

DISTRIBUTED LEDGER BASED CRYPTOGRAPHIC SYSTEMS AND METHODS FOR IMPROVING DATA INTEGRITY

BACKGROUND OF THE INVENTION

5 Related Applications

This application claims the priority of U.S. provisional application 63/088,412 filed on October 6, 2020, titled “ BLOCKCHAIN BASED MEDIA ANTI-TAMPERING METHODS AND SYSTEMS”.

10 This application claims the benefit of U.S. non-provisional application 16/801,114 filed on Feb. 25, 2020, titled “CREDENTIAL VERIFICATION AND ISSUANCE THROUGH CREDENTIAL SERVICE PROVIDERS”, which is incorporated herein by reference at its entirety.

Field of the Invention

15 The present invention is related to a distributed ledger based cryptographic systems and methods for improving the integrity and privacy of data; more specifically, a blockchain based cryptographic systems and methods for securing the integrity and privacy of private data which are generated from a data recording device and stored remotely in a remote data storage.

20 Description of Related Art

The emergence of distributed ledger technologies (DLT), including blockchains, offers a solution to improve data integrity. DLT is trusted because transactions recorded in such a distributed ledger across the entire network of nodes are very difficult or even impossible to alter. For this reason, various blockchain security methods have been implemented to provide
25 secure transactions across networks. The blockchain itself may be regarded as an open and distributed ledger that can record transactions between parties. After a block in the blockchain is recorded and linked, the data in any given block cannot be altered or tampered without altering all the other blocks; therefore, blockchain is well suited to many records management activities such as cryptocurrency. However, DLT such as blockchains may be secure and transparent at the
30 same time because transactions are recorded in a large number of blocks or network nodes forming the distributed transaction consensus network. Thus, recording data themselves in a

distributed ledger may cause privacy issues or even violate the related regulations. Furthermore, there has been few DLT based cryptographic system and method for maintaining the integrity and privacy of data which are generated from a data recording device and stored remotely in a remote data storage. Therefore, the present invention aims to implement DLT technology to
5 protect the integrity of the data which are generated from the data recording device, such as a surveillance camera, driving recorder, or a mobile device.

SUMMARY

The present disclosure relates to distributed ledger based cryptographic systems and
10 methods for improving data integrity. In the age of IOT (internet of things), various types of data are recorded by different devices at any moment and circulated via various wired and wireless networks. In addition, the advanced data processing technologies also cause the data recorded by a data recording device can be easily corrupted, falsified, forged, tampered, or altered without authority. As mentioned earlier, the distributed ledger technologies (DLT), including
15 blockchains, offers a solution to improve data integrity.

The present disclosure describes systems and methods to generate a commitment and store it in a distributed ledger. A commitment is a cryptographic algorithm that allows one to commit to a chosen value or statement while keeping it hidden to others, with the ability to reveal the committed value later. Commitment is binding because the one who committed can no
20 longer change the chosen value or statement. Here, the chosen statement is a data signature which is generated by a data recording device signing a data piece. Recording the commitment in a distributed ledger makes it very difficult or even impossible to alter the commitment. In addition, the committed value or statement, such as the data signature, cannot be changed as well. By revealing the commitment in the future, the data signature will be verified. And then the
25 authenticity of the data piece will be verified. At the same time, the commitment does not contain any personal information, and is not traceable or even linkable to the data piece recorded by a data recording device, such as a digital video recorder.

In one embodiment, the ecosystem for using distributed ledgers to improve data integrity may include a distributed integrity ledger, a distributed identity ledger, multiple network node
30 managers to manage transactions in both ledgers, a data recording device, a manufacturer to

make the device, a user to use the device, a data center to store recorded data pieces, and a verifier who needs to verify authenticity of the recorded data.

In one embodiment, the distributed integrity ledger, maintained by a first distributed transaction consensus network ("first distributed network"), is used to store commitments
5 generated by the data recording device to verify authenticity of recorded data pieces. The distributed identity ledger, maintained by a second distributed transaction consensus network ("second distributed network"), is used to store DIDs (decentralized identifier), credential schemas, credential definitions, and public keys associated with credential owners and credential publishers to verify the identity of the user, the data center, and the verifier. The multiple
10 network node managers may respectively manage a node in the first distributed network and the second distributed network. Thus, the network node managers may record transactions and retrieve information from both the distributed integrity ledger and the distributed identity ledger.

In one embodiment, the data recording device may be any device that is able to record data and transmit data to other related parties in various wired or wireless manners. In one
15 embodiment, the data recording device may be a digital recorder or driving recorder to record video clips of the surrounding traffic conditions.

In one embodiment, the manufacturer may store an original device certificate, a device signing private key and a device signing public key, and some other device related information in the data recording device. After the user purchases the data recording device, he/she has to
20 register the device with the manufacturer. The user also has to establish a DID, a user identification, in the distributed identity ledger. Then the user can begin to use the data recording device to record data pieces.

In one embodiment, each recorded data piece is encrypted by a data encryption key and an encrypted data piece is transmitted to the data center for storage. The data recording device
25 also generated a data signature, a commitment string and a commitment. The data signature and the commitment string may be stored in the data center. The commitment may be recorded in the distributed integrity ledger through a network node manager.

In one embodiment, the user may initiate the verification procedure for the verifier to eventually receive a data encryption key, an encrypted data piece, the commitment, the data
30 signature, and the commitment string. As a result, the verifier, such as the judge/the court, may decrypt the encrypted data piece by using the data encryption key, verify the authenticity of the

recorded data by using the data signature, verify the data signature by using the commitment and the commitment string, and verify the commitment by inquiring the distributed integrity ledger.

BRIEF DESCRIPTION OF THE DRAWINGS

5 FIG. 1 is a schematic view of the ecosystem for using distributed ledgers to improve data integrity in accordance with an embodiment of the present invention.

 FIG. 2 is a schematic view of the manufacturing stage of the data recording device in the distributed ledger based cryptographic system and method in accordance with the embodiment of the present invention.

10 FIG. 3 is a schematic view of the registration stage of the data recording device in the distributed ledger based cryptographic system and method in accordance with the embodiment of the present invention.

 FIG. 4 is a schematic view of the data recording stage of the distributed ledger based cryptographic system and method in accordance with the embodiment of the present invention.

15 FIG. 5 is a schematic view of the verification stage of the distributed ledger based cryptographic system and method in accordance with the embodiment of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

20 The terminology used in the description presented below is intended to be interpreted in its broadest reasonable manner, even though it is used in conjunction with a detailed description of certain specific embodiments of the technology. Certain terms may even be emphasized below; however, any terminology intended to be interpreted in any restricted manner will be specifically defined as such in this Detailed Description section.

25 The present disclosure relates to distributed ledger based cryptographic systems and methods for improving data integrity. In the age of IOT (internet of things), various types of data are recorded by different devices at any moment and circulated via various wired and wireless networks. In addition, the advanced data processing technologies also cause the data recorded by a data recording device can be easily corrupted, falsified, forged, tampered, or altered without authority. The emergence of distributed ledger technologies (DLT), including blockchains, offers
30 a solution to improve data integrity. DLT is trusted because transactions recorded in such a distributed ledger across the entire network of nodes are very difficult or even impossible to

alter. However, DLT may be transparent at the same time because transactions are recorded in a large number of network nodes forming the distributed transaction consensus network. Thus, recording data themselves in a distributed ledger may cause privacy issues or even violate the related regulations.

5 The present disclosure describes systems and methods to generate a commitment and store it in a distributed ledger. A commitment is a cryptographic algorithm that allows one to commit to a chosen value or statement while keeping it hidden to others, with the ability to reveal the committed value later. Commitment is binding because the one who committed can no longer change the chosen value or statement. Here, the chosen statement is a data signature
10 which is generated by a data recording device signing a data piece. Recording the commitment in a distributed ledger makes it very difficult or even impossible to alter the commitment. In addition, the committed value or statement, such as the data signature, cannot be changed as well. By revealing the commitment in the future, the data signature will be verified. And then the authenticity of the data piece will be verified. At the same time, the commitment does not
15 contain any personal information, and is not traceable or even linkable to the data piece recorded by a data recording device, such as a digital video recorder.

As shown in FIG. 1, The ecosystem 100 for using distributed ledgers to improve data integrity may include a distributed integrity ledger 110, a distributed identity ledger 120, multiple network node managers 130, 132, 134, 136 to manage transactions in both ledgers, a
20 data recording device 150, a manufacture 140 to make the device, a user 160 to use the device, a data center 170 to store recorded data pieces, and a verifier 170 who needs to verify authenticity of the recorded data.

The distributed integrity ledger 110, maintained by a first distributed transaction consensus network ("first distributed network") 115, is used to store commitments generated by
25 the data recording device 150 to verify authenticity of recorded data pieces. The distributed identity ledger 120, maintained by a second distributed transaction consensus network ("second distributed network") 125, is used to store DIDs (decentralized identifier), credential schemas, credential definitions, and public keys associated with credential owners and credential
publishers to verify the identity of the user 160, the data center 170, and the verifier 180. The
30 multiple network node managers 130, 132, 134, 136 may respectively manage a node in the first distributed network and the second distributed network. Thus, the network node managers may

record transactions and retrieve information from both the distributed integrity ledger 110 and the distributed identity ledger 120. The network node managers may be telecommunication carriers, such as ATT and Sprint in the United States, organizations operated by government agencies, or other DLT related companies, such as TBCASOFT. One of the network node
5 managers may be a network administrator. In one embodiment, certain function, such as generating a DID and recording it in the distributed identity ledger 120 is reserved for the network administrator.

The data recording device 150 may be any device that is able to record data and transmit data to other related parties via various wired or wireless manners. In one embodiment, the data
10 recording device 150 may be a digital recorder or driving recorder to record video clips of the surrounding traffic conditions. The data recording device 150 may have a recording module for recording a data piece, a memory module for storing an original device certificate signed by a manufacturer, a device signing public key, and a device signing private key; and a processor module for generating a data encryption key, encrypting the data piece with the data encryption
15 key, encrypting the data encryption key with the encryption public key.

The data recording device 150 is made by the manufacturer 140. As part of the manufacturing process, the manufacturer 140 may store an original device certificate, a device signing private key and a device signing public key, and some other device related information in the data recording device 150. After the user 160 purchases the data recording device 150, he/she
20 has to register the device with the manufacturer 140. The user also has to establish a DID, a user identification, in the distributed identity ledger.

Then the user 160 can begin to use the data recording device to record data pieces. Each recorded data piece is encrypted by a data encryption key and an encrypted data piece is transmitted to the data center 170 for storage. The data recording device 150 also generated a
25 data signature, a commitment string and a commitment. The data signature and the commitment string may be stored in the data center 170. The commitment may be recorded in the distributed integrity ledger 110 through a network node manager.

The need for verifying authenticity of the recorded data occurs in various situations. For example, when there is a car accident and the liability is in dispute, a user or a judge/a court may
30 need to review the video clip recorded by a driving recorder at the time of accident and verify authenticity of the recorded video clip. In one embodiment, the user 160 may initiate the

verification procedure for the verifier to eventually receive a data encryption key, an encrypted data piece, the commitment, the data signature, and the commitment string. As a result, the verifier, such as the judge/the court, may decrypt the encrypted data piece by using the data encryption key, verify the authenticity of the recorded data by using the data signature, verify the data signature by using the commitment and the commitment string, and verify the commitment by inquiring the distributed integrity ledger 110.

The manufacturer 140, the data recording device, the user, the data center, and the verifier may need to interact with either one of or both the distributed integrity ledger 110 and the distributed identity ledger 120 through one of the network node managers 130, 132, 134, 136. Each may have its own network node manager. In another embodiment, two or more may share the same network node manager.

As shown in FIG. 2, in a manufacturing stage, the manufacturer 140 generates an original device certificate by signing some product information, such as a module number and a serial number, with a manufacturer's private key. The manufacturer 140 has a pair of signing keys, a signing private key and a signing public key. As described above, the manufacturer's signing private key is used to sign the product information to generate a device certificate, including an original device certificate and an updated device certificate. The manufacturer's signing public key is distributed to others, such as a network node manager, who need to verify the authenticity of information provided and signed by the manufacturer's signing private key. The manufacturer 140 also generates a pair of signing keys for the data recording device 150, a device signing private key and a device signing public key. The manufacturer 140 stores the original device certificate, the device signing private key and the device signing public key in the data recording device. Additional product information, such as warranty period, country of origin, date of manufacture, etc., may be stored in the data recording device 150 as well. The manufacturer 140 may outsource manufacturing to OEM or ODM. Thus, the steps described above may be implemented by other parties working for the manufacturer 140. For this disclosure, these other parties are considered as the manufacturer 140.

FIG. 3 illustrates interactions between the user 160, the data recording device 150, the manufacturer 140, the network node manager 130, and the distributed identity ledger 120 in a registration stage. After the user 160 obtains the data recording device 150, for example through purchase, the user 160 initiates the device registration stage by turning on the data recording

device 150 and causing it to connect with the manufacturer 140 for information transmission via wired or wireless manner. At step 310, the user 160 may need to input his/her own personal information, such as user ID, name, gender, and birthday, to the data recording device 150. At step 320, the data recording device 150 provides to the manufacturer 140 the device signing public key, the original device certificate, the user related information received from the user at step 310, and other device related information, for example a car plate number of the car in which a driving recorder is installed. Some of the above-mentioned information may be optional. For example, the manufacturer 140 may still keep the device signing public key for the data recording device 150.

The manufacturer 140 then verifies the original device certificate by using the manufacturer's signing public key to confirm that the original device certificate is authentic. At step 330, after verification, the manufacturer 140 provides the device signing public, the original device certificate, and information related to the device and the user to the network node manager 130. At step 340, the user 160 may also provide his/her personal information, such as user ID, name, gender, and birthday and the device related information to the network node manager 132. As described before, the manufacturer 140 may use its own network node manager 130 and the user 160 uses his/her own network node manager 132. The manufacturer 140 and the user may also use the same network node manager (either 130 or 132).

Thus, between the two network node managers 130 and 132, the following processes have to be performed. First, the network node manager (130 or 132) may verify the original device certificate by using the manufacturer's signing public key. Second, the network node manager may verify the user's DID via the distributed identity ledger 120 and possibly also verify other user's information via the user's ID credential, given that the user 160 already has a DID recorded in the distributed identity ledger 120. If the user 160 is new, a DID has to be created to the user and recorded in the distributed identity ledger 120. The user's DID serves as a user identification. One example of DID is did.sovrin.V4SDRN84Z56d7YV7PBUE6f. Similar to virtual wallet addresses, DIDs are globally unique identifiers that are created by their owner or their network node managers. DIDs have their associated public keys and communication endpoints - addresses where messages can be delivered for that identity. The credential owners of the DIDs hold the corresponding private keys in their wallets that can be managed by their network node managers.

Third, the network node manager (either 130 or 132) generates a pair of encryption keys for the data recording device, an encryption private key and an encryption public key. Fourth, the network node manager (either 130 or 132) issues a user's device credential to the user 160. The user's device credential may include the following fields: user's DID, the device signing public key, the encryption public key, the device information, and the user information. In one embodiment, the device information may include the model number and the serial number of the driving recorder, the car plate number of the car in which the driving recorder is installed, and the model and color of the car. And the user information includes the driver's name, birthday, and the driver's license number. Fifth, the network node manager (either 130 or 132) generates a proof for the user's device credential including the encryption public key. Step 350 is for the two network node managers 130 and 132 to provide related information to each other to accomplish the above five processes.

At step 360, the network node manager (either 130 or 132) provides the manufacturer 140 the proof of the user's device credential, the encryption public key and the user's DID. The manufacturer 140 may verify the proof of the user's device credential. After verification is valid, the manufacturer 140 generates an updated device certificate which is signed by the manufacturer's signing private key. The updated device certificate includes the user's DID, the encryption public key, and the device related information. At step 370, the manufacturer 140 provides the data recording device 150 the proof for the updated device certificate, the user's DID, and the encryption public key.

FIG. 4 illustrates interactions between the data recording device 150, the data center, the network node manager 132, and the distributed integrity ledger 110 in a recording stage. Once its recording function is turned on, the data recording device 150 begins to record a data piece. First, the data recording device 150 generates a data encryption key for the recorded data piece and then encrypts the recorded data piece to derive an encrypted data piece. Second, the data recording device 150 encrypts the data encryption key by using the encryption public key to generate an encrypted data encryption key. Third, the data recording device 150 signs the recorded data piece with the device signing private key to generate a data signature. Fourth, the data recording device 150 generates a commitment string. A commitment string may be a random number, such as a 256 bits number like
ba3253876aed6bc22d4a6ff53d8406c6ad864195ed144ab5c87621b6c233b548. Then, the data

recording device 150 generates a commitment using the commitment string to lock the data signature. One example of the commitment is 3627909a29c31381a071ec27f7c9ca97726182aed29a7ddd2e54353322cfb30a. Fifth, the data recording device 150 generates a commitment signature by signing the commitment with the device signing private key.

At step 410, the data recording device 150 provides the data center 170 the encrypted data piece, the encrypted data encryption key, the data signature, the commitment, and the commitment string for storage. In some situations, the updated device certificate and the related device information may be provided from the data recording device 150 to the data center 170 as well. At step 420, the data recording device provides the commitment, the commitment signature, the updated device certificate, and the device related information to the network node manager 132.

The network node manager 132 first verifies the updated device certificate by using the manufacturer's signing public key and then verifies the commitment by using the commitment signature and the device signing public key. After confirmation, the network node manager 132 writes (records) the commitment to the distributed integrity ledger 110. Again the network node manager in the recording stage may be any network node manager.

FIG. 5 illustrates interactions between the user 160, the data center 170, the verifier 180, their respective network node managers 132, 134, 136, the distributed integrity ledger 110 and the distributed identity ledger 120 in a verification stage. In one embodiment, at step 510, the user 160 initiates the verification stage by providing a proof for the user's ID credential to the data center 170. The data center verifies the proof via its network node manager 134 to access the distributed identity ledger 120. After verification of the proof, at step 515, the data center 170 provides the user 160 the commitment and possibly also some meta data. The user 160 obtains a proof for the user's device credential via its network node manager 132 to access the distributed identity ledger 120. The user 160 signs the commitment with the user's signing private key to generate a user's authorization. At step 520, the user 160 provides the verifier 180 the commitment, a proof for the user's ID credential, and the user's authorization. At step 525, the verifier 180 verifies the proof for the user's ID credential, the user's authorization via its network node manager 136 to access the distributed identity ledger 120. Then the verifier 180 looks up the commitment to confirm its existence via its network node manager 136 to access the

distributed integrity ledger 110. After verification and confirmation, the verifier 180 signs the commitment and the user's authorization with the verifier's signing private key to generate the verifier's authorization. At step 530, the verifier 180 provides the data center 170 the commitment, the proof of the verifier's ID credential, the user's authorization, and the verifier's authorization. At step 535, the data center 170 verifies such information via its network node manager 134 to access both distributed ledgers 110, 120. After verification, the data center 170 signs the commitment, the user's authorization, and the verifier's authorization with its signing private key to generate the data center's authorization. At step 540, the data center 170 provides the user 160 and the verifier 180 the data center's authorization. At the same time, the data center 170 also provides the user the encrypted data encryption key. At the same time, the data center 170 also provides the encrypted data piece, the data signature, and the commitment string to the verifier.

Upon receipt of the data center's authorization and the encrypted data encryption key, at step 545, the user obtains encryption private key from the network node manager to decrypt the encrypted data encryption key to eventually derive the data encryption key. At step 550, the user 160 provides the verifier the data encryption key. The verifier then verifies the data center's authorization, decrypts the encrypted data piece to derive the original data piece, verifies authenticity of the data piece by the data signature, and verifies the data signature by the commitment.

Below is an embodiment of pseudo code for implementing the methods described above

.....

ssk: signing private key
25 spk: signing public key
esk: encryption private key
epk: encryption public key
aes_key: data encryption key
cipher key: encrypted data encryption key
30 cipher data: encrypted data piece
sig_com: commitment signature
signature: data signature

carrier: network node manager

```
***carrier (network node manager)***
class Carrier {
5   constructor(id_chain, data_chain) {
      carrier.id_chain = id_chain
      carrier.data_chain = data_chain

      spk, ssk = DigitalSignature.key_generation()
10   carrier.spk = spk
      carrier.ssk = ssk
      carrier.did = carrier.create_did(spk)
   }

15   function create_did() {
      return carrier.id_chain.create_did()
   }

   function issue(did, credential_detail) {
20   credential = Credential.issue(carrier.ssk, (did, credential_detail))
      carrier.add(did, credential)
      return credential
   }

25   function prove(credential) {
      return Credential.proof(credential)
   }

   function sign(document) {
30   return DigitalSignature.sign(carrier.ssk, document)
   }

   function certify_manufacturer(manufacturer_spk) {
35   return carrier.sign(manufacturer_spk)
   }

   function register_device(user_did, device_spk, device_certificate) {
      if device_certificate is valid do
40   // Generate a pair of encryption keys.
      epk, esk = AsymmetricEncryption.key_generation()
      // Issue a credential to bind the device to a user.
      credential = carrier.issue(did, (spk, epk))
      // Generate a proof based on the credential.
45   cred_proof = carrier.prove(credential)
```

```
        carrier.add(user_did, device_spk, device_certificate, credential, epk, esk)

        data_center = Allocate a datacenter.
5      return cred_proof, did, epk, data_center
    end
  }

  function get_cred_proof(user_spk) {
10    find credential from user_spk
    return carrier.prove(credential)
  }

  function decrypt(cipher_key, device_spk) {
15    find esk by device_spk.
    aes_key = AsymmetricEncryption.decrypt(esk, cipher_key)
    return aes_key
  }
}

20 ***data center***
class Datacenter {
  constructor(carrier) {
    spk, ssk = DigitalSignature.key_generation()
25    datacenter.spk = spk
    datacenter.ssk = ssk
    datacenter.carrier = carrier
    datacenter.did = carrier.create_did()
    carrier.issue(did, spk)
30    datacenter.data_chain = carrier.data_chain
  }

  function upload(cipher_data, cipher_key, signature, com, com_r, sig_com, device_certificate) {
    // Check 1: the certificate is valid.
35    if device_certificate is not valid do
      exit()
    end

    // Check 2: the signature sig_com is valid.
40    device_spk = device_certificate.get_public_key()
    if DigitalSignature.verify(device_spk, (com, sig_com)) is not valid do
      exit()
    end

45    // Check 3: the commitment com is valid.
    if Commit.verify(com, com_r, signature) is not valid do
```

```
        exit()
    end

    // If all checks pass, save the record.
5    data_center.add(cipher_data, cipher_key, signature, com, com_r)

    // Pass some information to the data_chain.
    datacenter.data_chain.upload(com, sig_com, device_certificate)
10 }

function sign(document) {
    return DigitalSignature.sign(data_center.ssk, document)
}

15 function select_data(cred_proof) {
    if Credential.verify(cred_proof) is not valid do
        exit()
    end
    // The user select one record and get its com.
20    if data_center.carrier.data_chain.check(com) does not exist do
        exit()
    end
    return com
}

25 function authorize(com, cred_proof_verifier, auth_1, auth_2) {
    // Check 1: the user's cred_proof is valid.
    if Credential.verify(cred_proof_verifier) is not valid do
        exit()
30    end

    // Check 2: the signature auth_1 is valid.
    verifier_spk = cred_proof_verifier.get_public_key()
    if DigitalSignature.verify(verifier_spk, (com, auth_1), auth_2) is not valid do
35        exit()
    end
    auth = data_center.sign(com, auth_1, auth_2)
    return auth
}

40 function get_record(com) {
    find (cipher_data, cipher_key, signature, com, com_r) from com
    return (cipher_data, cipher_key, signature, com, com_r)
}

45 }
```

```
***device (data recording device)***
class Device {
  constructor(spk, ssk, certificate) {
5    device.spk = spk
    device.ssk = ssk
    device.certificate = certificate
  }

10  function register(did, epk, data_center) {
    device.did = did
    device.epk = epk
    device.data_center = data_center
  }

15  function record() {
    video = device.record()

    aes_key = AES.key_generation()
20    cipher_data = AES.encrypt(aes_key, video)
    cipher_key = AsymmetricEncryption.encrypt(device.epk, aes_key)

    signature = DigitalSignature.sign(device.ssk, video)

25    com, com_r = Commit.commit(signature)

    sig_com = DigitalSignature.sign(device.ssk, com)

    device.data_center.upload(cipher_data, cipher_key, signature, com, com, sig_com,
30    device.certificate))
  }
}

***identity_chain (distributed identity ledger)***
35 class IdentityBlockchain {
  constructor() {
  }

  function create_did() {
40    Generate did.
    return did
  }
}

45
***integrity_chain (distributed integrity ledger)***
```

```
class IntegrityBlockchain {
    constructor() {
    }

5    function upload(com, sig_com, device_certificate) {
        // Check 1: the certificate is valid.
        if device_certificate is not valid do
            exit()
        end

10        // Check 2: the signature sig_com is valid.
        device_spk = device_certificate.get_public_key()
        if DigitalSignature.verify(device_spk, com, sig_com) is not valid do
            exit()
15        end

        // Save the commitment if all checks pass.
        integrity_blockchain.save(com)
    }

20    function check(com) {
        return com on integrity_blockchain or not
    }
}

25

***main***
// Initialize two blockchains, the id_chain and the data_chain.
30 id_chain = IdentityBlockchain()
   data_chain = IntegrityBlockchain()

// Initialize a carrier and a manufacturer.
manufacturer_carrier = Carrier(id_chain, data_chain)
35 manufacturer = Manufacturer(manufacturer_carrier)
   certificate = manufacturer_carrier.certify_manufacturer(manufacturer.epk)
   manufacturer.set_certificate(certificate)

// Initialize the user and a device.
40 user_carrier = Carrier(id_chain, data_chain)
   user = User(user_carrier, car_info, driver_info)
   device = manufacturer.manufact()
   // Register the device to a user.
   user.register_device(device, manufacturer)

45 // Record videos.
```

```
device.record()

// Initialize the verifier.
verifier_carrier = Carrier(id_chain, data_chain)
5  verifier = Verifier(verifier_carrier)

// Verify a recorded video.

// 1. 3 party authorization
10 // 1-1. Fetch the commitment from the data center.
    data_center = device.data_center
    cred_proof_user = user.get_cred_proof()
    com = data_center.select_data(cred_proof_user)

15 // 1-2. User authorizes access.
    auth_1 = user.authorize(com)

// 1-3. Verifier authorizes access.
    auth_2, cred_proof_verifier = verifier.authorize(com, cred_proof_user, auth_1)
20 // 1-4. Datacenter authorizes access.
    auth = data_center.authorize(com, cred_proof_verifier, auth_1, auth_2)
    (cipher_data, cipher_key, signature, com, com_r) = data_center.get_record(com)

25 // 2. Deliver the record.
    aes_key = user.decrypt_aes_key(auth, auth_2, cred_proof_verifier, cipher_key)
    cred_proof_device = user.carrier.get_cred_proof(user.device.spk)

// 3. Verify the integrity of data.
30 verifier.decrypt_video(aes_key, cipher_data, cred_proof_device, signature, com)

***manufacturer***
class Manufacturer {
35   constructor(carrier) {
        spk, ssk = DigitalSignature.key_generation()
        manufacturer.spk = spk
        manufacturer.ssk = ssk
        manufacturer.carrier = carrier
40     manufacturer.did = carrier.create_did()
        carrier.issue(did, spk)
    }

    function sign(document) {
45     return DigitalSignature.sign(manufacturer.ssk, document)
    }
}
```

```
function set_certificate(certificate) {
    manufacturer.certificate = certificate
}
5
function manufact() {
    spk, ssk = DigitalSignature.key_generation()
    certificate = manufacturer.sign(spk)
    device = Device(spk, ssk, certificate)
10    return device
}

function register(user_carrier, device_spk, device_certificate, user_car_info, user_driver_info)
{
15    if device_certificate is valid do
        // Redirect user to login to user_carrier.
        user_did = login()
        cred_proof, did, epk, data_center = user_carrier.register_device(user_did, device_spk,
device_certificate)
20        if Credential.verify(cred_proof) == true do
            return did, epk, data_center
        end
    end
}
25 }

***user***
class User {
    constructor(carrier, car_info, driver_info) {
30        user.car_info = car_info
        user.driver_info = driver_info
        spk, ssk = DigitalSignature.key_generation()
        user.spk = spk
        user.ssk = ssk
35        user.carrier = carrier
        user.did = carrier.create_did()
        carrier.issue(did, spk)
    }

40    function register_device(device, manufacturer) {
        Plug device to a pc.
        Connect to manufacturer's website.

        did, epk, data_center = manufacturer.register(user.carrier, device.spk, device.certificate,
45    user.car_info, user.driver_info)
```

```
        device.register(did, epk, data_center)
        user.add(device)
    }

5    function get_cred_proof() {
        return user.carrier.get_cred_proof(user.spk)
    }

    function sign(document) {
10        return DigitalSignature.sign(user.ssk, document)
    }

    function authorize(com) {
        if user.carrier.data_chain.check(com) exists do
15        return user.sign(com)
        end
    }

    function decrypt_aes_key(auth, auth_2, cred_proof_verifier, cipher_key) {
20        // Check 1: the user's cred_proof is valid.
        if Credential.verifier(cred_proof_verifier) is not valid do
            exit()
        end

25        // Check 2: the signature auth_1 is valid.
        verifier_spk = cred_proof_verifier.get_public_key()
        if DigitalSignature.verify(verifier_spk, (com, auth_1), auth_2) is not valid do
            exit()
        end

30        // Check3: the signature auth is valid.

        data_center_spk = device.data_center.spk
        if DigitalSignature.verify(data_center_spk, (com, auth_1, auth_2), auth) is not valid do
35        exit()
        end

        aes_key = user.carrier.decrypt(cipher_key, device_spk)
        return aes_key
40    }
}

***verifier***
45 class Verifier {
    constructor(carrier) {
```

```
    spk, ssk = DigitalSignature.key_generation()
    verifier.spk = spk
    verifier.ssk = ssk
    verifier.carrier = carrier
5    verifier.did = carrier.create_did()
    carrier.issue(did, spk)
}

function sign(document) {
10    return DigitalSignature.sign(verifier.ssk, document)
}

function authorize(com, cred_proof_user, auth_1) {
    // Check 1: the user's cred_proof is valid.
15    if Credential.verifier(cred_proof_user) is not valid do
        exit()
    end

    // Check 2: the signature auth_1 is valid.
20    user_spk = cred_proof_user.get_public_key()
    if DigitalSignature.verify(user_spk, com, auth_1) is not valid do
        exit()
    end
    auth_2 = verifier.sign(com, auth_1)
25    cred_proof_verifier = verifier.carrier.get_cred_proof(verifier.spk)
    if verifier.carrier.data_chain.check(com) does not exist do
        exit()
    end
    return auth_2, cred_proof_verifier
30 }

function decrypt_video(aes_key, cipher_data, cred_proof_device, signature, com) {
    // Check 1: the user's device cred_proof_device is valid.
    if Credential.verifier(cred_proof_device) is not valid do
35        exit()
    end

    // Check 2: the signature auth_1 is valid.
    video = AES.decrypt(aes_key, cipher_data)
40    device_spk = cred_proof_device.get_public_key()
    if DigitalSignature.verify(device_spk, video, signature) is not valid do
        exit()
    end

    // Check 3: the record exists on data_chain.
45    if verifier.carrier.data_chain.check(com) does not exist do
```

```
        exit()
    end
    Accept the video with confidence.
5  }
```

The foregoing description of embodiments is provided to enable any person skilled in the art to make and use the subject matter. Various modifications to these embodiments will be readily apparent to those skilled in the art, and the novel principles and subject matter disclosed herein may be applied to other embodiments without the use of the innovative faculty. The claimed subject matter set forth in the claims is not intended to be limited to the embodiments shown herein but is to be accorded the widest scope consistent with the principles and novel features disclosed herein. It is contemplated that additional embodiments are within the spirit and true scope of the disclosed subject matter. Thus, it is intended that the present invention covers modifications and variations that come within the scope of the appended claims and their equivalents.

WHAT IS CLAIMED IS:

1. A distributed ledger based cryptographic method for a network node manager to improve data integrity, comprising:
 - receiving, from a data recording device managed by a user, an original device certificate signed by a manufacturer of the data recording device;
 - verifying the original device certificate; and
 - verifying a user identification or generating the user identification in a distributed identity ledger.
2. The method of claim 1, further comprising:
 - generating an encryption private key and the encryption public key for the data recording device.
3. The method of claim 2, further comprising:
 - generating a user's device credential comprising an encryption public key, and a proof for the user's device credential.
4. The method of claim 1, wherein the data recording device is a digital video recorder.
5. A distributed ledger based cryptographic method for a network node manager to improve data integrity, comprising:
 - receiving, from a data recording device managed by a user, an updated device certificate signed by a manufacturer of the data recording device, a commitment generated to lock a data signature that is generated by signing a data piece with a device signing private key, and a commitment signature generated by signing the commitment with the device signing private key;
 - verifying the updated device certificate by using a manufacturer's signing public key;
 - verifying the commitment by using the commitment signature; and
 - recording, if verification is valid, the commitment in a distributed integrity ledger maintained by a first distributed transaction consensus network.

6. The method of claim 5, further comprising:
 - receiving the commitment and a request for confirming the commitment recorded in the distributed integrity ledger; and
 - confirming with the distributed integrity ledger that the commitment exists.
7. The method of claim 5, further comprising:
 - receiving a request for an encryption private key of the data recording device and an authorization from the user;
 - verifying the authorization from the user; and
 - providing the encryption private key to the user for decrypting the encrypted data encryption key.
8. A distributed ledger based cryptographic method for a data recording device to improve data integrity, comprising:
 - providing, to a network node manager, a device signing public key previously stored in the data recording device; and
 - receiving, from the network node manager, an encryption public key and a user identification recorded in a distributed identity ledger maintained by a second distributed transaction consensus network.
9. The method of claim 8, further comprising:
 - providing, to a manufacturer of the data recording device, an original device certificate stored in the data recording device;
 - receiving, from the manufacturer of the data recording device, an updated device certificate, including the user identification and the encryption public key, signed by the manufacturer; and
 - wherein the device signing public key are provided to the network node manager through the manufacturer and the encryption public key and the user identification are received from the network node manager through the manufacturer.
10. The method of claim 8, further comprising:

generating a data encryption key for a data piece;
encrypting the data piece with the data encryption key;
encrypting the data encryption key with the encryption public key;
generating a data signature by signing the data piece with a device signing private key;
generating a commitment string;
generating a commitment to be recorded in a distributed integrity ledger maintained by a first distributed transaction consensus network, by providing the data signature and the commitment string; and
generating a commitment signature by signing the commitment with the device signing private key.

11. The method of claim 8, wherein the data recording device is a digital video recorder.

12. A distributed ledger based cryptographic method for a verifier to improve data integrity, comprising:

receiving an encrypted data piece, a data encryption key, a data signature, a commitment, and a commitment string;

decrypting the encrypted data piece with the data encryption key to obtain the data piece recorded by a data recording device;

verifying authenticity of the data piece by using the data signature and a device signing public key;

verifying that the commitment is recorded in a distributed integrity ledger maintained by a first distributed transaction consensus network; and

verifying the commitment by using the data signature and the commitment string.

13. The method of 12, further comprising:

receiving an authorization from a user of the data recording device or a data center
storing the encrypted data piece, the commitment, the data signature, and the commitment string;
and

verifying the authorization with a user's signing public key or a data center's signing public key.

14. A data recording device, comprising:
 - a recording module for recording a data piece;
 - a memory module for storing an original device certificate signed by a manufacturer, a device signing public key, and a device signing private key; and
 - a processor module for generating a data encryption key, encrypting the data piece with the data encryption key, encrypting the data encryption key with the encryption public key.
15. The device of claim 14, wherein the processor module generates a data signature by signing the data piece with the device signing private key, a commitment string, and a commitment to lock the data signature.
16. The device of claim 14, wherein the recording module is a digital video recorder.