공한다.

[0005] 그러나 디지털 워터마킹 기술은 여러 한계와 취약성을 가지고 있다. 예를 들어 워터마킹 된 콘텐츠는 다양한 공격 방법에 의해 효력을 잃을 위험이 있다. 특히 워터마크를 불법적으로 복제하는 Copy Attack이라는 공격기법에 취약하다. Copy Attack으로 워터마크 자체를 불법 복제하면 원본과 복제본을 구분하는 것이 어렵게 된다. 이는 워터마크의 기본 목적인 콘텐츠 소유권 보호와 무단 복제 추적을 어렵게 하며, 기술의 보안성을 저하시키는 원인이 된다.

[0006] 이에 워터마크 Copy Attack과 관련된 방안들이 제안되기도 했다. 예를 들면 해밍 디스턴스나 유사도 비교 방법

이 있다. 하지만 상기 방법들은 임계값 설정에 따라서 가성 양성률(false positive rate) 및 가성 음성률(false negative rate)의 가능성이 존재하는 문제를 가지고 있다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 한국등록특허 10-2315605호

비특허문헌

[0008] (비특허문헌 0001) Liu, X., & Tang, X. (2020, December). Image authentication using QR code watermarking approach based on image segmentation. In 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (pp. 1572-1577). IEEE.

발명의 내용

해결하려는 과제

[0009] 이상의 이유로, 본 발명은 기존의 워터마킹 기법의 Copy Attack 취약성과 가성 양성률 및 음성률 문제를 해결하기 위한 NFT 기반의 워터마킹 장치 및 방법을 제공하는 것이다.

[0010] 본 발명의 기술적 과제들은 이상에서 언급한 기술적 과제로 제한되지 않으며, 언급되지 않은 또 다른 기술적 과제들은 아래의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0011] 이와 같은 목적을 달성하기 위한 본 발명의 일 실시 예에 따른 NFT 기반의 워터마킹 장치는, 워터마크 이미지 좌표를 결정하는 워터마크 위치 결정부; 이미지를 분할하고, 분할된 이미지에서 각각 이미지 특성을 추출하여 해싱한 후, 각 해시 값을 리스트 형태로 저장하는 이미지 분할 및 특성 추출부; 상기 이미지의 저작권 관련 정보를 해싱하고, 해시 값을 NFT ID로 설정하여 NFT를 발행하는 NFT 발행부; 상기 NFT ID와 이미지 해시 값(ImageHash)을 XOR 연산하여 워터마크 값을 계산하는 연산부; XOR 연산에 의해 계산된 워터마크 값을 상기 워터마크 위치 결정부에 의해 정해진 좌표에 삽입하는 워터마크 삽입부; 및 워터마크 값과 이미지 해시 값을 XOR 연산하여 Copy attack를 감지하는 Copy attack 감지부를 포함하여 구성되는 것을 특징으로 한다.

[0012] 실시 예에서, 상기 워터마크 위치 결정부는, 개인키를 해싱하는 해싱부; 상기 해싱부에 의해 해싱된 해시 값을 분할하는 분할부; 상기 분할부에 의해 분할된 16진수 해시 값인 제1 값과 제2 값을 각각 10진수로 변환하는 변환부; 및 상기 변환된 10진수인 제1-1 값과 제2-1값을 각각 이미지 크기로 모듈러 연산하여 워터마크 좌표를 제공하는 좌표 제공부를 포함한다.

[0013] 실시 예에서, 상기 저작권 관련 정보는, 이미지 생성 날짜, 작가 이름, 저장 위치 정보 중 적어도 하나를 포함한다.

[0014] 실시 예에서, 상기 연산부는, XOR 연산하여 계산된 워터마크 값과 이미지 해시 값을 다시 XOR 연산하면 기 발급된 NFT ID를 계산할 수 있다.

[0015] 실시 예에서, 상기 Copy attack 감지부는 원본 이미지와 copy attack 이미지의 NFT ID를 비교하여 copy attack 여부를 감지한다.

[0016] 본 발명의 다른 실시 예에 따른 NFT 기반의 워터마킹 방법은, 개인키 해싱을 통해 워터마크가 삽입될 이미지 좌표를 결정하는 제1 단계; 이미지를 분할하고, 분할된 각 이미지 영역의 특성을 추출하는 제2 단계; 상기 각 이미지 영역의 저작권 관련 정보를 해싱하고, 해시 값을 NFT ID로 설정하여 NFT를 발행하는 제3 단계; NFT ID와 이미지 해시 값을 XOR 연산하여 워터마크 값을 계산하는 제4 단계; 상기 워터마크 값을 상기 이미지 좌표에 삽입하는 제5 단계; 및 상기 워터마크 값과 이미지 해시 값을 XOR 연산하여 Copy attack를 감지하는 제6 단계를 포함하여 수행하는 것을 특징으로 한다.

[0017] 실시 예에서 상기 제1 단계는, 개인키를 해싱하는 단계; 상기 해싱된 해시 값을 X 좌표 값 계산용 제1 값과 Y 좌표 값 계산용 제2 값으로 분할하는 단계; 상기 분할된 제1 값과 제2 값을 10진수인 제1-1 값과 제2-1값으로 변환하는 단계; 및 상기 제1-1 값과 제2-1값을 각각 이미지 크기로 모듈러 연산하여 상기 이미지 좌표를 계산하는 단계를 포함하여 수행한다.

[0018] 실시 예에서 상기 제6 단계의 Copy attack 감지는, NFT ID의 계산 결과를 기초로 하여 수행한다.

발명의 효과

[0019] 개시된 발명의 일 측면에 따르면, 계산된 이미지 좌표에 워터마크를 삽입하는 NFT 기반의 워터마킹 방안을 제안하고 있고, 이에 이미지 워터마크에 대한 Copy Attack을 가성 양성률 및 가성 음성률 없이 감지할 수 있는 효과가 있다.

도면의 간단한 설명

[0020] 도 1은 본 발명의 바람직한 실시 예에 따른 NFT기반의 워터마킹 장치를 나타내는 구성도이다.

도 2는 도 1의 워터마크 위치 결정부의 구성도이다.

도 3은 본 발명의 바람직한 실시 예에 따른 NFT기반의 워터마킹 방법을 설명하는 모식도이다.

도 4 내지 도 15은 본 발명에 따라 이미지 워터마크의 Copy Attack을 측정하는 실험 과정을 보여준 예시 도면들이다.

발명을 실시하기 위한 구체적인 내용

[0021] 명세서 전체에 걸쳐 동일 참조 부호는 동일 구성요소를 지칭한다. 본 명세서가 실시예들의 모든 요소들을 설명하는 것은 아니며, 개시된 발명이 속하는 기술분야에서 일반적인 내용 또는 실시예들 간에 중복되는 내용은 생략한다. 명세서에서 사용되는 '부, 모듈, 부재, 블록'이라는 용어는 소프트웨어 또는 하드웨어로 구현될 수 있으며, 실시예들에 따라 복수의 '부, 모듈, 부재, 블록'이 하나의 구성요소로 구현되거나, 하나의 '부, 모듈, 부재, 블록'이 복수의 구성요소들을 포함하는 것도 가능하다.

[0022] 명세서 전체에서, 어떤 부분이 다른 부분과 '연결'되어 있다고 할 때, 이는 직접적으로 연결되어 있는 경우뿐 아니라, 간접적으로 연결되어 있는 경우를 포함하고, 간접적인 연결은 무선 통신망을 통해 연결되는 것을 포함한다.

[0023] 또한 어떤 부분이 어떤 구성요소를 '포함'한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있는 것을 의미한다.

[0024] 명세서 전체에서, 어떤 부재가 다른 부재 '상에' 위치하고 있다고 할 때, 이는 어떤 부재가 다른 부재에 접해 있는 경우뿐 아니라 두 부재 사이에 또 다른 부재가 존재하는 경우도 포함한다.

[0025] 제1, 제2 등의 용어는 하나의 구성요소를 다른 구성요소로부터 구별하기 위해 사용되는 것으로, 구성요소가 전술된 용어들에 의해 제한되는 것은 아니다.

[0026] 단수의 표현은 문맥상 명백하게 예외가 있지 않는 한, 복수의 표현을 포함한다.

[0027] 각 단계들에 있어 식별부호는 설명의 편의를 위하여 사용되는 것으로 식별부호는 각 단계들의 순서를 설명하는 것이 아니며, 각 단계들은 문맥상 명백하게 특정 순서를 기재하지 않는 이상 명기된 순서와 다르게 실시될 수 있다.

[0029] 이하 첨부된 도면들을 참고하여 개시된 발명의 작용 원리 및 실시예들에 대해 설명한다.

[0030] 도 1은 본 발명의 바람직한 실시 예에 따른 NFT기반의 워터마킹 장치를 나타내는 구성도이고, 도 2는 도 1의 워터마크 위치 결정부의 구성도이다.

[0031] 도 1을 참조하면, 본 발명의 워터마킹 장치(1000)는 워터마크 위치 결정부(110), 이미지 분할 및 특성 추출부(120), NFT 발행부(130), 연산부(140), 워터마크 삽입부(150), Copy attack 감지부(160)를 포함할 수 있다.

[0032] 워터마크 위치 결정부(110)는, 워터마크 좌표를 정하는 유닛이다. 이러한 워터마크 위치 결정부(110)는 도 2에

도시한 바와 같이 개인 키를 해싱하는 해싱부(111), 해싱된 해시 값을 1/2로 분할하는 분할부(112), 분할부에 분할된 16진수 해시 값인 제1 값과 제2 값을 각각 10진수로 변환하는 변환부(113), 변환된 10진수인 제1-1 값과 제2-1값을 각각 이미지 크기로 모듈러(Moduler) 연산하여 워터마크 좌표를 정해 제공하는 좌표 제공부(114)를 포함하여 구성할 수 있다. 모듈러 연산은 특정 수를 나눔 연산 했을 때 나머지를 반환하는 연산법이다.

[0033] 여기서 상기 제1 값은 X 좌표 계산에 사용되며, 상기 제2 값을 Y 좌표 계산에 사용된다.

[0034] 이미지 분할 및 특성 추출부(120)는, 이미지를 분할하고, 분할된 이미지에서 각각 색상, 픽셀 등의 이미지 특성을 추출하여 해싱한 후, 각 해시 값을 리스트 형태로 저장한다.

[0035] NFT 발행부(130)는, 이미지 생성 날짜, 작가 이름, 저장 위치 등의 저작권 관련 정보를 해싱하고, 해시 값을 NFT ID로 설정하여 NFT를 발행한다.

[0036] 연산부(140)는, 워터마크 값을 XOR 연산한다. 즉 상기 NFT ID와 이미지 해시값(ImageHash)을 XOR 연산하여 워터마크 값을 계산한다. 참고로 XOR 연산을 수행한 결과를 동일한 값으로 다시 XOR 연산하면 원래의 값이 복원되게 된다.

[0037] 그리고 워터마크 값과 이미지 해시 값을 XOR 연산하면 기존에 발급한 NFT ID를 도출해낼 수 있어야 한다.

[0038] 예를 들면 다음과 같이 표현할 수 있다.

$$\begin{aligned} \text{Watermark} &= \text{NFT ID} \text{ XOR ImageHash} \\ \text{NFT ID} &= \text{Watermark} \text{ XOR ImageHash} \end{aligned}$$

[0039]

[0040] 워터마크 삽입부(150)는, XOR 연산에 의해 계산된 워터마크 값을 상기 워터마크 위치 결정부에 의해 정해진 이미지 좌표에 삽입하는 유닛이다.

[0041] Copy attack 감지부(150)는 워터마크 값과 이미지 해시 값을 XOR 연산하여 Copy attack를 감지하는 유닛이다.

[0043] 도 3은 본 발명의 바람직한 실시 예에 따른 NFT기반의 워터마킹 방법을 설명하는 모식도이다. 도 3을 참조하여 이미지에 워터마크를 삽입하고 copy attack를 감지하는 과정에 대해 살펴본다.

[0044] 제1 단계로, 워터마크 위치 결정부가 워터마크를 삽입할 이미지 좌표를 계산하는 과정이다(S110).

[0045] 워터마크의 이미지 좌표 값을 계산하기 위해 먼저 해싱부(111)는 개인키를 해싱하고, 분할부(112)가 해싱된 해시 값을 1/2로 분할한다. 제1 값은 X 좌표 값을 계산하는데 사용되며, 제2 값은 Y 좌표 값을 계산하는데 사용된다.

[0046] 분할부(112)에 의해 분할된 제1 값과 제2 값은 16진수이다. 그래서 이를 10진수로 변환할 필요가 있다. 변환부(113)가 제1 값과 제2 값을 10진수인 제1-1 값과 제2-1값으로 변환한다. 16진수를 10진수로 변환은 일반적으로 사용되는 알고리즘 등을 사용할 수 있다. 이후 좌표 제공부(114)는 제1-1 값과 제2-1값을 각각 이미지 크기로 모듈러 연산하여 워터마크 좌표를 결정한다. 결정된 워터마크 좌표에는 이후 수행되는 과정에 의해 계산된 워터마크가 삽입될 것이다.

[0047] 제2 단계 이후 제4 단계는 워터마크 좌표에 삽입될 워터마크를 구하는 과정일 수 있다. 구체적으로 살펴본다.

[0048] 제2 단계는, 이미지 분할 및 특성 추출부(120)가 이미지 분할 및 특성 추출하는 과정이다(S120).

[0049] 이미지 분할은 하나의 이미지를 대상으로 n개로 분할하는 것이다. 이때 분할된 n개의 각 영역은 모두 동일하거나 상이할 수 있다. 그리고 분할된 n개의 이미지에서 각각 색상, 픽셀 등의 이미지 특성을 추출하여 해싱한 후, 각 해시 값을 리스트 형태로 저장한다.

[0050] 예를 들면 도 2의 2단계 과정을 보면 총 9개로 이미지 분할되고, 각 이미지 영역마다 해시 값이 추출된 후, 각각의 해시 값(H1~H9)이 리스트화되었음을 알 수 있다.

[0051] 제3 단계는, NFT 발행부(130)가 NFT를 발행하는 과정이다(S130).

[0052] NFT 발행은 저작권 정보를 기반으로 하며, 저작권 정보 해시 값을 NFT ID로 설정하여 발행할 수 있다. 해시 값은 이미지 생성날짜, 작가 이름, 저장 위치 등을 포함할 수 있고, 이러한 정보들을 해싱한 값일 수 있다.

- [0053] 제4 단계는, 연산부(140)가 워터마크 좌표에 삽입될 워터마크 값을 연산하는 과정이다(S140).
- [0054] 연산부(140)가 NFT ID와 이미지 해시 값을 XOR 연산하여 워터마크 값을 계산한다. 이때 앞서 언급한 바와 같이 계산된 워터마크 값은 이미지 해시 값과 XOR 연산을 하면, NFT ID를 계산할 수 있게 된다.
- [0055] 제5 단계는, 워터마크 삽입부(150)가 상기 1단계에서 계산된 이미지 좌표에 상기 4단계에 의해 계산된 워터마크 값을 삽입하는 과정이다(S150).
- [0056] 이러한 과정들에 의해 NFT를 기반으로 하여 워터마크를 이미지에 삽입할 수 있게 된다.
- [0057] 이후 제6 단계는 Copy attack 감지부(160)가 워터마크 값과 이미지 해시값을 XOR 연산하여 Copy attack를 감지하는 과정이다(S160). 예를 들어 아래 표와 같이 XOR 연산을 통해 원본 이미지와 Copy attack 이미지의 동일성 여부를 판단할 수 있다. 결과적으로 원본 이미지와 복제본인 Copy attack 이미지를 쉽게 구분할 수 있고, 따라서 무단 복제와 배포를 추적하고, 방지할 수 있는 기반을 제공할 수 있다.

Watermark 값	원본 이미지	Copy Attack 이미지
Watermark = NFT_ID XOR ImageHash = 1010 XOR 1100 = 0110	NFT_ID = Watermark XOR ImageHash = 0110 XOR 1100 = 1010	NFT_ID = Watermark XOR ImageHash = 0110 XOR 1101 = 1011
	↑	↑
	≠	↑

- [0058]
- [0059] 본 발명의 NFT 기반의 워터마킹 방안은 이미지 워터마크에 대한 Copy Attack을 가성 양성률 및 가성 음성률 없이 감지할 수 있음을 언급한 바 있다. 이에 가성 양성률 및 가성 음성률 없이 이미지 워터마크의 Copy Attack을 측정할 수 있음을 실험을 통해 확인하였고, 실험은 도 4 내지 도 15을 참조하면 알 수 있다.
- [0060] 예를 들어, 도 4는 이미지 폴더 내에 생성된 original 폴더에 원본 이미지 데이터 셋 100개를 준비된 도면이고, 도 5는 블록체인을 기반으로 계정 주소 및 개인키 리스트 작성 코드를 보여준 도면, 도 6은 각 이미지당 계정 및 개인키 할당 코드를 보여주고 있는 도면이다.
- [0061] 그리고 도 7은 이미지 저작권 정보, 즉 이미지 생성 날짜, 작가 이름, 저장 위치 등의 작성 코드, 도 8은 이미지 저작권 정보를 해싱하고 해시 값을 NFT ID로 하여 발행한 NFT 코드를 나타내고 있다.
- [0062] 그리고 도 9는 개인키 해싱과 모듈러 연산을 통해 계산된 워터마크 좌표 결정 코드를 나타낸 도면, 도 10은 이미지를 분할하고 이미지 특성을 추출하여 해싱한 코드를 나타낸 도면, 도 11은 NFT ID와 해시값을 XOR 연산하여 워터마크 값을 계산하기 위한 함수 코드, 도 12는 이미지 좌표에 워터마크 값을 삽입하는 워터마크 삽입 코드, 도 13은 copy attack 시뮬레이션 코드, 도 14는 XOR 연산 후 결과 표시 코드이다.
- [0063] 그리고 도 15는 결과 표시 코드를 CSV 파일로 정리한 도면이다. 상기 XOR 연산을 수행하고, 이미지 정보가 상이하면 NFT ID가 다르게 계산되는 것을 통해 copy attack을 감지하고, 그 결과는 Success로 표시하였다.
- [0064] 이를 보면, 가성 양성률 및 가성 음성률 없이 이미지 워터마크의 Copy Attack을 측정할 수 있었음을 알 수 있다.
- [0065] 이상의 설명은 본 발명의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자라면 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 다양한 수정 및 변형이 가능할 것이다. 따라서, 본 발명에 개시된 실시예들은 본 발명의 기술 사상을 한정하기 위한 것이 아니라 설명하기 위한 것이고, 이러한 실시예에 의하여 본 발명의 기술 사상의 범위가 한정되는 것은 아니다. 본 발명의 보호 범위는 아래의 청구범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술 사상은 본 발명의 권리범위에 포함되는 것으로 해석되어야 할 것이다.

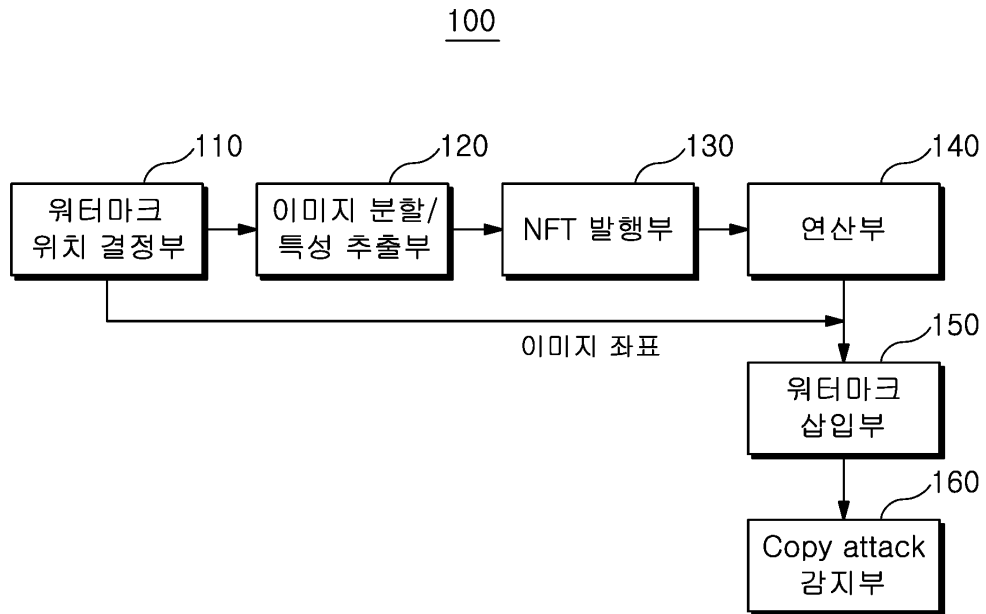
부호의 설명

- [0066] 100: NFT 기반의 워터마킹 장치
- 110: 워터마크 위치 결정부

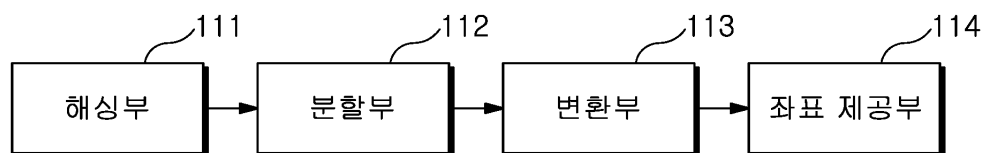
- 120: 이미지 분할/특성 추출부
- 130: NFT 발행부
- 140: 연산부
- 150: 워터마크 삽입부
- 160: Copy attack 감지부

도면

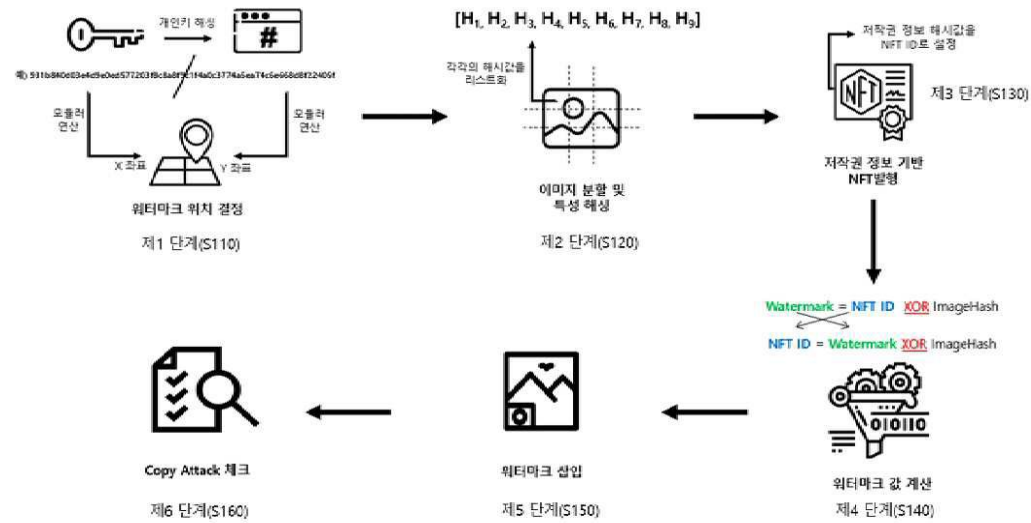
도면1



도면2



도면3



도면4

original

1_flower(640_426)	21_beach(1920_1440)	41_salad(1280_853)	61_sunset(640_480)	81_laptop(1920_1280)
2_flower(1279_1920)	22_beach(640_426)	42_salad(1920_1280)	62_sunset(1280_853)	82_room(640_484)
3_flower(1920_1280)	23_beach(1280_778)	43_cat(1280_426)	63_sunset(1920_1279)	83_room(1280_846)
4_flower(640_480)	24_beach(1920_1280)	44_cat(1280_781)	64_car(640_427)	84_room(1920_1233)
5_flower(1280_823)	25_bike(640_426)	45_cat(1920_1279)	65_car(1280_852)	85_room(640_417)
6_flower(1920_1280)	26_bike(1280_853)	46_pizza(640_424)	66_car(1920_1281)	86_room(1280_853)
7_sky(640_426)	27_bike(1920_1280)	47_pizza(1280_864)	67_car(640_426)	87_room(1920_1080)
8_sky(1280_853)	28_bike(640_426)	48_pizza(1920_1280)	68_car(1280_853)	88_pumpkin(640_426)
9_sky(1920_1200)	29_bike(1280_853)	49_pizza(640_426)	69_car(1920_960)	89_pumpkin(1280_853)
10_sky(640_360)	30_bike(1920_1281)	50_pizza(1280_717)	70_bookshelf(640_423)	90_pumpkin(1920_1189)
11_sky(1280_850)	31_coffee(640_426)	51_pizza(1920_1440)	71_bookshelf(1280_960)	91_pumpkin(640_427)
12_sky(1920_1440)	32_coffee(1280_853)	52_tea(640_428)	72_bookshelf(1920_1280)	92_pumpkin(1280_853)
13_tree(640_426)	33_coffee(1920_1281)	53_tea(1280_853)	73_bookshelf(640_426)	93_pumpkin(1920_1280)
14_tree(1280_853)	34_coffee(640_415)	54_tea(1920_1320)	74_bookshelf(1280_853)	94_candle(640_426)
15_tree(1920_1273)	35_coffee(1280_856)	55_tea(640_360)	75_bookshelf(1920_1352)	95_candle(1280_747)
16_tree(640_426)	36_coffee(1920_1005)	56_tea(1280_853)	76_laptop(640_426)	96_candle(1920_1280)
17_tree(1280_853)	37_salad(640_427)	57_tea(1920_1280)	77_laptop(1280_853)	97_candle(640_426)
18_tree(1920_1138)	38_salad(1280_854)	58_salad(640_426)	78_laptop(1920_1285)	98_candle(1280_853)
19_beach(640_516)	39_salad(1920_1280)	59_salad(1280_850)	79_laptop(640_427)	99_candle(1920_1246)
20_beach(1280_716)	40_salad(640_427)	60_sunset(1920_1277)	80_laptop(1280_853)	100_(1280_1920)

도면5

```
# 주석의 개인 키 목록 리소스
accounts_list = [
    ('0x1c811a53eCADAbdf2eb0fce75034a3642009fa7A', '0xc4a96dae9a854547c318fb8e0d34babbf1113d1bda28c55ca3d90080e5df'), #1
    ('0x4060ce882E9b4f5C3D0e51180fcb0f8181f2607', '0x2b7b8837add5cf5bb78a1e125543b4cd10cc57eda761c937c546006138cc5187'), #2
    ('0x3f8A55304F80b84b4bc1798114c32ac70dadA40', '0x59ea5af9d4c357af9bb91e95a8f061ed8ee0af574a42b0e81afcd2744a8e2f'), #3
    ('0xabcAcad1394E58E24e26707b208fCcc3b9f475D8', '0x7115977d47b83c0412e5bd768a4607896f6b374a4b88dc4e65a7d9337c7bc575'), #4
    ('0x00B13944a88C0c677782C3e301520CC699A64e', '0x5c638dad9acc3c34ab9437ab2052242f6608bf7308f7092d35e96243648eca42'), #5
    ('0x67b0b24E439a7E0ec0888c37e2e32B1A9A3449', '0x541f0d98444ec90f50bb4e2d5facc9f9d7880f3c7dc41370230825b75461018c'), #6
]
```

도면6

```
# 이미지 파일과 계정을 매핑
image_account_map = {
    '1_flower(640_426).jpg': accounts_list[0],
    '2_flower(1279_1920).jpg': accounts_list[1],
    '3_flower(1920_1280).jpg': accounts_list[2],
    '4_flower(640_480).jpg': accounts_list[3],
    '5_flower(1280_823).jpg': accounts_list[4],
    '6_flower(1920_1280).jpg': accounts_list[5],
}
```

도면7

```
# 이미지 저작권 정보를 담은 딕셔너리
image_metadata_dict = {
    '1_flower(640_426).jpg': {
        'createdDate': "2023-01-01",
        'author': "Author1",
        'location': "Location1"
    },
    '2_flower(1279_1920).jpg': {
        'createdDate': "2023-01-02",
        'author': "Author2",
        'location': "Location2"
    },
    '3_flower(1920_1280).jpg': {
        'createdDate': "2023-01-03",
        'author': "Author3",
        'location': "Location3"
    },
    '4_flower(640_480).jpg': {
        'createdDate': "2023-01-04",
        'author': "Author4",
        'location': "Location4"
    },
    '5_flower(1280_823).jpg': {
        'createdDate': "2023-01-05",
        'author': "Author5",
        'location': "Location5"
    },
}
```

도면8

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "../sc/ERC721URIStorage.sol";

contract ImageNFT is ERC721URIStorage {
    // 토큰 ID와 메타데이터 해시값 매핑
    mapping(uint256 => bytes32) private _tokenMetadataHashes;

    constructor() ERC721("ImageNFT", "IMG") {}

    function mintNFT(
        address recipient,
        string memory imageInfo
    ) public returns (uint256) {
        // 이미지 저작권 관련 정보를 해시, 토큰 ID로 사용
        bytes32 metadataHash = keccak256(abi.encodePacked(imageInfo));
        uint256 tokenId = uint256(metadataHash);

        // _exists 함수를 호출하여 해당 토큰 ID가 이미 발행되었는지 검사
        require(!_exists(tokenId), "Token already minted");

        // _mint 함수를 호출하여 recipient에게 새 토큰을 발행하고,
        _mint(recipient, tokenId);

        // 토큰 ID와 연결된 메타데이터의 해시값 저장
        _tokenMetadataHashes[tokenId] = metadataHash;

        return tokenId;
    }

    // 주어진 토큰 ID에 대한 메타데이터 해시값을 반환하는 함수
    function getTokenMetadataHash(uint256 tokenId) public view returns (bytes32) {
        require(_exists(tokenId), "ImageNFT: Metadata query for nonexistent token");
        return _tokenMetadataHashes[tokenId];
    }
}
```

도면9

```
def determine_position(private_key, image_width, image_height):
    hashed_key = hashlib.sha256(private_key.encode()).hexdigest()
    x_hash, y_hash = hashed_key[:len(hashed_key)//2], hashed_key[len(hashed_key)//2:]

    x_coord = int(x_hash, 16) % image_width
    y_coord = int(y_hash, 16) % image_height
    return (x_coord, y_coord)
```

도면10

```

1 image
def extract_features(image):
    # 이미지의 색상, 픽셀 등과 특정 주를 추출
    data = np.array(image)
    avg_color = np.mean(data, axis=(0, 1))
    hashed_features = hashlib.sha256(str(avg_color).encode()).hexdigest()
    return hashed_features

3 images
def split_image(image):
    # 이미지를 수평선으로 각 부분의 색상 계산
    width, height = image.size
    step_x = width // 3
    step_y = height // 3
    hashes = []
    for x in range(0, width, step_x):
        for y in range(0, height, step_y):
            part = image.crop((x, y, x + step_x, y + step_y))
            hashed_part = extract_features(part)
            hashes.append(hashed_part)
    return hashes

```

도면11

```

def xor_binary(bin_str1, bin_str2): # 이진수끼리 XOR 연산을 수행하는 함수
    # 두 이진수 문자열의 길이를 맞추고 XOR 연산을 수행
    max_length = max(len(bin_str1), len(bin_str2))
    bin_str1 = bin_str1.zfill(max_length)
    bin_str2 = bin_str2.zfill(max_length)
    xor_result = ''.join('1' if b1 != b2 else '0' for b1, b2 in zip(bin_str1, bin_str2))
    return xor_result

```

도면12

```

def add_watermark(input_image_path, watermark, position): # 워터마크를 삽입하는 함수
    image = Image.open(input_image_path).convert("RGBA")
    watermark_image = Image.new('RGBA', image.size, (0, 0, 0, 0))

    font = ImageFont.truetype('font/arial.ttf', size=10)

    draw = ImageDraw.Draw(watermark_image)
    watermark_text = watermark[:20] # 워터마크 글자수를 제한
    draw.text(position, watermark_text, fill=(0, 0, 0, 50), font=font)

    combined = Image.alpha_composite(image, watermark_image)
    return combined.convert("RGB") # RGBA에서 RGB로 변환하여 저장

```

도면13

```

def copy_attack_simulation(original_image_data, another_image_folder, copyattacked_folder):
    another_images = os.listdir(another_image_folder)
    another_images.sort() # 이미지 이름을 정렬

    for i, image_name in enumerate(another_images):
        index = i % len(original_image_data)
        next_index = (i + 1) % len(original_image_data)

        position = tuple(map(int, original_image_data[index]['WatermarkPosition'].split(',')))
        watermark = original_image_data[next_index]['Watermark']

        another_image_path = os.path.join(another_image_folder, image_name)
        copyattacked_image_path = os.path.join(copyattacked_folder, f'copyattacked_{image_name}')

        # 워터마크 적용 및 저장
        copyattacked_image = add_watermark(another_image_path, watermark, position)
        copyattacked_image.save(copyattacked_image_path, format="JPEG")

        # 이미지 해시 계산
        copyattacked_image = Image.open(another_image_path)
        copyattacked_image_hashes = split_image(copyattacked_image)
        copyattacked_imagehash = ''.join(copyattacked_image_hashes)

        copyattacked_imagehash_binary = bin(int(copyattacked_imagehash, 16))[2:].zfill(256)

```

도면14

```
# XOR 연산 수행
calculated_tokenid = xor_binary(copyattacked['WatermarkUsed'], copyattacked_imagehash_binary)

# 결과 확인
if int(calculated_tokenid, 2) != original['TokenID']:
    result = 'Success'
else:
    result = 'Fail'
```

도면15

G	H	I
OriginalTokenID	CalculatedTokenID	Result
1.03893E+77	17762718540452328	Success
2.47989E+76	27844492356830632	Success
1.13596E+77	9862614168272016	Success
3.93329E+76	9862614168272016	Success
9.95595E+76	5819848796389818	Success
1.51697E+76	8972818410142989	Success
1.15268E+77	8972818410142989	Success
1.09843E+77	5484735250203729	Success
7.08386E+76	7507607025458990	Success
3.22567E+76	7403135951371141	Success
8.92981E+76	5286591722824084	Success
8.87333E+76	6713835536881165	Success
8.4537E+76	6713835536881165	Success
8.52162E+76	3573515950103487	Success
9.68629E+76	3375990072173626	Success
5.84146E+76	2255236344647849	Success
8.46427E+76	4141991212704263	Success
4.79342E+76	1039437129408783	Success
6.62788E+76	1023959258668358	Success
7.85986E+76	1719916122755342	Success
1.25119E+76	2925136918399577	Success
5.22728E+76	5750129425999477	Success
1.12067E+77	8292037747975192	Success