



UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #
19/564,935

RECEIPT DATE / TIME
03/12/2026 03:34:54 PM Z ET

ATTORNEY DOCKET #
Q318247

Title of Invention

CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD

Application Information

APPLICATION TYPE	Utility - Nonprovisional Application under 35 USC 111(a)	PATENT #	-
CONFIRMATION #	3313	FILED BY	Hiroko Derello
PATENT CENTER #	74831260	FILING DATE	-
CUSTOMER #	23373	FIRST NAMED INVENTOR	Sang Yeob LEE
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	Hyunseok Park

Documents

TOTAL DOCUMENTS: 19

DOCUMENT	PAGES	DESCRIPTION	SIZE (KB)
Q318247TransmittalNewApp asfiled.pdf	1	Transmittal of New Application	164 KB
Q318247ADSasfiled.pdf	5	Application Data Sheet	169 KB
Warning: This is not a USPTO supplied ADS fillable form. Data in the form cannot be automatically loaded to other USPTO systems.			
Q318247Specificationasfiled-APP.TEXT.docx	96	Application body structured text document	70 KB
Warning: Document has [Balance SBCS characters and DBCS characters] option enabled and may result in reduction of pages, however, no data or text has been modified.			
Q318247Drawingsasfiled.pdf	5	Drawings-only black and white line drawings	235 KB

Q318247REF1KR1020150107796.pdf		32	Foreign Reference	553 KB
Q318247REF3KR101727525.pdf		14	Foreign Reference	870 KB
Q318247REF6KR102461050.pdf		25	Foreign Reference	868 KB
Q318247AssignmentwithDeclaration.pdf		3	Oath or Declaration filed	463 KB
Q318247REF8NPLKRNOA.pdf		5	Other reference- Patent/Application/Search Documents	85 KB
Q318247CertificationofMicroEntityStatusasfiled.pdf		1	Certification of Micro Entity (Gross Income Basis)	140 KB
Q318247REF2KR101284353.pdf		23	Foreign Reference	554 KB
Q318247REF7NPLKROA.pdf		15	Other reference- Patent/Application/Search Documents	228 KB
Q318247IDSandSB08asfiled.pdf		5	-	188 KB
Q318247IDSandSB08asfiled-TRAN.LET.pdf	(1-2)	2	Transmittal Letter	92 KB
Q318247IDSandSB08asfiled-IDS.pdf	(3-4)	2	Information Disclosure Statement (IDS) Form (SB08)	135 KB
Q318247IDSandSB08asfiled-IDS.SIZEFEE.pdf	(5-5)	1	Assertion regarding Information Disclosure Statement (IDS) Size Fee (SB/08C)	137 KB
Q318247REF4KR102062688.pdf		43	Foreign Reference	643 KB
Q318247REF5KR102307863.pdf		27	Foreign Reference	647 KB
Q318247PowerofAttorneyasfiled.pdf		2	Power of Attorney	525 KB
Q318247Specificationasfiled.pdf		96	Auxiliary PDF of Application	491 KB

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
Q318247TransmittalNewAppasfiled.pdf	6E9923E061B457A8F15D58573FC955C75248BE5876C8A89FF BD0088DC97B0900144CC4F7694B49F0FAA9D004115391AFFB 7C7276AD6651737359BD3718298569
Q318247ADSasfiled.pdf	725F464E03A816D94A22DDD250B9D9CB227298D10D7FD6C3 95BF015519B1E018115D69EF9817BE9CCF5942892A5C5057C 65F7BC1CB056DAC0EADB5D8F200292F
Q318247Specificationasfiled-APP.TEXT.docx	B063B25814818FDBF00EB00F056BB6B959C1AE95AA182A858 01F0D6C201870612C5938CE2F972B22309A581A6A15F70D333 57223CBA8D573F7AA3CB8A470A1B5
Q318247Drawingsasfiled.pdf	A39A1864519E1D44848A872D5114748126EC824B11E58EFC5 C9C5456D293E2FABE9A9ACFBD6EF8F7F5DAAA11AF5DD70D B3F3AB633FF225F238AB93BA8016DD90
Q318247REF1KR1020150107796.pdf	B79B507A6FB7A833CAA6D12F1CB0B96EFE6EA4B008E6FFA4 13A77AABEF4F1BF4492D9A5847497DB8657588591D877D6AB F7C8BD616EDE7395010CE66C716BAC9
Q318247REF3KR101727525.pdf	0E4E326CC54E60B650DC8FBA069F04DDD6F631B74C167CC6 F516C42C6D28F9B64ACE4A20BA4FA74A18411093221E9B235 4EF72223A5200C66FB5782E0A15F662
Q318247REF6KR102461050.pdf	EDF6AAA9F24FB5989B43CE087660E03425F5FCFBE332394BA 1CB8704DF975EBC9739CA17B3D096941806305A4AB31F7398 8B9062D1CFB54D5BFF5519FAABB180
Q318247AssignmentwithDeclaration.pdf	DA5A8D09E831F1DFE8C90F45EC1C806AE7A6BBED8E449950 8CC2D4C76AC70AA7CDAD9B3D44CB1324F274D6FA4F6DD40 7CCA303A9EAB60A00ADA7D8ACBC0C1775
Q318247REF8NPLKRNOA.pdf	AEBA1A0747C41D59576E6F864B042ADCC1489596D06D118D 7D3A141292075F5665019C21FCE0C6772BA664214A96075AD2 EC88074A619DE5BC032051369BA688
Q318247CertificationofMicroEntityStatusasfiled.pdf	7A854F930ADCE60521BF1926DCD92C20CCEA68C1E70FC7AF B952726F44EFE34BCCB7B54018F3C3FC7AC306478DFB11195 7FFACDAEAF27D559883CFEC48269F35
Q318247REF2KR101284353.pdf	767522836A695218B7E512D602B7A1AA7439BD0290FDB436A6 6D6B84E8AAD726DC5C3C8470752FBC446F8B0973F109FA4A A32AF4D362330FB81FF14E71E978BC
Q318247REF7NPLKROA.pdf	8D366641996216CCA4AF6208261043DDA74538CA806B77F4B 20DBEAF6C030548FCC5C102E75698F67F11E806787681CBAD 2E52EC677A59BACD8B5E62988EE244

Q318247IDSandSB08asfiled.pdf	C9CACFA9D0F1B474B85C8CCCFC266E34634A0257BD5060A C2AC742F36C0EED790DF149B0AB7596DE5C51D15D4A915B7 AAE3B9CEFC5D9B36B2132644589BD630B
Q318247IDSandSB08asfiled- TRAN.LET.pdf	178FC0D11B8F1324C7FF90FA3F63A0879EE6FDB0082C2B025 43A216CA8A03854A7E4645F41A5D8BDF48A83A893E6C8C023 5C1CBD542035C2A1FAD4A0DB9E2C02
Q318247IDSandSB08asfiled- IDS.pdf	99FCBBEB24EC8565F6626027582176187EC92145ADBBCDCD 5DF1E4CA49E82195D3D73708AF4676220EE36F2FAD5B4BBA EEBC8E5E9E277DBC72D5AE1D28C04C60
Q318247IDSandSB08asfiled- IDS.SIZEFEE.pdf	78CB0D25487E73C9A4BE41D44875B8096F17250EC AA57B003 0EDD7000B28ACB8405E400F8992527E56F449DA80B0639F75 53FF7024B5C852E794B96FF918DBEA
Q318247REF4KR102062688.pdf	83AD5F2F22971D66A4C95962EC03271F3F18B70D1AF26F4AB 046B7C057FD48B06B0CF672370CA088CAD2A0C08C40128D2 AF6EB52D106FAD114C351C262D6ACA1
Q318247REF5KR102307863.pdf	2EE9DC511CE583760861B5FDD8D396D9F2B8934B7BEF12467 05F6389C26BA3B84E7C384DFE440FF04F5A04E4AA141AFAB8 1E7B9796E208A8D4EA3AA7209E1A92
Q318247PowerofAttorneyasfiled.pdf	3A644CF38A97C7CC0FAC3CBC35C89C445976B70EBEAB1464 869CF2F260626C8B966A1C27581B448946B9FE5492271ED2D2 0DA8AAFE95F76E5EF35C37740285E4
Q318247Specificationasfiled.pdf	468BDDDB61D18E2A22641E8D01813CCE8B4C29E19F6F017B6 21A6CCF2C687D83C4A3292F33E2FC003E1DBCCEF0C2F24C2 649EFDD0AF5DBC8ECBDCC0CDB3B60FEC

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UTILITY PATENT APPLICATION TRANSMITTAL <i>(Only for new nonprovisional applications under 37 CFR 1.53(b))</i>		Attorney Docket No. Q318247	
		First Named Inventor Sang Yeob LEE	
		Title	CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD
		Priority Mail Express® Label No.	
APPLICATION ELEMENTS <i>See MPEP chapter 600 concerning utility patent application contents.</i>		Commissioner for Patents ADDRESS TO: P.O. Box 1450 Alexandria VA 22313-1450	
1. ☐ Fee Transmittal Form (PTO/SB/17 or equivalent) 2. ☐ Applicant asserts small entity status. See 37 CFR 1.27 3. ☒ Applicant certifies micro entity status. See 37 CFR 1.29. Applicant must attach form PTO/SB/15A or B or equivalent. 4. ☒ Specification [Total Pages <u>96</u>] Both the claims and abstract must start on a new page. (See MPEP § 608.01(a) for information on the preferred arrangement) 5. ☒ Drawing(s) (35 U.S.C. 113) [Total Sheets <u>5</u>] 6. Inventor's Oath or Declaration [Total Pages <u>3</u>] (including substitute statements under 37 CFR 1.64 and assignments serving as an oath or declaration under 37 CFR 1.63(e)) a. ☒ Newly executed (original or copy) b. ☐ A copy from a prior application (37 CFR 1.63(d)) 7. ☒ Application Data Sheet. *See note below. See 37 CFR 1.76 (PTO/AIA/14 or equivalent) 8. CD-ROM or CD-R in duplicate, large table or Computer Program (Appendix) ☐ Landscape Table on CD 9. ☐ Nucleotide and/or Amino Acid Sequence Submission (ST.26 Sequence Listing (xml))		ACCOMPANYING APPLICATION PAPERS	
		10. ☐ Assignment Papers (cover sheet & document(s)) Name of Assignee <u>Bo Mi KIM</u> 11. ☐ 37 CFR 3.73(c) Statement ☒ Power of Attorney (when there is an assignee) 12. ☐ English Translation Document (if applicable) 13. ☒ Information Disclosure Statement (PTO/SB/08 or PTO-1449) ☒ Copies of citations attached 14. ☐ Preliminary Amendment 15. ☐ Return Receipt Postcard (MPEP § 503)(Should be specifically itemized) 16. ☐ Certified Copy of Priority Document(s) (if foreign priority is claimed) 17. ☐ Nonpublication Request Under 35 U.S.C. 122(b)(2)(B)(i). Applicant must attach form PTO/SB/35 or equivalent. 18. ☒ Other: Certification of Micro Entity Status	
*Note: (1) Benefit claims under 37 CFR 1.78 and foreign priority claims under 1.55 must be included in an Application Data Sheet (ADS). (2) For applications filed under 35 U.S.C. 111, the application must contain an ADS specifying the applicant if the applicant is an assignee, person to whom the inventor is under an obligation to assign, or person who otherwise shows sufficient proprietary interest in the matter. See 37 CFR 1.46(b).			
☒ The USPTO is directed and authorized to charge all required fees, except for the Issue Fee and the Publication Fee, to Deposit Account No. 19-4880. Please also credit any overpayments to said Deposit Account.			
19. CORRESPONDENCE ADDRESS			
☒ The address associated with Customer Number: WASHINGTON OFFICE 23373 CUSTOMER NUMBER OR ☐ Correspondence address below			
Signature /Hyunseok Park/		Date March 12, 2026	
Name Hyunseok Park		Registration No. 74,833	

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	Q318247
		Application Number	Not Yet Assigned
Title of Invention	CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD		

The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76. This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.

Secrecy Order 37 CFR 5.2:

☐ Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)

Inventor Information:

Inventor 1						
Legal Name						
Prefix	Given Name	Middle Name	Family Name	Suffix		
	Sang Yeob		LEE			
Residence Information (Select One) ☐ US Residency ☒ Non US Residency ☐ Active US Military Service						
City	Seogwipo-si	Country of Residence	KR			
Mailing Address of Inventor:						
Address 1	34-1, 516-ro 97beon-gil					
Address 2						
City	Seogwipo-si, Jeju-do	State/Province				
Postal Code	63604	Country	KR			

Correspondence Information:

Enter either Customer Number or complete the Correspondence Information section below. For further information see 37 CFR 1.33(a).	
☐ An Address is being provided for the correspondence information of this application.	
Customer Number	23373
Email Address	sughrue@sughrue.com

Application Information:

Title of the Invention	CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD		
Attorney Docket Number	Q318247	Small Entity Status Claimed	☐
Application Type	Nonprovisional		
Subject Matter	Utility		
Total Number of Drawing Sheets (if any)	5	Suggested Figure for Publication (if any)	

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	Q318247
		Application Number	Not Yet Assigned
Title of Invention	CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD		

Filing By Reference:

Only complete this section when filing an application by reference under 35 U.S.C. 111(c) and 37 CFR 1.57(a). Do not complete this section if application papers including a specification and any drawings are being filed. Any domestic benefit or foreign priority information must be provided in the appropriate section(s) below (i.e., "Domestic Benefit/National Stage Information" and "Foreign Priority Information"). For the purposes of a filing date under 37 CFR 1.53(b), the description and any drawings of the present application are replaced by this reference to the previously filed application, subject to conditions and requirements of 37 CFR 1.57(a).		
Application number of the previously filed application	Filing date (YYYY-MM-DD)	Intellectual Property Authority or Country

Publication Information:

☐ Request Early Publication (Fee required at time of Request 37 CFR 1.219)
☐ Request Not to Publish. I hereby request that the attached application not be published under 35 U.S.C. 122(b) and certify that the invention disclosed in the attached application has not and will not be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

Representative Information:

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32). Either enter Customer Number or complete the Representative Name section below. If both sections are completed the Customer Number will be used for the Representative Information during processing.			
Please Select One:	☒ Customer Number	☐ US Patent Practitioner	☐ Limited Recognition (37 CFR 11.9)
Customer Number	23373		

Domestic Benefit/National Stage Information:

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, 365(c), or 386(c) or indicate National Stage entry from a PCT application. Providing benefit claim information in the Application Data Sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78. When referring to the current application, please leave the "Application Number" field blank.			
Prior Application Status			
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	Q318247
		Application Number	Not Yet Assigned
Title of Invention	CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD		

Foreign Priority Information:

This section allows for the applicant to claim priority to a foreign application. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55. When priority is claimed to a foreign application that is eligible for retrieval under the priority document exchange program (PDX) the information will be used by the Office to automatically attempt retrieval pursuant to 37 CFR 1.55(i)(1) and (2). Under the PDX program, applicant bears the ultimate responsibility for ensuring that a copy of the foreign application is received by the Office from the participating foreign intellectual property office, or a certified copy of the foreign priority application is filed, within the time period specified in 37 CFR 1.55(g)(1).

Application Number	Country	Filing Date (YYYY-MM-DD)	Access Code (if applicable)
10-2025-0034578	KR	2025-03-18	50FF

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications

☐ This application (1) claims priority to or the benefit of an application filed before March 16, 2013 and (2) also contains, or contained at any time, a claim to a claimed invention that has an effective filing date on or after March 16, 2013.

NOTE: By providing this statement under 37 CFR 1.55 or 1.78, this application, with a filing date on or after March 16, 2013, will be examined under the first inventor to file provisions of the AIA.

Authorization or Opt-Out of Authorization to Permit Access:

When this Application Data Sheet is properly signed and filed with the application, applicant has provided written authority to permit a participating foreign intellectual property (IP) office access to the instant application-as-filed (see paragraph A in subsection 1 below) and the European Patent Office (EPO) access to any search results from the instant application (see paragraph B in subsection 1 below).

Should applicant choose not to provide an authorization identified in subsection 1 below, applicant **must opt-out** of the authorization by checking the corresponding box A or B or both in subsection 2 below.

NOTE: This section of the Application Data Sheet is **ONLY** reviewed and processed with the **INITIAL** filing of an application. After the initial filing of an application, an Application Data Sheet cannot be used to provide or rescind authorization for access by a foreign IP office(s). Instead, Form PTO/SB/39 or PTO/SB/69 must be used as appropriate.

1. Authorization to Permit Access by a Foreign Intellectual Property Office(s)

A. Priority Document Exchange (PDX) - Unless box A in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the European Patent Office (EPO), the Japan Patent Office (JPO), the Korean Intellectual Property Office (KIPO), the State Intellectual Property Office of the People's Republic of China (SIPO), the World Intellectual Property Organization (WIPO), and any other foreign intellectual property office participating with the USPTO in a bilateral or multilateral priority document exchange agreement in which a foreign application claiming priority to the instant patent application is filed, access to: (1) the instant patent application-as-filed and its related bibliographic data, (2) any foreign or domestic application to which priority or benefit is claimed by the instant application and its related bibliographic data, and (3) the date of filing of this Authorization. See 37 CFR 1.14(h)(1).

B. Search Results from U.S. Application to EPO - Unless box B in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the EPO access to the bibliographic data and search results from the instant patent application when a European patent application claiming priority to the instant patent application is filed. See 37 CFR 1.14(h)(2).

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	Q318247
		Application Number	Not Yet Assigned
Title of Invention	CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD		

The applicant is reminded that the EPO's Rule 141(1) EPC (European Patent Convention) requires applicants to submit a copy of search results from the instant application without delay in a European patent application that claims priority to the instant application.

2. Opt-Out of Authorizations to Permit Access by a Foreign Intellectual Property Office(s)

- ☐ A. Applicant **DOES NOT** authorize the USPTO to permit a participating foreign IP office access to the instant application-as-filed. If this box is checked, the USPTO will not be providing a participating foreign IP office with any documents and information identified in subsection 1A above.
- ☐ B. Applicant **DOES NOT** authorize the USPTO to transmit to the EPO any search results from the instant patent application. If this box is checked, the USPTO will not be providing the EPO with search results from the instant application.

NOTE: Once the application has published or is otherwise publicly available, the USPTO may provide access to the application in accordance with 37 CFR 1.14.

Applicant Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

Applicant

If the applicant is the inventor (or the remaining joint inventor or inventors under 37 CFR 1.45), this section should not be completed. The information to be provided in this section is the name and address of the legal representative who is the applicant under 37 CFR 1.43; or the name and address of the assignee, person to whom the inventor is under an obligation to assign the invention, or person who otherwise shows sufficient proprietary interest in the matter who is the applicant under 37 CFR 1.46. If the applicant is an applicant under 37 CFR 1.46 (assignee, person to whom the inventor is obligated to assign, or person who otherwise shows sufficient proprietary interest) together with one or more joint inventors, then the joint inventor or inventors who are also the applicant should be identified in this section.

☒ Assignee	☐ Legal Representative under 35 U.S.C. 117	☐ Joint Inventor
☐ Person to whom the inventor is obligated to assign.	☐ Person who shows sufficient proprietary interest	

If applicant is the legal representative, indicate the authority to file the patent application, the inventor is:

Name of the Deceased or Legally Incapacitated Inventor:	
If the Applicant is an Organization check here.	☐

Organization Name				
Prefix	Given Name	Middle Name	Family Name	Suffix
	Bo Mi		KIM	

Mailing Address Information For Applicant:

Address 1	34-1, 516-ro 97beon-gil		
Address 2			
City	Seogwipo-si, Jeju-do	State/Province	

Application Data Sheet 37 CFR 1.76		Attorney Docket Number	Q318247
		Application Number	Not Yet Assigned
Title of Invention	CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD		
Country	KR	Postal Code	63604
Phone Number		Fax Number	
Email Address			

Assignee Information including Non-Applicant Assignee Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.				
Assignee				
Complete this section if assignee information, including non-applicant assignee information, is desired to be included on the patent application publication. An assignee-applicant identified in the "Applicant Information" section will appear on the patent application publication as an applicant. For an assignee-applicant, complete this section only if identification as an assignee is also desired on the patent application publication.				
If the Assignee or Non-Applicant Assignee is an Organization check here.				☐
Organization Name				
Prefix	Given Name	Middle Name	Family Name	Suffix
	Bo Mi		KIM	
Mailing Address Information For Assignee including Non-Applicant Assignee:				
Address 1	34-1, 516-ro 97beon-gil			
Address 2				
City	Seogwipo-si, Jeju-do	State/Province		
Country	KR	Postal Code	63604	
Phone Number		Fax Number		
Email Address				

Signature:

NOTE: This Application Data Sheet must be signed in accordance with 37 CFR 1.33(b). However, if this Application Data Sheet is submitted with the INITIAL filing of the application and either box A or B is not checked in subsection 2 of the "Authorization or Opt-Out of Authorization to Permit Access" section, then this form must also be signed in accordance with 37 CFR 1.14(c).			
This Application Data Sheet must be signed by a patent practitioner if one or more of the applicants is a juristic entity (e.g., corporation or association). If the applicant is two or more joint inventors, this form must be signed by a patent practitioner, all joint inventors who are the applicant, or one or more joint inventor-applicants who have been given power of attorney (e.g., see USPTO Form PTO/AIA/81) on behalf of all joint inventor-applicants.			
See 37 CFR 1.4(d) for the manner of making signatures and certifications.			
Signature	/Hyunseok Park/	Date (YYYY-MM-DD)	2026-03-12
Name	Hyunseok Park	Registration Number	74,833

CLOUD BLOCKCHAIN SYSTEM AND METHOD FOR CONTROLLING THE SAME

CROSS-REFERENCE TO RELATED APPLICATION

5 This application claims priority to and the benefit of Korean Patent Application No. 10-2025-0034578, filed on March 18, 2025, the disclosure of which is incorporated herein by reference in its entirety.

BACKGROUND

10 1. Field of the Invention

 The present disclosure relates to a cloud blockchain system and a control method thereof. More particularly, the present disclosure relates to a cloud blockchain system and a control method thereof, configured to encrypt, at a user terminal, original data using an authentication key to generate an encrypted block, to register request information, including the encrypted block and a timestamp, on a gate server, to access, by a cloud pocket, the gate server to verify the registered request information, to restore the original data by decrypting the encrypted block contained in the request information, and to manage the blockchain by either storing the restored original data as a new block or modifying existing data with the original data.

20

2. Discussion of Related Art

 Blockchain is a distributed data storage technology that encapsulates data in blocks, connects them in a chain format, and replicates and stores the data across

numerous computers simultaneously. Blockchain is also referred to as a public transaction ledger.

This blockchain technology provides excellent security and offers advantages such as high data stability, the prevention of data loss, and the impossibility of
5 unauthorized modification.

Due to these advantages, blockchain is utilized for virtual currencies and connected via Peer-to-Peer (P2P) networks. However, time delays occur during the generation and transmission of coins. Furthermore, once a coin and its associated keys are generated, the viewing keys and data should be distributed to all P2P-
10 connected devices. This process results in a time delay ranging from as little as 30 minutes to more than 2 hours. Since these delays are physical in nature, blockchain-based coins may reduce but cannot entirely eliminate them, which presents challenges in managing such coins as a functional currency.

[Related Art Document]

15 [Patent Document]

(Patent Document 1) Korean Patent Registration No. 10-1727525
(Registration date: April 11, 2017)

SUMMARY OF THE INVENTION

20 An object of the present disclosure is to provide a cloud blockchain system and a method for controlling the same. The system and method are configured to: encrypt, at a user terminal, original data using an authentication key to generate an encrypted block; register request information, including the encrypted block and a

timestamp, on a gate server; verify, by a cloud pocket, the request information by accessing the gate server and restore the original data by decrypting the encrypted block contained in the request information; and store the restored original data as a new block in a blockchain or modify existing data with the original data.

5 Another object of the present disclosure is to provide a cloud blockchain system and a method for controlling the same configured to: log in, at a user terminal, to a web server using an ID and a password and receive an authentication block provided by the web server in response to a request; register the received authentication block on a gate server; verify authentication, at a cloud pocket, by
10 checking the consistency between the authentication block registered by the user terminal and an authentication block registered by the web server on the gate server; generate and register an authentication key with a set expiration period on the gate server upon successful authentication; and register original data in the form of a block into a cloud blockchain by interworking with the gate server and the cloud pocket,
15 utilizing the authentication key registered on the gate server by the user terminal.

A cloud blockchain system according to an embodiment of the present disclosure includes a gate server that mediates between a user terminal and a cloud pocket. The user terminal may be configured to access one of a plurality of gate servers to be issued an authentication key from the cloud pocket, register an
20 authentication block provided by a web server into one of a plurality of tables within the connected gate server, and receive an authentication key block corresponding to the registered authentication block. The cloud pocket may be configured to: access the gate server at a preset interval and check whether data is registered in the plurality

of tables; when data is registered in at least one table, verify the data; if the data is an authentication block, interwork with the gate server to compare the authentication block registered by the web server with the authentication block registered by the user terminal for consistency; upon a match, verify the ID and password within the authentication block and generate an authentication key based on the ID, password, current time, and a preset expiration period; generate a user account encryption/decryption key consisting of a inquiry account key, a modification account key, and an computation key; and register an authentication key block, including the generated authentication key and the user account encryption/decryption key, into a table of the gate server.

In an exemplary embodiment, the user terminal may log into the web server using an identifier (ID) and a password. The web server may be configured to: generate an authentication block comprising the ID and the password based on an authentication block generation request from the user terminal; transmit the generated authentication block to the user terminal; and register the generated authentication block on a gate server interworking with the web server. The user terminal may be configured to receive the authentication block transmitted from the web server in response to the authentication block generation request.

According to an embodiment of the present disclosure, a method for controlling a cloud blockchain system includes: accessing, by a user terminal, one of a plurality of gate servers to be issued an authentication key from a cloud pocket; registering, by the user terminal, an authentication block provided by a web server into one of a plurality of tables within the connected gate server; accessing, by the cloud

pocket, the gate server at a preset interval to check whether data is registered in the plurality of tables; verifying, by the cloud pocket, the registered data when it is determined that data is registered in at least one of the plurality of tables; comparing, by the cloud pocket, the authentication block registered by the web server with the authentication block registered by the user terminal for consistency by interworking with the gate server when the verified data is an authentication block; verifying, by the cloud pocket, an ID and a password included in the authentication block and generating an authentication key based on the verified ID, the password, a current time, and a preset expiration period when the authentication blocks match; generating, by the cloud pocket, a user account encryption/decryption key comprising: a inquiry account key of a preset size for interpreting encrypted data, a modification account key of a preset size for encrypting data, and an computation key of a preset size configured to prevent pattern interpretation using an arithmetic coding method; registering, by the cloud pocket, an authentication key block comprising the generated authentication key and the generated user account encryption/decryption key into the table of the gate server; and confirming, by the user terminal, the authentication key block registered in the table of the gate server.

According to an exemplary embodiment related to the present disclosure, the method may further include: generating, by the user terminal, an encrypted block by encrypting original data to be registered in the cloud blockchain using the user account encryption/decryption key included in the authentication key block, in order to store data in the cloud blockchain managed by the cloud pocket or to query data registered in the cloud blockchain; generating, by the user terminal, a request statement including

the generated encrypted block and the authentication key included in the authentication key block; accessing, by the user terminal, the gate server; registering, by the user terminal, the generated request statement into one of a plurality of tables within the connected gate server; accessing, by the cloud pocket, the gate server at the preset
5 interval to check whether data is registered in the plurality of tables; verifying, by the cloud pocket, the data registered in the table when it is determined that data is registered in at least one of the plurality of tables; performing, by the cloud pocket, a verification function on the request statement when the data registered in the table is the request statement; and performing, by the cloud pocket, any one of a block
10 generation function, a block modification function, and a block inquiry function in the cloud blockchain based on original data decrypted from the request statement, when the verification of the request statement is successfully completed.

According to an exemplary embodiment related to the present disclosure, the performing of the verification function on the request statement includes: verifying,
15 by the cloud pocket, whether a time difference between a current time and a timestamp included in the request statement exceeds a preset reference value to validate the data; registering, by the cloud pocket, information indicating that validity verification of the data has failed into the table of the gate server and terminating a connection with the gate server, when the request statement is determined to be invalid as a result of the
20 validity verification; verifying, by the cloud pocket, whether a header value included in the request statement satisfies a preset specification, when the request statement is determined to be valid as a result of the validity verification; and generating original data by decrypting, by the cloud pocket, an encrypted block included in the request

statement using the user account encryption/decryption key generated based on the authentication block, when the header value included in the request statement satisfies the preset specification.

According to an exemplary embodiment related to the present disclosure, the
5 method may further include: registering, by the cloud pocket, a result of performing a function corresponding to the request statement in the table of the gate server after accessing the gate server at the preset interval; and confirming, by the user terminal, the result of performing the function registered in the table of the gate server.

According to an exemplary embodiment related to the present disclosure, the
10 method may further include: connecting, by the cloud pocket, to a specific main server among a plurality of main servers, and storing a result of performing a function corresponding to the request statement in a specific table within a database of the specific main server; transmitting, by the specific main server, a specific vault ID corresponding to the specific table to the cloud pocket; registering, by the specific
15 main server, the result of performing the function in respective databases of remaining main servers among the plurality of main servers, excluding the specific main server, by interworking with each of the remaining main servers; receiving, by the cloud pocket, the specific vault ID transmitted from the specific main server; registering, by the cloud pocket, the specific vault ID in the table of the gate server after accessing the
20 gate server at the preset interval; and confirming, by the user terminal, the specific vault ID registered in the table of the gate server.

The present disclosure generates an encrypted block by encrypting original data at a user terminal using an authentication key, registers request information

including the encrypted block and a timestamp on a gate server, and after verifying the request information by accessing the gate server from a cloud pocket, restores the original data by decrypting the encrypted block included in the request information. By either storing the corresponding original data as a new block in a blockchain or
5 modifying existing data with the corresponding original data, the blockchain is configured in a cloud pocket connected to a network rather than on a personal PC or device. Consequently, no information is stored on the personal device, and only an encrypted block protocol connecting the user and the cloud pocket exists. In this state, numerous cloud pockets are interconnected via a P2P network to enable currency
10 functions through real-time processing, providing effects such as allowing real-time modification, transmission, and generation, enabling monetary volume control, and tracking the flow of virtual currency.

Furthermore, the present disclosure provides an effect of enabling a user terminal to efficiently perform functions such as block generation and modification by
15 using an authentication key with a preset expiration period. This is achieved through a process where: the user terminal logs into a web server using an ID and a password; the user terminal receives an authentication block provided by the web server in response to a request and registers the received authentication block on a gate server; the cloud pocket verifies authentication by checking the consistency between the
20 authentication block registered by the user terminal and the authentication block registered by the web server on the gate server; upon successful authentication, the cloud pocket generates an authentication key with a set expiration period and registers it on the gate server; and the user terminal, utilizing the authentication key registered

on the gate server, interworks with the gate server and the cloud pocket to register original data in the form of a block into a cloud blockchain.

BRIEF DESCRIPTION OF THE DRAWINGS

5 The above and other objects, features and advantages of the present invention will become more apparent to those of ordinary skill in the art by describing exemplary embodiments thereof in detail with reference to the accompanying drawings, in which:

FIG. 1 is a block diagram illustrating a configuration of a cloud blockchain system according to an embodiment of the present disclosure;

10 FIG. 2 is a diagram illustrating an example of a structure of a user account encryption/decryption key according to an embodiment of the present disclosure;

FIG. 3 is a diagram illustrating an example of a relationship between a Vault ID and a table in a database of a main server according to an embodiment of the present disclosure; and

15 FIGS. 4 to 6 are flowcharts illustrating a method for controlling a cloud blockchain system according to an embodiment of the present disclosure.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

It should be noted that the technical terms used in the present disclosure are
20 employed solely to describe specific embodiments and are not intended to limit the scope of the present disclosure. In addition, unless specifically defined otherwise, technical terms used herein should be interpreted with the meaning generally understood by a person of ordinary skill in the art to which the present disclosure

belongs, and should not be interpreted in an excessively broad or overly restrictive sense.

Furthermore, if a technical term used in the present disclosure is an incorrect term that fails to accurately express the technical spirit of the disclosure, it should be replaced and understood with a technical term that can be correctly understood by those skilled in the art. Additionally, general terms used in the present disclosure should be interpreted according to their dictionary definitions or based on the context of the description, and should not be interpreted in an excessively restrictive sense.

Singular expressions used in the present disclosure include plural expressions unless the context clearly indicates otherwise. The terms such as "consists of" or "comprises" in the present disclosure should not be interpreted as necessarily including all of the various components or steps described in the disclosure. It should be understood that some of the components or steps may not be included, or additional components or steps may be further included.

Furthermore, terms including ordinal numbers such as "first," "second," and the like may be used to describe various components in the present disclosure, but the components should not be limited by these terms. These terms are used only for the purpose of distinguishing one component from another. For example, without departing from the scope of the present disclosure, a first component could be referred to as a second component, and similarly, a second component could also be referred to as a first component.

Hereinafter, preferred embodiments of the present disclosure will be described in detail with reference to the accompanying drawings. The same or similar

components are assigned the same reference numerals regardless of the figure numbers, and redundant descriptions thereof will be omitted.

In describing the present disclosure, detailed descriptions of related well-known technologies will be omitted if it is determined that such descriptions may unnecessarily obscure the gist of the present disclosure. Furthermore, it should be noted that the accompanying drawings are provided only for the purpose of facilitating an easy understanding of the technical spirit of the present disclosure, and the spirit of the present disclosure should not be construed as being limited by the accompanying drawings.

FIG. 1 is a block diagram illustrating a configuration of a cloud blockchain system 10 according to an embodiment of the present disclosure.

As shown in FIG. 1, the cloud blockchain system 10 includes a user terminal 100, a plurality of gate servers 200, a plurality of cloud pockets 300, and a plurality of main servers 400. Not all components of the cloud blockchain system 10 shown in FIG. 1 are essential components; the cloud blockchain system 10 may be implemented with more components than those shown in FIG. 1, or may be implemented with fewer components than those shown in FIG. 1.

The user terminal 100 may be implemented in or applied to various types of terminals, such as a smartphone, a portable terminal, a mobile terminal, a foldable terminal, a personal digital assistant (PDA), a portable multimedia player (PMP) terminal, a telematics terminal, a navigation terminal, a personal computer (PC), a laptop computer, a slate PC, a tablet PC, an ultrabook, a wearable device (including, for example, a smartwatch, smart glasses, a head-mounted display (HMD), etc.), a

Wibro terminal, an internet protocol television (IPTV) terminal, a smart TV, a digital broadcasting terminal, an audio-video navigation (AVN) terminal, an audio/video (A/V) system, a flexible terminal, a digital signage device, an artificial intelligence (AI) speaker, and a spatial computer (including, for example, Vision Pro and the like).

5 The gate server 200, the cloud pocket 300, and the main server 400 may be implemented in the form of a web server, a database server, a proxy server, or the like. In addition, one or more of a network load balancing mechanism or various software for allowing the gate server 200, the cloud pocket 300, and the main server 400 to operate on the Internet or other networks may be installed therein, and thus, they may
10 be implemented as a computerized system. Further, the network may be an HTTP network, and may include a private line, an intranet, or any other arbitrary network. Furthermore, connections among the user terminal 100, the gate server 200, the cloud pocket 300, and the main server 400 may be established via a secure network to prevent data from being attacked by an arbitrary hacker or other third parties. Additionally,
15 the gate server 200, the cloud pocket 300, and the main server 400 may include a plurality of database servers, and may be implemented in a manner where such database servers are separately connected to the gate server 200, the cloud pocket 300, and the main server 400 through any type of network connection, including a distributed database server architecture.

20 Each of the user terminal 100, the gate server 200, the cloud pocket 300, and the main server 400 may include a communication unit (not shown) for performing communication functions with other terminals, a storage unit (not shown) for storing various information and programs (or applications), a display unit (not shown) for

displaying various information and program execution results, an audio output unit (not shown) for outputting audio information corresponding to the various information and the program execution results, and a control unit (not shown) for controlling various components and functions of each terminal.

5 The user terminal 100 communicates with a web server (not shown), the plurality of gate servers 200, and the like. In this case, the user terminal 100 may be a terminal possessed by a user who performs a block generation function, a block modification function, a block inquiry function, a subscription function, and the like through a dedicated application and/or a website provided by the web server.

10 Furthermore, the user terminal 100, interworking with the web server, signs up as a member to be provided with the block generation function, the block modification function, the block inquiry function, the subscription function, and the like through the dedicated application and/or the website provided by the web server, and registers personal information and the like on the web server. In this case, the
15 personal information includes an identifier (ID), an email address, a password, a name, a gender, a date of birth, contact information, an address (or address information), and the like.

 Furthermore, the user terminal 100 registers a personal information usage consent form or the like, which is preset in the form of an electronic document, on the
20 web server by interworking with the web server, in order to use functions (or services) provided by the web server.

 In addition, the user terminal 100 may sign up as a user on the web server by using SNS account information or other site account information of a user of the user

terminal 100. Here, the SNS account information may include information related to Facebook, Twitter, Instagram, Kakao Story, Naver Blog, and the like. Further, the other site account information may include information related to YouTube, Kakao, Naver, and the like. Additionally, mobile messenger account information may
5 include information related to KakaoTalk, LINE, Viber, WeChat, WhatsApp, Telegram, Snapchat, and the like.

Furthermore, during a member registration procedure, the user terminal 100 may complete the member registration procedure for the web server only after an authentication function through an identity verification means (including, for example,
10 a mobile phone, a credit card, an i-PIN, a digital certificate, and the like) is completed. Here, the digital certificate includes a joint certificate, a financial certificate, a private certificate, and the like.

In addition, after the member registration is completed, the user terminal 100 installs a dedicated app (or an application, a program, or a specific app) provided from
15 the web server in the user terminal 100 in order to use a service provided from the web server. In this case, the dedicated app includes a native app, a mobile web app, a responsive web app (RWD), an adaptive web design (AWD) app, a hybrid app, and the like, and may be an app for performing a block generation function, a block modification function, a block inquiry function, a subscription function, and the like.

20 Furthermore, after membership registration is completed, the user terminal 100 may display a discount coupon provided by the web server through the dedicated app. In this case, the discount coupon may be a coupon including discount information of a certain ratio for using a file block generation function, a block

modification function, a block inquiry function, a subscription function, and the like, provided by the web server.

In addition, the user terminal 100 performs a payment function according to the subscription function by interworking with the web server and a payment server
5 (not shown) to perform the functions provided by the web server. In this case, the web server may perform the payment function through a credit card payment, an automatic transfer through interworking with a bank payment account, a payment using cash-equivalent points or cash remaining in an account of the user terminal 100 registered in the web server, or a simple payment including KakaoPay, NaverPay, and
10 the like.

When the payment fails, the user terminal 100 receives information indicating that the payment has failed (including, for example, an insufficient balance, a limit exceeded, and the like) transmitted from the web server (or the payment server), and outputs (or displays) the received information indicating that the payment has failed.

15 In addition, after the payment function is normally performed, the user terminal 100 receives a result of performing the payment function transmitted from the web server, and outputs (or displays) the received payment function result. Here, the result of performing the payment function includes a subscription period, a payment amount, payment date and time information, and the like.

20 Furthermore, the user terminal 100 logs into (or accesses/connects to) the web server (not shown) using an identifier (ID) (or a device ID of the user terminal 100) and a password. In this case, the user terminal 100 may be in a state of having registered as a member to use a service provided by the web server.

Additionally, the user terminal 100 requests the web server to generate an authentication block according to a request of a user of the user terminal 100.

Furthermore, the web server generates an authentication block comprising an identifier (ID) (or a device ID of the user terminal 100) and a password associated with the user terminal 100, in response to an authentication block generation request (or authentication block generation request information) of the user terminal 100, and provides (or transmits) the generated authentication block to the user terminal 100.

In addition, the web server registers (or stores) the generated authentication block in a gate server 200 (or any one of a plurality of tables within the gate server 200) interworking with the web server. In this case, the gate server 200 registers (or stores) the authentication block provided from the web server into any one table among a plurality of tables (or a plurality of vaults) within the gate server 200.

Additionally, the user terminal 100 receives the authentication block provided (or transmitted) from the web server in response to the previous authentication block generation request (or authentication block generation request information). Here, after receiving the authentication block, the user terminal 100 may terminate a login state (or a connection state/link state) with the web server (or log out).

The user terminal 100 accesses (or connects to) a specific gate server 200 among the plurality of gate servers 200 to be issued an authentication key (or an authentication key block) from the cloud pocket 300. In this case, the user terminal 100 and the gate server 200 communicate using a preset HTTP protocol (or HTTPS protocol). Here, data (or information) transmitted and received between the user

terminal 100 and the gate server 200 may be transmitted and received by combining internal encryption with the HTTPS protocol.

For example, the user terminal 100 generates an encrypted block (e.g., a block of 512 bytes, 1024 bytes, etc.) by encrypting data. When the generated encrypted block
5 (or an encrypted data block) is included as byte data in the HTTP protocol (or set as bytes in an argument of HTTPS), the HTTP protocol is automatically encrypted.

Furthermore, the user terminal 100 registers (or stores) the received authentication block in the connected specific gate server 200.

That is, the user terminal 100 registers (or stores) the authentication block into
10 a specific table (or a specific vault) among a plurality of tables (or a plurality of vaults) within the specific gate server 200.

In addition, the user terminal 100 generates an encrypted block (or encrypted data) by encrypting original data to be registered in the cloud blockchain using the user account encryption/decryption key included in the authentication key block, in order
15 to store (or generate/modify) data in the cloud blockchain managed by the cloud pocket 300 or to query data registered in the cloud blockchain. Here, the original data (or a target block/target data) includes not only content (or information/data) (including, for example, text, numbers, binary data, and the like) but also requirements (including, for example, generation request information, modification request information, inquiry
20 request information, and the like). Additionally, the encrypted block may include a header value (or a header value related to encryption) as well as the original data. If the size of the original data is large, it may be divided into preset units (including, for

example, 512 bytes, 1024 bytes, and the like) to configure a plurality of encrypted blocks.

In this case, when the user terminal 100 generates the encrypted block, the user terminal 100 checks a timestamp within the authentication key block. If a
5 validity period (or validity time) of the timestamp has expired based on a current time, the user terminal 100 displays information indicating that the validity period of the authentication key (or the authentication key block) has expired, interrupts (or stops) a process of generating the encrypted block, and may display (or output) information indicating a state in which the authentication block must be re-issued through
10 communication with the web server (or re-issuance guide information for the authentication block).

Furthermore, the user terminal 100 generates a request statement (or a command statement, a request block, or a request statement block) comprising the generated encrypted block and the authentication key included in the authentication
15 key block.

In addition, the user terminal 100 accesses (or communicates with/connects to) one specific gate server 200 among the plurality of gate servers 200. In this case, the user terminal 100 may be in a state of maintaining the previous connection with the specific gate server 200.

20 Additionally, the user terminal 100 registers (or stores) the generated request statement in the connected specific gate server 200.

That is, the user terminal 100 registers (or stores) the request statement into another specific table (or another specific vault) among the plurality of tables within

the specific gate server 200. In this case, the user terminal 100 may cumulatively store the request statement in the specific table where data (including, for example, the authentication block and the like) was previously registered, or may initialize the specific table and then store the request statement therein.

5 As such, the user terminal 100 may store (or register) data to be processed (including, for example, inquiry, generation, modification, and the like) through the cloud blockchain in the specific gate server 200 interworking with the user terminal 100.

10 Furthermore, the user terminal 100 confirms (or displays/outputs) authentication failure information registered (or stored) in the table (or vault) of the specific gate server 200. In this case, while connected to the specific gate server 200, the user terminal 100 may receive notification information (or guide information) transmitted from the specific gate server 200 as new information (including, for example, the authentication failure information and the like) is newly registered in the
15 table, and confirm the authentication failure information based on the received notification information. Additionally, the user terminal 100 may further display guide information indicating that a new authentication block should be re-issued through the aforementioned steps.

20 The user terminal 100 confirms (or displays, outputs, or downloads) the authentication key block registered (or stored) in the table (or the vault) of the specific gate server 200. In this case, while connected to the specific gate server 200, the user terminal 100 may receive notification information (or guide information) transmitted from the specific gate server 200 as new information (including, for example, the

authentication key block and the like) is newly registered in the table of the specific gate server 200, and confirm the authentication key block registered in the table based on the received notification information.

Furthermore, the user terminal 100 confirms (or displays/outputs) information
5 indicating a state where validity verification of the data registered (or stored) in the table (or the vault) of the specific gate server 200 has failed. In this case, while connected to the specific gate server 200, the user terminal 100 may receive other notification information (or other guide information) transmitted from the specific gate server 200 as new information (including, for example, information indicating that the
10 validity verification of the data has failed) is newly registered in the table of the specific gate server 200, and confirm the information indicating that the validity verification of the data has failed based on the received other guide information. Additionally, the user terminal 100 may further display guide information indicating that a new authentication key must be re-issued through the aforementioned steps.

15 Furthermore, the user terminal 100 confirms (or displays/outputs) information indicating a state where specification verification of the data registered (or stored) in the table (or the vault) of the specific gate server 200 has failed. In this case, while connected to the specific gate server 200, the user terminal 100 may receive other notification information (or other guide information) transmitted from the specific gate
20 server 200 as new information (including, for example, information indicating that the specification verification of the data has failed) is newly registered in the table of the specific gate server 200, and confirm the information indicating that the specification verification of the data has failed based on the received other guide information.

Additionally, the user terminal 100 may further display guide information indicating that a new authentication key should be re-issued through the aforementioned steps.

Furthermore, the user terminal 100 confirms (or displays/outputs) a result of performing a function registered (or stored) in the table (or the vault) of the specific gate server 200. In this case, while connected to the specific gate server 200, the user terminal 100 may receive notification information (or guide information) transmitted from the specific gate server 200 as new information (including, for example, the result of performing the function) is newly registered in the table of the specific gate server 200, and confirm the result of performing the function registered in the table based on the received notification information.

The user terminal 100 confirms (or displays/outputs) a specific vault ID registered (or stored) in the table (or the vault) of the specific gate server 200.

Furthermore, the user terminal 100 uses the specific vault ID to perform a modification function, an inquiry function, or the like, regarding information (or data) related to a corresponding case in the future.

The gate server 200 communicates with the web server, the user terminal 100, the cloud pocket 300, and the like. Here, the gate server 200 may temporarily store data (or information) provided from the user terminal 100, provide the temporarily stored data in response to a request from the cloud pocket 300, temporarily store a result of performing a function provided from the cloud pocket 300 in response to the data, and provide the temporarily stored result of performing the function to the user terminal 100. The gate server 200 may be a server that performs a role of a mediator (or a relay function).

In addition, the gate server 200 is provided in plurality, and the plurality of gate servers 200 share and store data respectively stored in each gate server 200 with each other. In this case, the plurality of gate servers 200 may share (or synchronize data) regarding data stored (or registered) in each of the gate servers 200 with each other.

In addition, a plurality of cloud pockets 300 may access one gate server 200. Here, the plurality of gate servers 200 have different addresses (including, for example, IP addresses, URL addresses, and the like).

For example, a first cloud pocket 300 to a 100th cloud pocket 300 may access a first gate server 200, and a 101st cloud pocket 300 to a 200th cloud pocket 300 may access a second gate server 200.

Additionally, a single cloud pocket 300 may also access a plurality of gate servers 200.

As such, connections between the plurality of gate servers 200 and the plurality of cloud pockets 300 may be managed by a preset policy (including, for example, a policy where the 1st cloud pocket to the 100th cloud pocket access the first gate server 200, and the like).

Furthermore, the gate server 200 interworks with the web server to register (or store) the authentication block provided by the web server into the gate server 200 (or any table among a plurality of tables within the gate server 200).

In addition, the gate server 200 is configured to interwork with each of remaining gate servers 200, excluding the gate server 200 itself, among the total

plurality of gate servers 200, to respectively store (or register/update) the authentication block in the remaining gate servers 200.

Furthermore, the gate server 200 connects to the user terminal 100 for issuing an authentication key (or an authentication key block) from the cloud pocket 300. In
5 this case, the user terminal 100 and the gate server 200 communicate using a preset HTTP protocol (or HTTPS protocol).

Moreover, the gate server 200 interworks with the user terminal 100 to register (or store) the authentication block in any one specific table (or a specific vault) among a plurality of tables (or vaults) within the specific gate server 200. In this case, the
10 specific gate server 200 may allocate at least one table related to the user terminal 100 among the plurality of tables managed by the specific gate server 200, and register the authentication block in any one of the at least one allocated table.

In addition, the gate server 200 connects to at least one cloud pocket 300 among the plurality of cloud pockets 300. In this case, a connection between the gate
15 server 200 and the cloud pocket 300 may be a unidirectional communication in which only the cloud pocket 300 accesses the specific gate server 200.

Furthermore, when data registered by the user terminal 100 exists in at least one table among the plurality of tables within the gate server 200, the gate server 200 transmits (or provides) the data to the cloud pocket 300 in response to a request from
20 the cloud pocket 300.

Furthermore, the gate server 200 interworks with the cloud pocket 300 to register (or store) the authentication key block generated by the cloud pocket 300 into the table (or the vault) of the specific gate server 200. Here, the gate server 200 stores

the authentication key block in addition to data previously stored (including, for example, the authentication block and the like) in the table, or deletes the data stored in the table and then stores the authentication key block. In this case, the gate server 200 may respectively store the authentication key block, the authentication block, and
5 the like in different tables among the plurality of tables of the specific gate server 200.

In addition, the gate server 200 interworks with the user terminal 100 to register (or store) the request statement generated by the user terminal 100 into another specific table (or another specific vault) among the plurality of tables within the gate server 200.

10 Additionally, the gate server 200 interworks with the cloud pocket 300 to register (or store) a result of performing a function (including, for example, a result of performing a block generation function, a block modification function, a block inquiry function, and the like) generated by the cloud pocket 300 into the table of the specific gate server 200. Here, the gate server 200 stores the result of performing the function
15 in addition to data previously stored (including, for example, the authentication block, the request statement, and the like) in the table, or deletes the data stored in the table and then stores the result of performing the function. In this case, the gate server 200 may separately store the authentication key block, the authentication block, the request statement, the result of performing the function, and the like in different tables among
20 the plurality of tables of the specific gate server 200. Here, even when the data such as the authentication key block, the authentication block, the request statement, and the result of performing the function are stored separately, the gate server 200 may

confirm which user terminal 100 the respective data is associated with by using a header (or a header value) included in each piece of data.

The cloud pocket (CP) 300 (also referred to as a cloud server, a cloud pocket server, or a blockchain cloud pocket) communicates with the plurality of gate servers 200, the plurality of main servers 400, and the like. Here, at least one gate server 200 among the plurality of gate servers 200 is allocated to the cloud pocket 300 according to a preset policy (or condition), and the cloud pocket 300 may connect to the allocated at least one gate server 200. In this case, the cloud pocket 300 accesses the gate server 200 through unidirectional communication and accesses the gate server 200 at a preset interval. Furthermore, the plurality of cloud pockets 300 may share (or synchronize) data stored (or registered) in each of the cloud pockets 300 with each other.

Additionally, the cloud pocket 300 provides a cloud service utilizing blockchain technology, which covers from infrastructure (or hardware) configuring a blockchain network to application development and operation.

15 A server in an Internet Data Center (IDC) may serve as the cloud pocket 300.

For example, in the case of a Linux server, a plurality of independently running threads may be launched. In this case, one thread becomes the cloud pocket 300.

As another method, if a hub is connected to an IDC and a plurality of devices are attached to the hub, each individual device may serve as the cloud pocket 300.

20 In this manner, both an independent process CP on a server side and a device CP are possible.

All of these CPs are connected via a blockchain.

For example, if a hundred processes exist in one server and thirty smartphones are connected with five smartphones per hub, there are a total of two hundreds and fifty cloud pockets 300, including the hundred processes and one hundred and fifty smartphones.

5 In addition, all of these cloud pockets 300 identically possess the same data.

Furthermore, the cloud pocket 300 manages (or configures/operates) a cloud blockchain. Here, the cloud blockchain is a system combining cloud computing and blockchain technology, and may be a method of providing data storage, processing, security, and management related to a blockchain network in a cloud environment.

10 As described above, the cloud blockchain system 10 according to the present invention is a technology in which a blockchain is configured in a cloud pocket connected to a network, rather than being configured in a personal PC or device.

Accordingly, because coins are connected via the blockchain to the cloud pocket 300 on the network connected to a coin holder (or a user of the user terminal
15 100), no information is stored in the personal device. However, only an encrypted block protocol for establishing a connection between the user and the cloud pocket 300 exists in the user terminal 100.

In addition, numerous cloud pockets 300 are connected to each other via P2P on a network, and because multiple wallets are stored in one cloud pocket 300, a P2P
20 speed is very high, and thus real-time transmission may be processed.

Furthermore, the cloud pocket 300 manages (or stores) a pocket server list, which is information related to other cloud pockets 300. Here, the pocket server list includes information on which cloud pocket 300 a wallet (or a cryptocurrency/virtual

currency wallet) corresponding to a user account is stored (or managed) in, and includes cloud pocket information (or an address/URL of the cloud pocket 300) for each piece of wallet information.

In addition, when the cloud pocket 300 is an independent terminal, the cloud
5 pocket 300 may have one address.

Additionally, one process of one large-scale server may serve as the cloud pocket 300. In this case, the pocket server list may be located in the large-scale server.

That is, the cloud pocket 300 may be a terminal (including, for example, a PC and the like) or may be a shell process of a server.

10 Therefore, when the cloud pocket 300 establishes a connection, it accesses via an IP address if the IP address is available, and in the case of being a process of a specific server, it connects using the pocket server list of the specific server.

In addition, the cloud pocket 300 connects to (or communicates with) the specific gate server 200 at a preset interval, and then checks whether data (including,
15 for example, an authentication block) is in a registered (or stored) state in a plurality of tables (or vaults) within the specific gate server 200. In this case, at least one gate server 200 to be managed by the cloud pocket 300 is in an allocated (or set) state among a plurality of gate servers 200, and the cloud pocket 300 may respectively access the at least one gate server 200 according to the interval to check whether data
20 is registered in the tables. Here, the cloud pocket 300 is one of a plurality of cloud pockets 300 and is preset to communicate with the specific gate server 200, and the connection between the cloud pocket 300 and the specific gate server 200 may be

unidirectional communication in which only the cloud pocket 300 accesses the specific gate server 200.

As a result of the check, if data is in a registered state in at least one table among the plurality of tables, the cloud pocket 300 confirms the data registered in the
5 table.

That is, the cloud pocket 300 regularly accesses the specific gate server 200 at constant time intervals (or cycles) while running a loop (or a thread), and checks whether there is a task (or data) to be processed by the cloud pocket 300. Here, the task to be processed by the cloud pocket 300 may be a function related to data
10 registered in the table within the specific gate server 200 by an arbitrary user terminal 100 or the like.

Furthermore, as a result of the check, if data is not registered in any of the plurality of tables, the cloud pocket 300 interworks with the specific gate server 200 again in a next cycle (or a next thread) to perform a process of checking whether data
15 is registered in the tables.

In addition, when the data is an authentication block, the cloud pocket 300 interworks with the specific gate server 200 to compare (or verify) whether the authentication block registered by the web server in the specific gate server 200 matches the authentication block registered by the user terminal 100 in the specific
20 gate server 200. Here, the cloud pocket 300 may perform parsing (or syntax analysis) on the data, confirm a type (or content) of the data based on a result of the parsing, and confirm whether the data is an authentication block including an ID and a password based on the confirmed type of the data. In this case, the cloud pocket 300 may also

confirm whether the data is an authentication block based on a header of the data (or additional information included in the data).

As a result of the comparison (or the check), when the authentication block registered (or stored) in the specific gate server 200 by the web server does not match
5 the authentication block registered (or stored) in the specific gate server 200 by the user terminal 100, the cloud pocket 300 registers (or stores) authentication failure information in the table (or the vault) of the specific gate server 200, and terminates the connection with the specific gate server 200. In this case, the cloud pocket 300 stores the authentication failure information in addition to data previously stored
10 (including, for example, the authentication block and the like) in the table, or deletes the data stored in the table and then stores the authentication failure information. Here, the cloud pocket 300 may respectively store the authentication failure information and the authentication block in different tables among the plurality of tables of the specific gate server 200.

15 Furthermore, as a result of the comparison (or the check), when the authentication block registered by the web server matches the authentication block registered by the user terminal 100, the cloud pocket 300 confirms the identifier (ID) (or device ID), the password, and the like included in the authentication block.

In addition, the cloud pocket 300 generates (or issues) an authentication key
20 based on the confirmed ID, password, current time, and a preset validity period. In this case, the authentication key includes the ID (or device ID), the password, and a timestamp including validity period information for the authentication key. Here, the

validity period may be a preset value (including, for example, 3 minutes, 5 minutes, 10 minutes, and the like).

Additionally, the cloud pocket 300 generates a user account encryption/decryption key. Here, the user account encryption/decryption key may be
5 used in a preset state or may be generated and used upon request, and comprises (or includes) an inquiry account key (or a decryption key), a modification account key (or an encryption key), and a computation key. The user account encryption/decryption key is not an account key used for access (login), but a key used for decrypting an encrypted block or encrypting general data. Furthermore, the cloud pocket 300 uses
10 a computational encryption method as an encryption method for the cloud blockchain. Accordingly, the computation key is additionally added along with the two keys (inquiry and modification keys). The encryption method using computation (or a computational encoding scheme) is a method already widely used in computers and thus will not be described in detail herein.

15 That is, the cloud pocket 300 generates the user account encryption/decryption key, which is a byte key of a preset size (e.g., 256 bytes times three), and the byte key indicates that the key unit is in bytes.

As shown in FIG. 2, the cloud pocket 300 generates the user account encryption/decryption key consisting of an inquiry account key of a preset size (e.g.,
20 256 bytes) for interpreting (or decrypting) encrypted data, a modification account key of a preset size (e.g., 256 bytes) for encrypting data, and a computation key of a preset size (e.g., 256 bytes) for making interpretation via patterns impossible by using the computational encoding scheme.

In an embodiment of the present invention, it is mainly described that the cloud pocket 300 generates the user account encryption/decryption key when a user account is created (or when the authentication key is generated, or when the user terminal 100 registers the authentication block in the gate server 200). However, the present invention is not limited thereto, and the cloud pocket 300 may use a pre-stored user account encryption/decryption key, which is commonly used by the plurality of cloud pockets 300, to generate the authentication key block.

Furthermore, the cloud pocket 300 registers (or stores) an authentication key block including the generated authentication key and the generated user account encryption/decryption key into the table (or the vault) of the specific gate server 200. In this case, the cloud pocket 300 may store the authentication key block in addition to data previously stored (including, for example, the authentication block and the like) in the table, or may delete the data stored in the table and then store the authentication key block. In this case, the cloud pocket 300 may respectively store the authentication key block, the authentication block, and the like in different tables among a plurality of tables of the specific gate server 200.

In addition, the cloud pocket 300 interworks with one main server 400 among the plurality of main servers 400 to store (or register) the authentication key block in one record within one table among a plurality of tables in a database (not shown) of the main server 400. In this case, as the authentication key block is stored in a specific record within a specific table of the database, the cloud pocket 300 receives a vault ID (or a vault ID corresponding to the specific table) provided from the main

server 400. Here, the cloud pocket 300 may also store the vault ID when storing the authentication key block in the table within the specific gate server 200.

As described above, the cloud pocket 300 and the main server 400 are connected, such that the cloud pocket 300 may possess its own data block and may
5 perform overall flow control and management by sending the data to the main server 400.

In addition, after connecting to (or communicating with) the specific gate server 200 at the preset interval, the cloud pocket 300 checks whether data (including, for example, a request statement) is in a registered (or stored) state in a plurality of
10 tables (or vaults) within the specific gate server 200.

As a result of the check, if data is in a registered state in at least one table among the plurality of tables, the cloud pocket 300 confirms the data registered in the table (or vault).

That is, the cloud pocket 300 regularly accesses the specific gate server 200 at
15 constant time intervals (or periods) while running a loop (or a thread), and checks whether there is a task (or data) to be processed by the cloud pocket 300. Here, the task to be processed by the cloud pocket 300 may be a function related to data registered in the table within the specific gate server 200 by an arbitrary user terminal 100 or the like.

20 Furthermore, as a result of the check, if data is not registered in any of the plurality of tables, the cloud pocket 300 interworks with the specific gate server 200 again in a next cycle (or a next thread) to perform a process of checking whether data is registered in the tables.

In addition, when the data is a request statement, the cloud pocket 300 primarily performs validity verification (or determination) on the data (or the request statement) based on a timestamp included in the request statement. Here, the cloud pocket 300 may perform parsing (or syntax analysis) on the data, confirm a type (or content) of the data based on a result of the parsing, and confirm whether the data is a request statement including an encrypted block, an authentication key, and the like based on the confirmed type of the data. In this case, the cloud pocket 300 may also confirm whether the data is a request statement based on a header of the data (or additional information included in the data).

That is, when the data is a request statement, the cloud pocket 300 verifies (or determines) whether a time difference between a current time and a timestamp included in the request statement has passed (or exceeded) a preset reference value (including, for example, 3 seconds, 5 seconds, 10 seconds, and the like) based on the current time and the timestamp included in the request statement, in order to verify validity of the data (or the request statement).

As a result of the validity verification of the data, when the request statement is determined to be invalid, the cloud pocket 300 registers (or stores) information indicating that the validity verification of the data has failed in the table (or the vault) of the specific gate server 200, and terminates the connection with the specific gate server 200. Here, the cloud pocket 300 stores information indicating that the validity verification of the data has failed in addition to data previously stored (including, for example, the authentication block, the request statement, and the like) in the table of the specific gate server 200, or deletes the data stored in the table and then stores the

information indicating that the validity verification of the data has failed. Here, the cloud pocket 300 may respectively store the information indicating that the validity verification of the data has failed, the authentication block, the request statement, and the like in different tables among a plurality of tables of the specific gate server 200.

5 That is, if the time difference between the current time and the timestamp included in the request statement has passed (or exceeded) the preset reference value (including, for example, 3 seconds, 5 seconds, 10 seconds, and the like), the cloud pocket 300 registers (or stores) information indicating that the validity verification of the data has failed in the table (or the vault) of the specific gate server 200, and
10 terminates the connection with the specific gate server 200.

 In addition, as a result of the validity verification of the data (or a result of determining the validity of the data/request statement), if the request statement is determined to be valid, the cloud pocket 300 secondarily checks (or determines) whether a header value included in the request statement matches (or satisfies) a preset
15 specification (or condition). Here, the header value may be a value generated in a process of encrypting the encrypted block included in the request statement.

 Specifically, if the time difference between the current time and the timestamp included in the request statement has not passed (or has not exceeded) the preset reference value (including, for example, 3 seconds, 5 seconds, 10 seconds, and the
20 like), the cloud pocket 300 determines that the request statement is valid, and secondarily checks (or determines) whether the header value included in the request statement matches (or satisfies) the preset specification (or condition).

As a result of the check (or the determination), if the header value included in the request statement does not match (or satisfy) the preset specification (or condition), the cloud pocket 300 registers (or stores) information indicating that the specification verification of the data has failed in the table (or the vault) of the specific gate server 200, and terminates the connection with the specific gate server 200. Here, the cloud pocket 300 stores the information indicating that the specification verification of the data has failed in addition to data previously stored (including, for example, an authentication block, a request statement, and the like) in the table, or deletes the data stored in the table and then stores the information indicating that the specification verification of the data has failed. In this case, the cloud pocket 300 may respectively store the information indicating that the specification verification of the data has failed, the authentication block, the request statement, and the like in different tables among a plurality of tables of the specific gate server 200.

Furthermore, as a result of the check (or the determination), if the header value included in the request statement matches (or satisfies) the preset specification (or condition), the cloud pocket 300 tertiarily decrypts the encrypted block included in the request statement using a preset user account encryption/decryption key (or the user account encryption/decryption key generated based on the authentication block associated with the user terminal 100 previously) to generate (or restore) original data.

That is, as a result of the secondary specification verification of the data, when the header value included in the request statement matches (or satisfies) the preset specification (or condition), the cloud pocket 300 separates the request statement into an authentication key and an encrypted block, and decrypts the separated encrypted

block using the user account encryption/decryption key to generate (or restore) original data.

In addition, the cloud pocket 300 performs a block generation function, a block modification function, a block inquiry function (or a block content inquiry
5 function), and the like, in the cloud blockchain based on the original data and the like.

That is, the cloud pocket 300 performs parsing (or syntax analysis) on the original data and identifies a meaning (or content) of the original data according to the parsing (or a result of the parsing).

Furthermore, when a requirement included in the request statement is a new
10 block generation request (or when the identified meaning/content is a new block generation request), the cloud pocket 300 checks whether the block is to be generated in the cloud pocket 300 among a pocket server list managed by the cloud pocket 300.

That is, the cloud pocket 300 checks, through the pocket server list, whether the new block generation request is a task (or function) to be performed in the cloud
15 pocket 300, which is a local server. Here, the pocket server list includes information on which cloud pocket 300 a wallet (or a cryptocurrency/virtual currency wallet) corresponding to a user account is stored (or managed) in, and includes cloud pocket information (or an address/URL of the cloud pocket 300) for each piece of wallet information.

20 As a result of the checking, if the block generation is to be performed in the cloud pocket 300, the cloud pocket 300 performs the block generation function in the cloud blockchain based on the original data.

That is, as a result of the checking, when it is confirmed that the new block generation request is a task (or function) to be performed in the cloud pocket 300, which is the local server that has checked the request statement, the cloud pocket 300 performs the block generation function based on the original data in the cloud pocket 300. In this case, the cloud pocket 300 may also perform a block generation function of creating a new wallet for a user associated with the request statement and managing information related to the newly created wallet.

In addition, the cloud pocket 300 updates a result of performing the block generation function related to the new block generation (or updates according to the new block generation in the cloud blockchain).

In addition, the cloud pocket 300 updates results of performing the block generation function related to the new block generation in each of remaining cloud pockets 300, excluding the cloud pocket 300 itself among all the cloud pockets 300, by using threads.

Furthermore, as a result of the checking, in the case of block generation in another cloud pocket 300 (or in the case where the block generation is not in the local server that checked the request statement), the cloud pocket 300 identifies another cloud pocket 300 in charge of performing the block generation function based on the pocket server list managed by the cloud pocket 300, and provides the request statement (or the original data) to the identified another cloud pocket 300.

Accordingly, the other cloud pocket 300 performs the block generation function in the cloud blockchain based on the request statement (or the original data). At this time, the other cloud pocket 300 may perform a block generation function of

newly creating a user wallet associated with the request statement and managing information related to the newly created wallet.

In addition, after completing the performance of the block generation function, the cloud pocket 300 updates a result of performing the block generation function related to the new block generation (or a result of performing the block generation function related to the new block generation in the other cloud pocket 300) (or an update according to the new block generation in the cloud blockchain). In this case, the cloud pocket 300 may update result information on the user wallet newly created according to the request statement.

Furthermore, the cloud pocket 300 updates the results of performing the block generation function related to the new block generation in each of the remaining cloud pockets 300, excluding the cloud pocket 300 itself among all the cloud pockets 300, by using threads.

At this time, the other cloud pocket 300 updates the result of performing the block generation function related to the new block generation in the other cloud pocket 300, and may update the result of performing the block generation function related to the new block generation in each of the remaining cloud pockets 300, excluding the other cloud pocket 300 itself among all the cloud pockets 300, by using threads.

Additionally, when a requirement included in the request statement is a block modification request (or when the identified meaning is a block modification request), the cloud pocket 300 checks whether a primary modification cloud pocket 300 is local based on the pocket server list managed by the cloud pocket 300.

That is, regarding the block modification request, the cloud pocket 300 checks, through the pocket server list, whether the primary modification cloud pocket 300 is a task (or function) to be performed in the cloud pocket 300, which is the local server.

As a result of the checking, when the primary modification is to be performed
5 in the cloud pocket 300, the cloud pocket 300 performs a primary block modification function in the cloud blockchain based on the original data. Here, the primary block modification function indicates a block modification function targeting a target party.

Specifically, when it is confirmed that the primary modification cloud pocket 300 is a task (or function) to be performed in the local server that has checked the
10 request statement, the cloud pocket 300 performs the primary block modification function based on the original data. Here, the cloud pocket 300 may perform a block modification function of deducting an amount of cryptocurrency included in the request statement from a user's wallet associated with the request statement.

Furthermore, as a result of the checking, in the case where the primary
15 modification is to be performed in another cloud pocket 300 (or in the case where the primary block modification is not in the local server that checked the request statement), the cloud pocket 300 identifies another cloud pocket 300 in charge of performing the primary block modification function based on the pocket server list, and provides the request statement (or the original data) to the identified another cloud
20 pocket 300.

Accordingly, the other cloud pocket 300 performs the primary block modification function in the cloud blockchain based on the request statement (or the original data). Here, the other cloud pocket 300 may perform a block modification

function of deducting the amount of cryptocurrency included in the request statement from the user's wallet associated with the request statement.

In addition, the cloud pocket 300 interworks with the other cloud pocket 300 to ascertain (or confirm) whether the performance of the primary block modification
5 function in the other cloud pocket 300 is completed.

Furthermore, after the primary block modification function is completed, the cloud pocket 300 checks whether a secondary modification cloud pocket 300 is local based on the pocket server list.

That is, regarding the block modification request, the cloud pocket 300
10 determines, through the pocket server list, whether the secondary modification cloud pocket 300 is a task (or function) to be performed in the cloud pocket 300, which is the local server.

As a result of the determination, when the secondary modification is to be performed in the cloud pocket 300, the cloud pocket 300 performs a secondary block
15 modification function in the cloud blockchain based on the original data. Here, the secondary block modification function indicates a block modification function targeting a counterparty (or a recipient party).

Specifically, when it is determined that the secondary modification cloud pocket 300 is a task (or function) to be performed in the cloud pocket 300, which is
20 the local server that has checked the request statement, the cloud pocket 300 performs the secondary block modification function in the cloud blockchain based on the original data. In this case, the cloud pocket 300 may perform a block modification

function of processing (e.g., deducting or updating) an amount of cryptocurrency included in the request statement in a wallet associated with the request statement.

Furthermore, as a result of the determination, in the case where the secondary modification is to be performed in another cloud pocket 300 (or in the case where the secondary block modification is not in the local server that checked the request statement), the cloud pocket 300 identifies another cloud pocket 300 in charge of performing the secondary block modification function based on the pocket server list, and provides the request statement (or the original data) to the identified another cloud pocket 300.

Accordingly, the other cloud pocket 300 performs the secondary block modification function in the cloud blockchain based on the request statement (or the original data). Here, the other cloud pocket 300 may perform a block modification function of processing the amount of cryptocurrency included in the request statement in the user's wallet associated with the request statement.

In addition, the cloud pocket 300 interworks with the other cloud pocket 300 to ascertain (or confirm) whether the performance of the secondary block modification function in the other cloud pocket 300 is completed.

Furthermore, after completing the performance of the primary and secondary block modification functions, the cloud pocket 300 updates a result of performing the block modification function related to the block modification (e.g., results from the local cloud pocket 300, the other cloud pocket 300, etc.) (or updates according to the block modification in the cloud blockchain). Here, the cloud pocket 300 may update

result information regarding changes in the wallets of the target party and the counterparty according to the transfer of cryptocurrency based on the request statement.

Additionally, the cloud pocket 300 updates the results of performing the block modification function related to the block modification in each of remaining cloud
5 pockets 300, excluding the cloud pocket 300 itself among all the cloud pockets 300, by using threads.

As described above, when the requirement included in the request statement is a block modification request (or when the identified meaning is a block modification request / in a case of wishing to transmit coins), a blockchain connection order is
10 determined such that a cloud pocket 300 where a sender's block is stored and a cloud pocket 300 where a recipient's block is stored are preferentially modified. Once the modification function in the cloud pocket 300 of the sender and the cloud pocket 300 of the recipient is completed, the system is configured to update the result of performing the block modification function in each of all the remaining cloud pockets
15 300 among the plurality of cloud pockets 300. This applies a method distinguished from a conventional blockchain connection based on unspecified random P2P methods.

Furthermore, when the requirement included in the request statement is a block inquiry request (or when the identified meaning is a block inquiry request), the cloud pocket 300 checks whether the block inquiry is to be performed in the cloud
20 pocket 300 among a pocket server list managed by the cloud pocket 300.

That is, the cloud pocket 300 checks, through the pocket server list, whether the block inquiry request is a task (or function) to be performed in the cloud pocket 300, which is a local server.

As a result of the checking, if the block inquiry is to be performed in the cloud pocket 300, the cloud pocket 300 performs a function of checking information (or information/data within a block) corresponding to the block inquiry request in the cloud blockchain.

5 Specifically, when it is confirmed that the block inquiry request is a task (or function) to be performed in the local server that has checked the request statement, the cloud pocket 300 performs the block inquiry function based on the original data in the cloud blockchain. In this case, the cloud pocket 300 may perform an inquiry function of checking a balance remaining in a user's wallet associated with the request
10 statement.

In addition, the cloud pocket 300 updates a result of performing the block inquiry function (or an update according to the block inquiry in the cloud blockchain).

The cloud pocket 300 updates the results of performing the block inquiry function in each of the remaining cloud pockets 300, excluding the cloud pocket 300
15 itself among all the cloud pockets 300, by using threads.

Moreover, as a result of the checking, in the case of a block inquiry in another cloud pocket 300 (or in the case where the block inquiry is not in the local server that checked the request statement), the cloud pocket 300 identifies another cloud pocket 300 in charge of performing the block inquiry function based on the pocket server list,
20 and provides the request statement (or the original data) to the identified another cloud pocket 300.

Accordingly, another cloud pocket 300 performs a block inquiry function in the cloud blockchain based on the request statement (or the original data). In this

case, the other cloud pocket 300 may perform an inquiry function of checking a balance remaining in a user's wallet associated with the request statement.

Furthermore, after completing the performance of the block inquiry function, the cloud pocket 300 updates a result of performing the block inquiry function (or a
5 result of performing the block inquiry function related to the block inquiry in the other cloud pocket 300) (or updates according to the block inquiry in the cloud blockchain). Here, the cloud pocket 300 may update information regarding the balance remaining in the user's wallet according to the request statement.

In addition, the cloud pocket 300 updates the results of performing the block
10 inquiry function in each of remaining cloud pockets 300, excluding the cloud pocket 300 itself among all the cloud pockets 300, by using threads.

In this case, the other cloud pocket 300 updates the result of performing the block inquiry function in the other cloud pocket 300, and may also update the results of performing the block inquiry function in each of the remaining cloud pockets 300,
15 excluding the other cloud pocket 300 itself among all the cloud pockets 300, by using threads.

As described above, the cloud pocket 300 implements a blockchain (or a cloud blockchain) through a plurality of cloud pockets 300, thereby reducing risks such as hacking.

Moreover, after connecting to (or communicating with/accessing) the specific
20 gate server 200 at a preset interval, the cloud pocket 300 registers (or stores) a result of performing a function (including, for example, a result of performing a block generation function, a block modification function, a block inquiry function, and the

like) corresponding to the request statement into the table (or the vault) of the specific gate server 200. Here, the cloud pocket 300 stores the result of performing the function in the table where data (including, for example, an authentication block, an authentication key block, a request statement, and the like) was previously registered
5 in relation to the user terminal 100, or deletes the data stored in the table and then stores the result of performing the function. In this case, the cloud pocket 300 may respectively store the authentication block, the authentication key block, the request statement, the result of performing the function, and the like in different tables among a plurality of tables of the specific gate server 200.

10 As described above, while conventional blockchain-based virtual currencies are somewhat unsuitable for functioning as a currency, the cloud pocket 300 (or the cloud blockchain system 10) is configured to enable monetary functions while maintaining the advantages of a blockchain.

Furthermore, the cloud blockchain system 10 allows for real-time
15 modification, transmission, and generation, enables control over the money supply, and can track the flow of currency.

The cloud pocket 300 connects to (or accesses/communicates with) one of the plurality of main servers 400, and stores (or registers) a result of performing a function (including, for example, a result of performing a block generation function, a block
20 modification function, a block inquiry function, and the like) into a specific table within a database (not shown) in the connected main server 400.

Additionally, the cloud pocket 300 receives a specific vault ID related to the specific table in which the result of performing the function is stored, provided from the main server 400.

After connecting to (or communicating with/accessing) the specific gate server 200 at a preset interval, the cloud pocket 300 registers (or stores) the specific vault ID into the table (or the vault) of the specific gate server 200. Here, the cloud pocket 300 may cumulatively store the specific vault ID in the table within the specific gate server 200 where data (including, for example, an authentication block, an authentication key block, a request statement, and the like) was previously registered in relation to the user terminal 100, or may initialize the table and then store the specific vault ID therein. In this case, the cloud pocket 300 may respectively store the authentication block, the authentication key block, the request statement, the specific vault ID, and the like in different tables among the plurality of tables of the specific gate server 200.

The main server 400 communicates with the plurality of cloud pockets 300. Here, the main server 400 is provided in plurality, and the plurality of main servers 400 share data stored in each of the main servers 400. In this case, the plurality of main servers 400 may share (or synchronize) data stored (or registered) in each of the main servers 400 with each other.

Furthermore, the main server 400 is configured as (or includes) a database server (not shown).

The main server 400 connects to (or accesses/communicates with) at least one cloud pocket 300 among the plurality of cloud pockets 300. Here, a connection

between the cloud pocket 300 and the main server 400 is configured as bidirectional communication, and they communicate using a preset TCP/IP protocol. In this case, the TCP/IP protocol is custom-developed, and protocol information thereof is private (or non-disclosed).

5 The main server 400 is configured to strictly avoid using general public server protocols (including, for example, HTTP, TELNET, FTP, and the like), thereby preventing any access through publicly authorized protocols from external sources.

 Furthermore, the main server 400 is connected to the cloud pocket 300 through bidirectional communication. In a server-client architecture, the main server 400 is
10 configured as a server, and the cloud pocket 300 is configured as a client.

 Additionally, when connecting via TCP/IP, the main server 400 keeps connection port numbers private.

 That is, the main server 400 does not permit any access other than from the cloud pocket 300.

15 Security functions of the main server 400 are implemented through an IP address and a port number of the main server 400, device authentication of the cloud pocket 300, a custom protocol, and the like.

 Since the main server 400 is inaccessible from the outside, it transmits and receives data (or information) with the cloud pocket 300 in an unencrypted state.

20 As shown in FIG. 3, the main server 400 manages a database comprising vault IDs and tables corresponding to the vault IDs.

 That is, the main server 400 performs a function of reading or modifying a specific record of a specific table in a specific database.

For example, as shown in FIG. 3, the main server 400 matches a member table with a first vault ID (or a vault ID corresponding to a first vault). An external device (e.g., the cloud pocket 300) is configured to access a record of the member table in the main server 400 by using the first vault ID instead of using the actual "member" table name. Within the main server 400, the table (or table name) corresponding to the vault ID is used for internal processing.

Furthermore, the main server 400 interworks with the cloud pocket 300 to store (or register) a result of performing a function (including, for example, a result of performing a block generation function, a block modification function, a block inquiry function, and the like) regarding a specific request statement from the cloud pocket 300 into a specific table in the database within the main server 400.

In addition, the main server 400 provides a specific vault ID corresponding to the specific table where the result of performing the function is stored to the cloud pocket 300. In this case, instead of directly providing information related to the table where the function result is stored to the cloud pocket 300, the main server 400 provides a preset vault ID corresponding to the table. This is configured to prevent the cloud pocket 300 from identifying direct information regarding data (or the result of performing the function) stored in the database of the main server 400.

Furthermore, the main server 400 interworks with each of the remaining main servers 400, excluding the corresponding one main server 400 among the plurality of main servers 400, to respectively store (or share, register, or update) a result of performing a function (including, for example, a result of performing a block

generation function, a block modification function, a block inquiry function, and the like) in databases respectively configured in the remaining main servers 400.

As such, the main server 400 can prevent intrusion by external hacking through the following features:

- 5 1. The main server 400 keeps only one port open, and the port number is private and cannot be known from the outside.
2. The main server 400 uses a proprietary protocol that is unknown to the outside.
3. No TCP/IP communication other than the connection with the
10 cloud pocket 300 is connected to the main server 400. Since even protocols such as SSH or TFTP are not connected, external hacking cannot gain access using publicly authorized protocols.
4. Communication between the main servers 400 is performed using
15 a single port. However, this communication allows only main server URLs and fundamentally blocks other external accesses.

As described above, the user terminal 100 encrypts original data using an authentication key to generate an encrypted block, and registers request information including the encrypted block and a timestamp into the gate server 200. Then, the cloud pocket 300 accesses the gate server 200 to confirm the request information,
20 decrypts the encrypted block included in the request information to restore the original data, and may store the original data as a new block in the blockchain or modify existing data into the original data.

In addition, as described above, after a user logs in to the web server using an ID and a password through the user terminal 100, the user terminal 100 receives an authentication block provided by the web server upon request and registers the received authentication block into the gate server 200. Then, the cloud pocket 300
5 verifies authentication by checking whether the authentication block registered in the gate server 200 by the user terminal 100 matches the authentication block registered in the gate server 200 by the web server. If authentication is successful, the cloud pocket 300 generates an authentication key for which a validity period is set and registers it in the gate server 200. The user terminal 100 can then register the original
10 data in the form of a block into the cloud blockchain by interworking with the gate server 200 and the cloud pocket 300 using the authentication key registered in the gate server 200.

Hereinafter, a method for controlling the cloud blockchain system 10 according to the present invention will be described in detail with reference to FIGS.
15 1 to 6.

FIGS. 4 to 6 are flowcharts illustrating the method for controlling the cloud blockchain system according to an embodiment of the present invention.

First, the user terminal 100 logs in to (or accesses/connects to) a web server (not shown) using an identifier (ID) (or a device ID of the user terminal 100) and a
20 password. In this case, the user terminal 100 may be in a state of having signed up as a member to use a service provided by the web server.

Furthermore, the user terminal 100 requests the web server to generate an authentication block according to a request of a user of the user terminal 100.

In response to the authentication block generation request (or authentication block generation request information) of the user terminal 100, the web server generates an authentication block comprising the ID (or the device ID of the user terminal 100) and the password associated with the user terminal 100, and provides
5 (or transmits) the generated authentication block to the user terminal 100.

In addition, the web server registers (or stores) the generated authentication block in a gate server 200 (or an arbitrary table among a plurality of tables within the gate server 200) interworking with the web server. In this case, the gate server 200 registers (or stores) the authentication block provided from the web server in any one
10 table among a plurality of tables (or a plurality of vaults) within the gate server 200.

Furthermore, the gate server 200 is configured to interwork with each of remaining gate servers 200, excluding the corresponding gate server 200 itself, among the total plurality of gate servers 200, to respectively store (or register/update) the authentication block in the remaining gate servers 200.

15 As an example, a first user terminal 100 logs in to a first web server (not shown) for trading virtual currency (or cryptocurrency) using a first ID and a first password of the first user terminal, and transmits first authentication block generation request information to the first web server.

Then, the first web server receives the first authentication block generation request information transmitted from the first user terminal, generates a first
20 authentication block comprising the first ID and the first password based on the received first authentication block generation request information, and transmits the generated first authentication block to the first user terminal.

Furthermore, the first web server stores the generated first authentication block in a second gate server 200 interworking with the first web server among the plurality of gate servers 200.

As another example, a second user terminal 100 logs in to a second web server
5 for trading virtual currency (or cryptocurrency) using a second ID and a second password of the second user terminal, and transmits second authentication block generation request information to the second web server.

Then, the second web server receives the second authentication block generation request information transmitted from the second user terminal, generates a
10 second authentication block comprising the second ID and the second password based on the received second authentication block generation request information, and transmits the generated second authentication block to the second user terminal.

In addition, the web server stores the generated second authentication block in a 23rd gate server 200 interworking with the second web server among the plurality
15 of gate servers 200 (S410).

Thereafter, the user terminal 100 receives the authentication block provided (or transmitted) from the web server in response to the previous authentication block generation request (or the authentication block generation request information).

Furthermore, the user terminal 100 accesses (or connects to) one specific gate
20 server 200 among the plurality of gate servers 200 to be issued an authentication key (or an authentication key block) from the cloud pocket 300. In this case, the user terminal 100 and the gate server 200 communicate using a preset HTTP protocol (or HTTPS protocol).

In addition, the user terminal 100 registers (or stores) the received authentication block in the connected specific gate server 200.

That is, the user terminal 100 registers (or stores) the authentication block into any one specific table (or a specific vault) among a plurality of tables (or vaults) within
5 the specific gate server 200.

As an example, the first user terminal receives the first authentication block transmitted from the first web server in response to the previous first authentication block generation request.

Then, the first user terminal accesses the second gate server among the
10 plurality of gate servers 200.

Furthermore, the first user terminal registers the first authentication block into a second table among a plurality of tables managed by the second gate server.

As another example, the second user terminal receives the second authentication block transmitted from the second web server in response to the
15 previous second authentication block generation request.

In addition, the second user terminal 100 accesses the 23rd gate server among the plurality of gate servers 200.

Further, the second user terminal 100 registers the second authentication block into a 23rd table among a plurality of tables managed by the 23rd gate server (S420).

20 Thereafter, the cloud pocket 300 connects to (or communicates with/accesses) the specific gate server 200 at a preset interval, and then checks whether data (including, for example, an authentication block and the like) is in a registered (or stored) state in a plurality of tables (or vaults) within the specific gate server 200.

Here, the cloud pocket 300 is any one of a plurality of cloud pockets 300 and is in a state preset to communicate with the specific gate server 200, and a connection between the cloud pocket 300 and the specific gate server 200 may be unidirectional communication in which only the cloud pocket 300 accesses the specific gate server
5 200.

As a result of the checking, if the data is in a registered state in at least one table among the plurality of tables, the cloud pocket 300 confirms the data registered in the table.

That is, the cloud pocket 300 regularly accesses the specific gate server 200 at
10 constant time intervals (or periods) while running a loop (or a thread) and checks whether there is a task (or data) to be processed by the cloud pocket 300. Here, the task to be processed by the cloud pocket 300 may be a function related to data registered in a table within the specific gate server 200 by an arbitrary user terminal 100 or the like.

15 As an example, a third cloud pocket 300 accesses the second gate server at a preset interval and checks whether arbitrary data is in a registered state in a plurality of tables managed by the second gate server.

As another example, the third cloud pocket 300 accesses the 23rd gate server at a preset interval and checks whether arbitrary data is in a registered state in a
20 plurality of tables managed by the 23rd gate server (S430).

Thereafter, if the data is an authentication block, the cloud pocket 300 interworks with the specific gate server 200 to compare (or check) whether the authentication block registered in the specific gate server 200 by the web server

matches the authentication block registered in the specific gate server 200 by the user terminal 100.

As an example, when the first authentication block is in a registered state in the second table managed by the second gate server, the third cloud pocket compares
5 whether the first authentication block previously registered in the second gate server by the first web server matches the first authentication block registered in the second table of the second gate server by the first user terminal.

As another example, when the second authentication block is in a registered state in the 23rd table managed by the 23rd gate server, the third cloud pocket compares
10 whether the second authentication block previously registered in the 23rd gate server by the second web server matches the second authentication block registered in the 23rd table of the 23rd gate server by the second user terminal (S440).

As a result of the comparison (or the check), if the authentication block registered (or stored) in the specific gate server 200 by the web server does not match
15 the authentication block registered (or stored) in the specific gate server 200 by the user terminal 100, the cloud pocket 300 registers (or stores) authentication failure information in the table (or the vault) of the specific gate server 200, and terminates the connection with the specific gate server 200. Here, the cloud pocket 300 stores the authentication failure information in addition to data previously stored (including,
20 for example, the authentication block and the like) in the table of the specific gate server 200, or deletes the data stored in the table and then stores the authentication failure information.

In addition, the user terminal 100 confirms (or displays/outputs) the authentication failure information registered (or stored) in the table (or the vault) of the specific gate server 200. In this case, while the user terminal 100 is in a state of being connected to the specific gate server 200, as new information (including, for example, authentication failure information) is registered in the table of the specific gate server 200, the user terminal 100 may receive notification information (or guidance information) transmitted from the specific gate server 200 and check the authentication failure information registered in the table based on the received notification information. Furthermore, the user terminal 100 may additionally display guidance information indicating that a new authentication block should be issued through the previous steps.

As an example, as a result of the comparison, if the first authentication block registered in the second gate server by the first web server does not match the first authentication block registered in the second table of the second gate server by the first user terminal, the third cloud pocket cumulatively stores the authentication failure information in the second table of the second gate server and terminates the connection with the second gate server.

In addition, the first user terminal displays the authentication failure information registered in the second table within the second gate server being accessed.

As another example, as a result of the comparison, when the second authentication block registered in the 23rd gate server by the second web server does not match the second authentication block registered in the 23rd table of the 23rd gate server by the second user terminal, the second cloud pocket stores authentication

failure information in the 23rd table of the 23rd gate server and terminates the connection with the 23rd gate server.

Furthermore, the second user terminal displays the authentication failure information registered in the 23rd table within the 23rd gate server being accessed
5 (S450).

In addition, as a result of the comparison (or the check), when the authentication block registered (or stored) in the specific gate server 200 by the web server matches the authentication block registered (or stored) in the specific gate server 200 by the user terminal 100, the cloud pocket 300 confirms the ID (or device ID), the
10 password, and the like included in the authentication block.

Additionally, the cloud pocket 300 generates (or issues) an authentication key based on the confirmed ID, password, current time, and a preset validity period. In this case, the authentication key includes the ID (or device ID), the password, and a timestamp including validity period information for the authentication key. Here, the
15 validity period may be a preset value (including, for example, 3 minutes, 5 minutes, 10 minutes, and the like).

As an example, when the first authentication block registered in the second gate server by the first web server matches the first authentication block registered in the second table of the second gate server by the first user terminal, the third cloud
20 pocket confirms the first ID and the first password included in the first authentication block, and generates a first authentication key consisting of the confirmed first ID, first password, and a first timestamp. Here, the first timestamp is displayed (or managed/configured) in an integer format and may be a value (e.g., 1:15:20 PM on

March 1, 2025) to which a preset validity period (e.g., 5 minutes) is applied from a current time (e.g., 1:10:20 PM on March 1, 2025).

As another example, when the second authentication block registered in the 23rd gate server by the second web server matches the second authentication block registered in the 23rd table of the 23rd gate server by the second user terminal, the
5 third cloud pocket confirms the second ID and the second password included in the second authentication block, and generates a second authentication key consisting of the confirmed second ID, second password, and a second timestamp. Here, the second timestamp is displayed (or managed/configured) in an integer format and may
10 be a value (e.g., 7:25:30 PM on March 1, 2025) to which a preset validity period (e.g., 5 minutes) is applied from a current time (e.g., 7:20:30 PM on March 1, 2025) (S460).

Thereafter, the cloud pocket 300 generates a user account encryption/decryption key. Here, the user account encryption/decryption key may be used in a preset state or may be generated and used upon request, and comprises (or
15 includes) an inquiry account key (or a decryption key), a modification account key (or an encryption key), and a computation key.

Specifically, the cloud pocket 300 generates the user account encryption/decryption key, which is a byte key of a preset size (e.g., 256 bytes times 3), and the byte key indicates that the key unit is in bytes.

20 As an example, the third cloud pocket generates a first user account encryption/decryption key consisting of a first inquiry account key of a preset size (e.g., 256 bytes) for interpreting (or decrypting) encrypted data, a first modification account key of a preset size (e.g., 256 bytes) for encrypting data, and a first computation key

of a preset size (e.g., 256 bytes) for making interpretation via patterns impossible by using a computational encoding scheme.

As another example, the third cloud pocket generates a second user account encryption/decryption key consisting of a second inquiry account key of a preset size
5 (e.g., 256 bytes), a second modification account key of a preset size (e.g., 256 bytes), and a second computation key of a preset size (e.g., 256 bytes) (S470).

Thereafter, the cloud pocket 300 registers (or stores) an authentication key block including the generated authentication key and the generated user account encryption/decryption key in the table (or the vault) of the specific gate server 200.
10 Here, the cloud pocket 300 stores the authentication key block in addition to data previously stored (including, for example, the authentication block and the like) in the table of the specific gate server 200, or deletes the data stored in the table and then stores the authentication key block.

Furthermore, the cloud pocket 300 interworks with one main server 400
15 among a plurality of main servers 400 to store (or register) the authentication key block in one record within one table among a plurality of tables in a database (not shown) of the main server 400. In this case, as the authentication key block is stored in a specific record within a specific table of the database, the cloud pocket 300 receives a vault ID (or a vault ID corresponding to the specific table) provided from the main
20 server 400. Here, when the cloud pocket 300 stores the authentication key block in the table within the specific gate server 200 accessed at a preset interval, the cloud pocket 300 may store the vault ID together with the authentication key block.

Furthermore, the user terminal 100 checks (or displays, outputs, or downloads) the authentication key block registered (or stored) in the table (or the vault) of the specific gate server 200. In this case, while the user terminal 100 is connected to the specific gate server 200, as new information (including, for example, the authentication key block) is registered in the table, the user terminal 100 may receive notification information (or guidance information) transmitted from the specific gate server 200 and check the authentication key block registered in the table based on the received notification information.

As an example, the third cloud pocket 300 generates a first authentication key block including the generated first authentication key and the generated first user account encryption/decryption key, and cumulatively stores the first authentication key block in a second table of a second gate server.

In addition, the third cloud pocket interworks with a 14th main server 400 among a plurality of main servers 400 to store the first authentication key block in a 14th record within a 4th table in a database of the 14th main server.

Then, the 14th main server transmits a 4th vault ID, which corresponds to the 4th table where the first authentication key block is stored, to the third cloud pocket.

The third cloud pocket then receives the 4th vault ID transmitted from the 14th main server.

Subsequently, the first user terminal checks the first authentication key block registered in the second table within the second gate server being accessed, and downloads and temporarily stores the first authentication key block.

As another example, the third cloud pocket generates a second authentication key block including the generated second authentication key and the generated second user account encryption/decryption key, and stores the second authentication key block in a 23rd table of a 23rd gate server.

5 In addition, the third cloud pocket interworks with the 14th main server 400 to store the second authentication key block in a 34th record within a 34th table in the database of the 14th main server.

Then, the 14th main server transmits a 34th vault ID corresponding to the 34th table to the third cloud pocket.

10 The third cloud pocket then receives the 34th vault ID transmitted from the 14th main server.

In addition, the second user terminal checks the second authentication key block registered in the 23rd table within the accessed 23rd gate server, and downloads and temporarily stores the second authentication key block (S480).

15 Thereafter, in order to store (or generate/modify) data (or information) in a cloud blockchain managed by the cloud pocket 300 or to inquire about data (or information) registered in the cloud blockchain, the user terminal 100 encrypts original data to be registered in the cloud blockchain using the user account encryption/decryption key included in the authentication key block, thereby generating
20 an encrypted block (or encrypted data). Here, the original data (or a target block/target data/data block to be encrypted) includes not only content (or information/data) but also a requirement (including, for example, generation request information, modification request information, inquiry request information, and the

like). Further, the encrypted block may include a header value (or a header value related to encryption) in addition to the original data.

As an example, the first user terminal generates a first encrypted block by encrypting first original data (for example, "I want to know the total amount in my
5 wallet"), which includes information for checking a balance in a cryptocurrency wallet associated with a person who is a first user of the first user terminal, using a first user account encryption/decryption key included in the first authentication key block.

As another example, the second user terminal generates a second encrypted block by encrypting second original data (for example, "Transfer 1 Bitcoin from the
10 Bitcoins in my wallet to another person's cryptocurrency wallet"), which includes information for transferring 1 Bitcoin kept in a cryptocurrency wallet of another person who is a second user of the second user terminal to the cryptocurrency wallet of the person, using a second user account encryption/decryption key included in the second authentication key block (S490).

15 Thereafter, the user terminal 100 generates a request statement (or a command statement/request statement/request statement block) including the generated encrypted block, the authentication key included in the authentication key block, and the like.

Additionally, the user terminal 100 accesses (or communicates with/connects
20 to) a specific gate server 200 among the plurality of gate servers 200.

Furthermore, the user terminal 100 registers (or stores) the generated request statement in the connected specific gate server 200.

That is, the user terminal 100 registers (or stores) the request statement in any one other specific table (or another specific vault) among a plurality of tables (or vaults) within the specific gate server 200.

As an example, the first user terminal generates a first request statement
5 including the generated first encrypted block, the first authentication key included in the first authentication key block, and the like.

Then, the first user terminal accesses the second gate server among the plurality of gate servers 200 and registers the generated first request statement in a second table among a plurality of tables managed by the accessed second gate server.

10 As another example, the second user terminal generates a second request statement including the generated second encrypted block, the second authentication key included in the second authentication key block, and the like.

Then, the second user terminal accesses the 23rd gate server among the plurality of gate servers 200 and registers the generated second request statement in a
15 24th table among a plurality of tables managed by the accessed 23rd gate server (S500).

Thereafter, the cloud pocket 300 connects to (or communicates with/accesses) the specific gate server 200 at the preset interval, and then checks whether data (including, for example, a request statement) is in a registered (or stored) state in a plurality of tables (or vaults) within the specific gate server 200.

20 As a result of the checking, if the data is in a registered state in at least one table (or vault) among the plurality of tables (or vaults), the cloud pocket 300 confirms the data registered in the table (or vault).

That is, the cloud pocket 300 regularly accesses the specific gate server 200 at constant time intervals (or cycles) while running a loop (or a thread) and checks whether there is a task (or data) to be processed by the cloud pocket 300. Here, the task to be processed by the cloud pocket 300 may be a function related to data registered in a table within the specific gate server 200 by an arbitrary user terminal 100 or the like.

As an example, a third cloud pocket 300 accesses the second gate server at a preset interval and checks whether arbitrary data is in a registered state in a plurality of tables managed by the second gate server.

10 As another example, the third cloud pocket 300 accesses the 23rd gate server at a preset interval and checks whether arbitrary data is in a registered state in a plurality of tables managed by the 23rd gate server (S510).

Thereafter, if the data is a request statement, the cloud pocket 300 primarily verifies (or determines) validity of the data (or the request statement) based on a
15 timestamp included in the request statement.

That is, when the data is a request statement, the cloud pocket 300 verifies (or determines) whether a time difference between a current time and a timestamp included in the request statement has passed (or exceeded) a preset reference value (including, for example, 3 seconds, 5 seconds, 10 seconds, and the like) based on the
20 current time and the timestamp, in order to verify the validity of the data (or the request statement).

As an example, when the first request statement is in a registered state in the second table among the plurality of tables managed by the second gate server, the third

cloud pocket calculates a first time difference between the current time and a first timestamp (e.g., 1:15:20 PM on March 1, 2025) included in the first request statement, and determines whether the calculated first time difference has passed a preset first reference value (e.g., 5 seconds).

5 As another example, when the second request statement is in a registered state in the 24th table among the plurality of tables managed by the 23rd gate server, the third cloud pocket calculates a second time difference between the current time and a second timestamp (e.g., 7:25:30 PM on March 1, 2025) included in the second request statement, and determines whether the calculated second time difference has passed a
10 preset second reference value (e.g., 3 seconds) (S520).

 As a result of the validity verification of the data (or the result of determining the validity of the data/request statement), if the request statement is determined to be invalid, the cloud pocket 300 registers (or stores) information indicating that the validity verification of the data has failed in the corresponding table (or vault) of the
15 specific gate server 200, and terminates the connection with the specific gate server 200. Here, the cloud pocket 300 stores the information indicating the failure of the validity verification in addition to data previously stored (including, for example, an authentication block, a request statement, and the like) in the table of the specific gate server 200, or deletes the data stored in the table and then stores the information
20 indicating the failure of the validity verification.

 Specifically, if the time difference between the current time and the timestamp included in the request statement has passed (or exceeded) the preset reference value (including, for example, 3 seconds, 5 seconds, 10 seconds, and the like), the cloud

pocket 300 registers (or stores) the information indicating that the validity verification of the data has failed in the corresponding table (or vault) of the specific gate server 200, and terminates the connection with the specific gate server 200.

In addition, the user terminal 100 checks (or displays/outputs) information
5 indicating that the validity verification of the corresponding data registered (or stored)
in the corresponding table (or vault) of the specific gate server 200 has failed. In this
case, while the user terminal 100 is in a state of being connected to the specific gate
server 200, as new information (including, for example, information indicating that the
validity verification of the data has failed) is newly registered in the corresponding
10 table of the specific gate server 200, the user terminal 100 receives other notification
information (or other guidance information) transmitted from the specific gate server
200. The user terminal 100 may then check the information indicating that the
validity verification of the data registered in the table of the specific gate server 200
has failed based on the received other guidance information. Furthermore, the user
15 terminal 100 may additionally display guidance information indicating that a new
authentication key should be issued through the previous steps.

As an example, as a result of the validity determination for the first request
statement, when the first time difference has passed the preset first reference value
(e.g., 5 seconds), the third cloud pocket 300 cumulatively stores information indicating
20 that the validity verification of the first request statement has failed in the second table
of the second gate server, and terminates the connection with the second gate server.

Additionally, the first user terminal displays the information indicating that the validity verification of the first request statement registered in the second table within the accessed second gate server has failed.

5 As another example, as a result of the validity determination for the second request statement, when the calculated second time difference has passed the preset second reference value (e.g., 3 seconds), the third cloud pocket stores information indicating that the validity verification of the second request statement has failed in the 24th table of the 23rd gate server, and terminates the connection with the 23rd gate server.

10 Also, the second user terminal displays the information indicating that the validity verification of the second request statement registered in the 24th table within the accessed 23rd gate server has failed (S530).

In addition, as a result of the validity verification of the data (or the result of determining the validity of the data/request statement), if the request statement is
15 determined to be valid, the cloud pocket 300 secondarily checks (or determines) whether a header value included in the request statement matches (or satisfies) a preset specification (or condition). Here, the header value may be a value generated in a process of encrypting an encrypted block included in the request statement.

Specifically, if the time difference between the current time and the timestamp
20 included in the request statement has not passed (or has not exceeded) the preset reference value (including, for example, 3 seconds, 5 seconds, 10 seconds, and the like), the cloud pocket 300 determines that the request statement is valid, and

secondarily checks (or determines) whether a header value included in the request statement matches (or satisfies) a preset specification (or condition).

As an example, as a result of the validity determination for the first request statement, when the first time difference has not passed the preset first reference value
5 (e.g., 5 seconds), the third cloud pocket 300 checks whether a first header value included in the first request statement matches the preset specification.

As another example, as a result of the validity determination for the second request statement, when the calculated second time difference has not passed the preset second reference value (e.g., 3 seconds), the third cloud pocket checks whether a
10 second header value included in the second request statement matches the preset specification (S540).

As a result of the check (or the determination), if the header value included in the request statement does not match (or satisfy) the preset specification (or condition), the cloud pocket 300 registers (or stores) information indicating that the specification
15 verification of the data has failed in the corresponding table (or vault) of the specific gate server 200, and terminates the connection with the specific gate server 200. Here, the cloud pocket 300 stores the information indicating that the specification verification of the data has failed in addition to data previously stored (including, for example, an authentication block, a request statement, and the like) in the table of the
20 specific gate server 200, or deletes the data stored in the table and then stores the information indicating that the specification verification of the data has failed.

Furthermore, the user terminal 100 checks (or displays/outputs) the information indicating that the specification verification of the data registered (or

stored) in the corresponding table (or vault) of the specific gate server 200 has failed. In this case, while the user terminal 100 is in a state of being connected to the specific gate server 200, as new information (including, for example, the information indicating that the specification verification of the data has failed) is newly registered in the table
5 of the specific gate server 200, the user terminal 100 receives other notification information (or other guidance information) transmitted from the specific gate server 200. The user terminal 100 may then check the information indicating that the specification verification of the data has failed based on the received other guidance information. In addition, the user terminal 100 may additionally display guidance
10 information indicating that a new authentication key must be issued through previous steps.

As an example, when the first header value included in the first request statement does not satisfy the preset specification, the third cloud pocket 300 cumulatively stores information indicating that the specification verification of the
15 first request statement has failed in the second table of the second gate server, and terminates the connection with the second gate server.

In addition, the first user terminal displays the information indicating that the specification verification of the first request statement, which is registered in the second table within the accessed second gate server, has failed.

20 As another example, when the second header value included in the second request statement does not satisfy the preset specification, the third cloud pocket stores information indicating that the specification verification of the second request

statement has failed in the 24th table of the 23rd gate server, and terminates the connection with the 23rd gate server.

Further, the second user terminal displays the information indicating that the specification verification of the second request statement registered in the 24th table
5 within the accessed 23rd gate server has failed (S550).

Furthermore, as a result of the checking (or the result of the determining), when a header value included in a request statement matches (or satisfies) a preset specification (or condition), the cloud pocket 300 tertiarily decrypts an encrypted block included in the request statement using a preset user account encryption/decryption
10 key (or the user account encryption/decryption key previously generated based on the authentication block associated with the user terminal 100) to generate (or restore) original data.

That is, as a result of the secondary specification verification of the data, when the header value included in the request statement matches (or satisfies) the preset
15 specification (or condition), the cloud pocket 300 separates the request statement into an authentication key and an encrypted block, and decrypts the separated encrypted block using the user account encryption/decryption key to generate (or restore) the original data.

As an example, when the first header value included in the first request
20 statement satisfies the preset specification, the third cloud pocket separates the first request statement into the first authentication key and the first encrypted block.

In addition, the third cloud pocket decrypts the separated first encrypted block using the first user account encryption/decryption key to generate the first original data (for example, "I want to know the total amount in my wallet").

5 As another example, when the second header value included in the second request statement satisfies the preset specification, the third cloud pocket separates the second request statement into the second authentication key and the second encrypted block.

In addition, the third cloud pocket 300 decrypts the separated second encrypted block using the second user account encryption/decryption key to generate
10 the second original data (for example, "Transfer 1 Bitcoin from the Bitcoins in my wallet to a specific person's cryptocurrency wallet") (S560).

Thereafter, the cloud pocket 300 performs a block generation function, a block modification function, a block inquiry function (or a block content inquiry function), and the like in a cloud blockchain based on the original data.

15 That is, the cloud pocket 300 performs parsing on the original data and identifies a meaning (or content) of the original data according to the parsing (or a result of the parsing).

Furthermore, when a requirement included in the request statement is a new block generation request (or when the identified meaning/content is a new block
20 generation request), the cloud pocket 300 checks whether block generation is to be performed in the cloud pocket 300 among a pocket server list managed by the cloud pocket 300.

Specifically, the cloud pocket 300 checks, through the pocket server list, whether the new block generation request is a task (or function) to be performed in the cloud pocket 300, which is a local server. Here, the pocket server list includes information on which cloud pocket 300 a wallet (or a cryptocurrency/virtual currency
5 wallet) corresponding to a user account is stored (or managed) in, and includes cloud pocket information (or an address/URL of the cloud pocket 300) for each piece of wallet information.

As a result of the checking, if the block generation is to be performed in the cloud pocket 300, the cloud pocket 300 performs the block generation function in the
10 cloud blockchain based on the original data.

Specifically, when it is confirmed as a result of the checking that the new block generation request is a task (or function) to be performed in the cloud pocket 300, which is the local server that has checked the request statement, the cloud pocket 300 performs the block generation function based on the original data in the cloud pocket
15 300. In this case, the cloud pocket 300 may perform a block generation function of newly creating a user's wallet associated with the request statement and managing information associated with the newly created wallet.

In addition, the cloud pocket 300 updates a result of performing the block generation function associated with new block generation (or updates according to the
20 new block generation in the cloud blockchain).

Furthermore, the cloud pocket 300 updates the result of performing the block generation function related to the new block generation in each of the remaining cloud

pockets 300, excluding the cloud pocket 300 itself among all the cloud pockets 300, by using threads.

In addition, as a result of the checking, in the case where the block generation is to be performed in another cloud pocket 300 (or in the case where the block generation is not in the local server that checked the request statement), the cloud pocket 300 identifies another cloud pocket 300 in charge of performing the block generation function based on the pocket server list managed by the cloud pocket 300, and provides the request statement (or the original data) to the identified another cloud pocket 300.

Accordingly, the other cloud pocket 300 performs the block generation function in the cloud blockchain based on the request statement (or the original data). In this case, the other cloud pocket 300 may perform a block generation function of newly creating a user's wallet associated with the request statement and managing information associated with the newly created wallet.

Furthermore, after completing the performance of the block generation function, the cloud pocket 300 updates a result of performing the block generation function related to the new block generation (or a result of performing the block generation function related to the new block generation in the other cloud pocket 300) (or updates according to the new block generation in the cloud blockchain). In this case, the cloud pocket 300 may update result information regarding the newly created user's wallet according to the request statement.

Additionally, the cloud pocket 300 updates the results of performing the block generation function related to the new block generation in each of the remaining cloud

pockets 300, excluding the cloud pocket 300 itself among all the cloud pockets 300, by using threads.

Furthermore, when a requirement included in the request statement is a block modification request (or when the identified meaning is a block modification request),
5 the cloud pocket 300 checks whether a primary modification cloud pocket 300 is local based on the pocket server list managed by the cloud pocket 300.

That is, regarding the block modification request, the cloud pocket 300 checks, through the pocket server list, whether the primary modification cloud pocket 300 is a task (or function) to be performed in the cloud pocket 300, which is the local server.

10 As a result of the checking, when the primary modification is to be performed in the cloud pocket 300, the cloud pocket 300 performs a primary block modification function in the cloud blockchain based on the original data. Here, the primary block modification function indicates a block modification function targeting a target party.

That is, when it is confirmed that the primary modification cloud pocket 300
15 is a task (or function) to be performed in the cloud pocket 300, which is the local server that checked the request statement, the cloud pocket 300 performs a primary block modification function in the cloud blockchain based on the original data. In this case, the cloud pocket 300 may perform a block modification function of deducting an amount of cryptocurrency included in the request statement from a user's wallet
20 associated with the request statement.

Furthermore, as a result of the checking, in the case where the primary modification is to be performed in another cloud pocket 300 (or in the case where the primary block modification is not in the local server that checked the request

statement), the cloud pocket 300 identifies another cloud pocket 300 in charge of performing the primary block modification function based on the pocket server list managed in the cloud pocket 300, and provides the request statement (or the original data) to the identified another cloud pocket 300.

5 Accordingly, the other cloud pocket 300 performs the primary block modification function in the cloud blockchain based on the request statement (or the original data). In this case, the other cloud pocket 300 may perform a block modification function of deducting the amount of cryptocurrency included in the request statement from the user's wallet associated with the request statement.

10 In addition, the cloud pocket 300 interworks with the other cloud pocket 300 to ascertain (or confirm) whether the performance of the primary block modification function in the other cloud pocket 300 is completed.

 Furthermore, after the primary block modification function is completed, the cloud pocket 300 checks whether a secondary modification cloud pocket 300 is local
15 based on the pocket server list managed in the cloud pocket 300.

 That is, regarding the block modification request, the cloud pocket 300 determines, through the pocket server list, whether the secondary modification cloud pocket 300 is a task (or function) to be performed in the cloud pocket 300, which is the local server.

20 As a result of the determination, when the secondary modification is to be performed in the cloud pocket 300, the cloud pocket 300 performs a secondary block modification function in the cloud blockchain based on the original data. Here, the

secondary block modification function indicates a block modification function targeting a counterparty (or a recipient party).

Specifically, when it is determined that the secondary modification cloud pocket 300 is a task (or function) to be performed in the cloud pocket 300, which is the local server that has checked the request statement, the cloud pocket 300 performs the secondary block modification function in the cloud blockchain based on the original data. In this case, the other cloud pocket 300 (or the local cloud pocket 300) may perform a block modification function of deducting the amount of cryptocurrency included in the request statement from the user's wallet associated with the request statement.

Furthermore, as a result of the determination, in the case where the secondary modification is to be performed in another cloud pocket 300 (or in the case where the secondary block modification is not in the local server that checked the request statement), the cloud pocket 300 identifies another cloud pocket 300 (specifically, a different cloud pocket from the one used in the primary modification) in charge of performing the secondary block modification function based on the pocket server list managed in the cloud pocket 300, and provides the request statement (or the original data) to the identified another cloud pocket 300.

Accordingly, the other cloud pocket 300 performs the secondary block modification function in the cloud blockchain based on the request statement (or the original data). In this case, the other cloud pocket 300 may perform a block modification function of deducting an amount of cryptocurrency included in the request statement from a user's wallet associated with the request statement.

In addition, the cloud pocket 300 interworks with the other cloud pocket 300 to ascertain (or confirm) whether the performance of the secondary block modification function in the other cloud pocket 300 is completed.

Furthermore, after completing the performance of the primary and secondary
5 block modification functions, the cloud pocket 300 updates a result of performing the block modification function related to the block modification (e.g., results from the local cloud pocket 300, the other cloud pocket 300, etc.) (or updates according to the block modification in the cloud blockchain). In this case, the cloud pocket 300 may update result information regarding changes in the wallets of the target party and the
10 counterparty according to the transfer of cryptocurrency based on the request statement.

Additionally, the cloud pocket 300 updates the results of performing the block modification function related to the block modification in each of remaining cloud pockets 300, excluding the cloud pocket 300 itself among all the cloud pockets 300, by using threads.

15 Furthermore, when a requirement included in the request statement is a block inquiry request (or when the identified meaning is a block inquiry request), the cloud pocket 300 checks whether the block inquiry is to be performed in the cloud pocket 300 among the pocket server list managed by the cloud pocket 300.

That is, the cloud pocket 300 checks, through the pocket server list, whether
20 the block inquiry request is a task (or function) to be performed in the cloud pocket 300, which is the local server.

As a result of the checking, if the block inquiry is to be performed in the cloud pocket 300, the cloud pocket 300 performs a function of checking information (or

information/data within a block) corresponding to the block inquiry request in the cloud blockchain.

Specifically, when it is confirmed that the block inquiry request is a task (or function) to be performed in the cloud pocket 300, which is the local server that has
5 checked the request statement, the cloud pocket 300 performs a block inquiry function based on the original data in the cloud blockchain. In this case, the cloud pocket 300 may perform an inquiry function of checking a balance remaining in a user's wallet associated with the request statement.

Furthermore, the cloud pocket 300 updates a result of performing the block
10 inquiry function (or updates according to the block inquiry in the cloud blockchain).

In addition, the cloud pocket 300 updates the results of performing the block inquiry function in each of remaining cloud pockets 300, excluding the cloud pocket 300 itself among all the cloud pockets 300, by using threads.

Moreover, as a result of the checking, in the case of a block inquiry in another
15 cloud pocket 300 (or in the case where the block inquiry is not in the local server that checked the request statement), the cloud pocket 300 identifies another cloud pocket 300 in charge of performing the block inquiry function based on the pocket server list managed in the cloud pocket 300, and provides the request statement (or the original data) to the identified another cloud pocket 300.

20 Accordingly, the other cloud pocket 300 performs the block inquiry function in the cloud blockchain based on the request statement (or the original data). In this case, the other cloud pocket 300 may perform the inquiry function of checking the balance remaining in the user's wallet associated with the request statement.

Furthermore, after completing the performance of the block inquiry function, the cloud pocket 300 updates the result of performing the block inquiry function (or the result of performing the block inquiry function related to the block inquiry in the other cloud pocket 300) (or an update according to the block inquiry in the cloud blockchain). In this case, the cloud pocket 300 may update information regarding the balance remaining in the user's wallet according to the request statement.

Additionally, the cloud pocket 300 updates the results of performing the block inquiry function in each of the remaining cloud pockets 300, excluding the cloud pocket 300 itself among all the cloud pockets 300, by using threads.

As an example, the third cloud pocket performs parsing on the first original data included in the first request statement, and identifies a first meaning (e.g., a balance inquiry) of the first original data (e.g., "I want to know the total amount in my wallet") according to the parsing.

Furthermore, when the first requirement included in the first request statement is a block inquiry, the third cloud pocket checks, through a third pocket server list managed by the third cloud pocket, whether the block inquiry is to be performed in the third cloud pocket (or whether a person's cryptocurrency wallet associated with the first request statement is set in the third cloud pocket).

As a result of the checking, if it is a block inquiry in the third cloud pocket (or the person's cryptocurrency wallet is set in the third cloud pocket), the third cloud pocket confirms fourth data (e.g., "the balance is 0.5 Bitcoin in the person's wallet") corresponding to the block inquiry request in the cloud blockchain.

In addition, the third cloud pocket updates a result of performing the block inquiry function related to the block inquiry.

Furthermore, the third cloud pocket updates the result of performing the block inquiry function in each of the remaining cloud pockets 300, excluding the third cloud pocket itself among all the cloud pockets 300, by using threads.

As another example, the third cloud pocket performs parsing on the second original data included in the second request statement, and identifies a second meaning (e.g., block modification) of the second original data (e.g., "Transfer 1 Bitcoin from the Bitcoins in my wallet to a specific person's cryptocurrency wallet") according to the parsing.

Furthermore, when the identified second meaning is block modification, the third cloud pocket checks, through the third pocket server list managed by the third cloud pocket, whether a primary modification related to the block modification request is a task to be performed in the local server (or checks whether another person's cryptocurrency wallet for the transfer is set in the third cloud pocket).

As a result of the checking, when the primary modification is to be performed in the third cloud pocket (or when the other person's cryptocurrency wallet for the transfer is set in the third cloud pocket), the third cloud pocket performs a primary block modification function (e.g., a block modification function by deducting 1 Bitcoin from the wallet associated with the other person) based on the second original data in the cloud blockchain.

In addition, the third cloud pocket 300 determines, based on the pocket server list managed in the third cloud pocket, whether a secondary modification related to the

block modification request is a task to be performed in the local server (or determines whether the cryptocurrency wallet of the specific person, who is the recipient, is in a state set in the third cloud pocket).

As a result of the determination, when the secondary modification is to be
5 performed in another cloud pocket, the third cloud pocket identifies a seventh cloud pocket 300 in charge of performing the secondary block modification function based on the pocket server list managed in the third cloud pocket, and transmits the second request statement (or the second original data) to the identified seventh cloud pocket.

Furthermore, the seventh cloud pocket receives the second request statement
10 (or the second original data) transmitted from the third cloud pocket, and performs the secondary block modification function (e.g., a block modification function by adding 1 Bitcoin to the wallet associated with the specific person) based on the received second request statement (or second original data).

In addition, the third cloud pocket updates a result of performing the block
15 modification function related to the block modification.

Further, the third cloud pocket updates the result of performing the block modification function in each of the remaining cloud pockets 300, excluding the third cloud pocket itself among all the cloud pockets 300, by using threads (S570).

Thereafter, the cloud pocket 300 connects to (or communicates with/accesses)
20 the specific gate server 200 at a preset interval, and then registers (or stores) a result of performing a function (including, for example, a result of performing a block generation function, a block modification function, a block inquiry function, and the like) corresponding to the request statement into the corresponding table (or vault) of

the specific gate server 200. Here, the cloud pocket 300 stores the result of performing the function in the table within the specific gate server 200 where data (including, for example, an authentication block, an authentication key block, a request statement, and the like) was previously registered in relation to the user terminal 100,
5 or deletes data being stored in the table and then stores the result of performing the function corresponding to the request statement.

Furthermore, the user terminal 100 checks (or displays/outputs) the result of performing the function registered (or stored) in the corresponding table (or vault) of the specific gate server 200. In this case, while the user terminal 100 is in a state of
10 being connected to the specific gate server 200, as new information (including, for example, the result of performing the function) is newly registered in the corresponding table of the specific gate server 200, the user terminal 100 may receive notification information (or guidance information) transmitted from the specific gate server 200 and check the result of performing the function registered in the table based
15 on the received notification information.

As an example, the third cloud pocket 300 accesses the second gate server at a preset interval, and cumulatively stores a result of performing a block inquiry function (e.g., Chul-soo's wallet balance is 0.5 Bitcoin) in a second table of the second gate server.

20 In addition, the first user terminal 100 displays the result of performing the block inquiry function (e.g., Chul-soo's wallet balance is 0.5 Bitcoin) stored in the second table within the accessed second gate server.

As another example, the third cloud pocket 300 accesses the 23rd gate server at a preset interval, and stores a result of performing a block modification function (e.g., 1 Bitcoin was normally transferred from Young-hee's wallet to Hong Gil-dong, and Young-hee's wallet balance is 12 Bitcoins) in a 24th table of the 23rd gate server.

5 Furthermore, the second user terminal 100 displays the result of performing the block modification function (e.g., 1 Bitcoin was normally transferred from Young-hee's wallet to Hong Gil-dong, and Young-hee's wallet balance is 12 Bitcoins) stored in the 24th table within the accessed 23rd gate server (S580).

 Thereafter, the cloud pocket 300 connects to (or accesses/communicates with)
10 any one main server 400 among a plurality of main servers 400, and stores (or registers) a result of performing a function (including, for example, a result of performing a block generation function, a block modification function, a block inquiry function, and the like) in a specific table within a database (not shown) inside the connected main server 400. Here, the connection between the cloud pocket 300 and the main server 400 is
15 configured as bidirectional communication, and they communicate using a preset TCP/IP protocol. In this case, the corresponding TCP/IP protocol is custom-developed, and protocol information thereof is private (or non-disclosed).

 Furthermore, the cloud pocket 300 receives a specific vault ID related to the specific table in which the corresponding result of performing the function is stored,
20 provided from the main server 400.

 In addition, the cloud pocket 300 connects to (or communicates with/accesses) the corresponding specific gate server 200 at a preset interval, and then registers (or stores) the specific vault ID in the corresponding table (or vault) of the specific gate

server 200. Here, the cloud pocket 300 may cumulatively store the specific vault ID in the table (or vault) among a plurality of tables (or vaults) within the specific gate server 200 where data (including, for example, an authentication block, an authentication key block, a request statement, and the like) was previously registered
5 in relation to the user terminal 100, or may initialize the corresponding table and store the specific vault ID. In this case, the cloud pocket 300 may respectively store the authentication block, the authentication key block, the request statement, the specific vault ID, and the like in different tables among the plurality of tables of the specific gate server 200.

10 Furthermore, the main server 400 interworks with each of the remaining main servers 400, excluding the corresponding one main server 400 among the plurality of main servers 400, to respectively store (or register/update) the functional result (including, for example, a block generation result, a block modification result, a block inquiry result, and the like) in databases respectively configured in the remaining main
15 servers 400.

In addition, the user terminal 100 checks (or displays/outputs) a specific vault ID registered (or stored) in the corresponding table (or vault) of the specific gate server 200.

20 Additionally, the user terminal 100 uses the specific vault ID when it intends to perform a modification function, an inquiry function, or the like regarding information (or data) related to the corresponding case in the future.

As an example, the third cloud pocket 300 connects to a fourth main server among the plurality of main servers 400 and transmits the block inquiry result to the connected fourth main server.

5 Then, the fourth main server receives the block inquiry result transmitted from the third cloud pocket.

The fourth main server stores the received block inquiry result in a 44th table among a plurality of tables managed by the fourth main server, and transmits a 44th vault ID, which corresponds to the 44th table where the block inquiry result is stored, to the third cloud pocket.

10 Furthermore, the fourth main server interworks with all remaining main servers 400, excluding the fourth main server itself, to share the block inquiry result so that each of the remaining main servers 400 respectively stores the block inquiry result.

15 The third cloud pocket receives the 44th vault ID transmitted from the fourth main server.

After connecting to the second gate server at a preset interval, the third cloud pocket stores the 44th vault ID in a second table among a plurality of tables within the second gate server. In this case, the third cloud pocket updates the 44th vault ID in each of the remaining cloud pockets 300, excluding the third cloud pocket itself among
20 all the cloud pockets 300, by using threads.

In addition, the first user terminal displays the 44th vault ID registered in the second table within the accessed second gate server.

As another example, the third cloud pocket 300 connects to a fourth main server among the plurality of main servers 400 and transmits the result of performing a block modification function to the connected fourth main server.

Furthermore, the fourth main server receives the result of performing the block
5 modification function transmitted from the third cloud pocket.

In addition, the fourth main server stores the received result of performing the block modification function in a 55th table among a plurality of tables managed by the fourth main server, and transmits a 55th vault ID (corresponding to the 55th table where the result is stored) to the third cloud pocket.

10 The fourth main server also interworks with all remaining main servers 400, excluding the fourth main server itself, to share the result of performing the block modification function so that each of the remaining main servers 400 respectively stores the result.

The third cloud pocket receives the 55th vault ID transmitted from the fourth
15 main server.

After connecting to the 23rd gate server at a preset interval, the third cloud pocket stores the 55th vault ID in a 24th table among a plurality of tables within the 23rd gate server. In this case, the third cloud pocket updates the 55th vault ID in each of the remaining cloud pockets 300, excluding the third cloud pocket itself among all
20 the cloud pockets 300, by using threads.

The second user terminal then displays the 55th vault ID registered in the 24th table within the accessed 23rd gate server (S590).

As described above, according to an embodiment of the present invention, the user terminal 100 encrypts original data using an authentication key to generate an encrypted block, and registers request information (including the encrypted block and a timestamp) into the gate server 200. The cloud pocket 300 then accesses the gate server 200 to confirm the request information, decrypts the encrypted block included in the request information to restore the original data, and stores the original data as a new block in the blockchain or modifies existing data with the original data.

By configuring the blockchain in cloud pockets 300 connected to a network rather than on a personal PC or device, no information is stored on the personal device.

In a state where only an encrypted block protocol connecting the user and the cloud pocket 300 exists, numerous cloud pockets are connected to each other in a P2P (Peer-to-Peer) manner on the network. This enables monetary functions through real-time processing, allowing for real-time modification, transmission, and generation, as well as enabling control over the money supply and tracking the flow of virtual currency.

Furthermore, as described above, according to an embodiment of the present invention, after a user terminal logs into a web server using an ID and a password, the user terminal receives an authentication block provided by the web server upon request and registers the received authentication block in a gate server. Subsequently, a cloud pocket confirms authentication by checking whether the authentication block registered in the gate server by the user terminal matches an authentication block registered in the gate server by the web server. If the authentication is successful, the cloud pocket generates an authentication key for which a validity period is set and registers the generated authentication key in the gate server. Thereafter, the user

terminal interworks with the gate server and the cloud pocket using the authentication key registered in the gate server to register original data in a cloud blockchain in a block format. Accordingly, the user terminal can efficiently perform block generation, modification, and the like using the authentication key for which the
5 validity period is set.

In the detailed description of the present invention above, only specific embodiments thereof have been described. However, it should be understood that the present invention is not limited to the specific forms described in the detailed description, but rather includes all modifications, equivalents, and alternatives falling
10 within the spirit and scope of the present invention defined by the appended claims.

That is, the present invention is not limited to the above-described specific exemplary embodiments. Those skilled in the art may variously modify the present invention without departing from the gist of the present invention claimed by the appended claims and the modifications are within the scope of the claims.

WHAT IS CLAIMED IS:

1. A cloud blockchain system, comprising:

a gate server;

5 a user terminal; and

a cloud pocket;

wherein the gate server is configured to mediate between the user terminal and the cloud pocket;

wherein the user terminal is configured to access a gate server among a plurality of gate servers to be issued an authentication key from the cloud pocket, register an authentication block provided by a web server into one of a plurality of tables within the gate server, and receive an authentication key block corresponding to the registered authentication block;

wherein the cloud pocket is configured to access the gate server at a preset interval and check whether data is registered in the plurality of tables, verify registered data when at least one of the plurality of tables contains data, if the verified data is the authentication block, compare the authentication block registered by the web server on the gate server with the authentication block registered by the user terminal on the gate server for consistency by interworking with the gate server, if the comparison results in a match, verify an identifier (ID) and a password included in the authentication block, and generate the authentication key based on the verified ID, the password, a current time, and a preset expiration period, generate a user account encryption/decryption key comprising an inquiry account key, a modification account

key, and an computation key, and register the authentication key block, including the generated authentication key and the generated user account encryption/decryption key, into a table of the gate server;

wherein the user terminal is further configured to log into the web server using
5 the ID and the password;

wherein the web server is configured to generate the authentication block comprising the ID and the password based on an authentication block generation request from the user terminal, transmit the generated authentication block to the user terminal, and register the generated authentication block on the gate server
10 interworking with the web server;

wherein the user terminal is further configured to receive the authentication block transmitted from the web server in response to the authentication block generation request; and

wherein a connection between the cloud pocket and the gate server is a
15 unidirectional communication in which only the cloud pocket accesses the gate server.

2. A method for controlling a cloud blockchain system, comprising:

accessing, by a user terminal, a gate server among a plurality of gate servers to be issued an authentication key from a cloud pocket;

20 registering, by the user terminal, an authentication block provided by a web server into one of a plurality of tables within the gate server;

accessing, by the cloud pocket, the gate server at a preset interval to check whether data is registered in the plurality of tables;

verifying, by the cloud pocket, registered data when it is determined that at least one of the plurality of tables is in a state where data is registered;

comparing, by the cloud pocket, the authentication block registered by the web server on the gate server with the authentication block registered by the user terminal
5 on the gate server for consistency by interworking with the gate server, when the verified data is the authentication block;

verifying an identifier (ID) and a password included in the authentication block and generating, by the cloud pocket, the authentication key based on the verified ID, the password, a current time, and a preset expiration period, when the
10 authentication block registered by the web server matches the authentication block registered by the user terminal;

generating, by the cloud pocket, a user account encryption/decryption key comprising a inquiry account key of a preset size for interpreting encrypted data, a modification account key of a preset size for encrypting data, and an computation key
15 of a preset size configured to prevent pattern interpretation using an arithmetic coding method;

registering, by the cloud pocket, an authentication key block comprising the generated authentication key and the generated user account encryption/decryption key into the table of the gate server; and

20 confirming, by the user terminal, the authentication key block registered in the table of the gate server;

wherein the user terminal logs into the web server using the ID and the password;

wherein the web server is configured to generate the authentication block comprising the ID and the password based on an authentication block generation request from the user terminal, transmit the generated authentication block to the user terminal, and register the generated authentication block on the gate server
5 interworking with the web server;

wherein the user terminal receives the authentication block transmitted from the web server in response to the authentication block generation request; and

wherein a connection between the cloud pocket and the gate server is a unidirectional communication in which only the cloud pocket accesses the gate server.

10

3. The method of claim 2, further comprising:

generating, by the user terminal, an encrypted block by encrypting original data to be registered in the cloud blockchain managed by the cloud pocket using the user account encryption/decryption key included in the authentication key block, in
15 order to store data in the cloud blockchain or to query data registered in the cloud blockchain;

generating, by the user terminal, a request statement comprising the generated encrypted block and the authentication key included in the authentication key block;

accessing, by the user terminal, the gate server;

20 registering, by the user terminal, the generated request statement into one of the plurality of tables within the gate server;

accessing, by the cloud pocket, the gate server at the preset interval and checking whether data is registered in the plurality of tables within the gate server;

verifying, by the cloud pocket, the registered data when it is determined that at least one of the plurality of tables is in a state where data is registered;

performing, by the cloud pocket, a verification function on the request statement, when the verified data is the request statement; and

5 performing, by the cloud pocket, any one of a block generation function, a block modification function, and a block inquiry function in the cloud blockchain based on the original data decrypted from the request statement, when the verification of the request statement is successfully completed.

10 4. The method of claim 3, wherein the performing of the verification function on the request statement comprises:

verifying, by the cloud pocket, whether a time difference between a current time and a timestamp included in the request statement exceeds a preset reference value to validate the data;

15 registering, by the cloud pocket, information indicating that validity verification of the data has failed into the table of the gate server and terminating a connection with the gate server, when the request statement is determined to be invalid as a result of the validity verification;

20 verifying, by the cloud pocket, whether a header value included in the request statement satisfies a preset specification, when the request statement is determined to be valid as a result of the validity verification; and

generating the original data by decrypting, by the cloud pocket, the encrypted block included in the request statement using the user account encryption/decryption

key generated based on the authentication block, when the header value included in the request statement satisfies the preset specification.

5. The method of claim 3, further comprising:

5 registering, by the cloud pocket, a result of performing a function corresponding to the request statement into the table of the gate server after accessing the gate server at the preset interval; and

 confirming, by the user terminal, the result of performing the function registered in the table of the gate server.

10

6. The method of claim 3, further comprising:

 connecting, by the cloud pocket, to a main server among a plurality of main servers, and storing a result of performing a function corresponding to the request statement in a table within a database of the main server;

15 transmitting, by the main server, a vault ID corresponding to the table to the cloud pocket;

 registering, by the main server, the result of performing the function in respective databases of remaining main servers among the plurality of main servers, excluding the main server, by interworking with each of the remaining main servers;

20 receiving, by the cloud pocket, the vault ID transmitted from the main server;

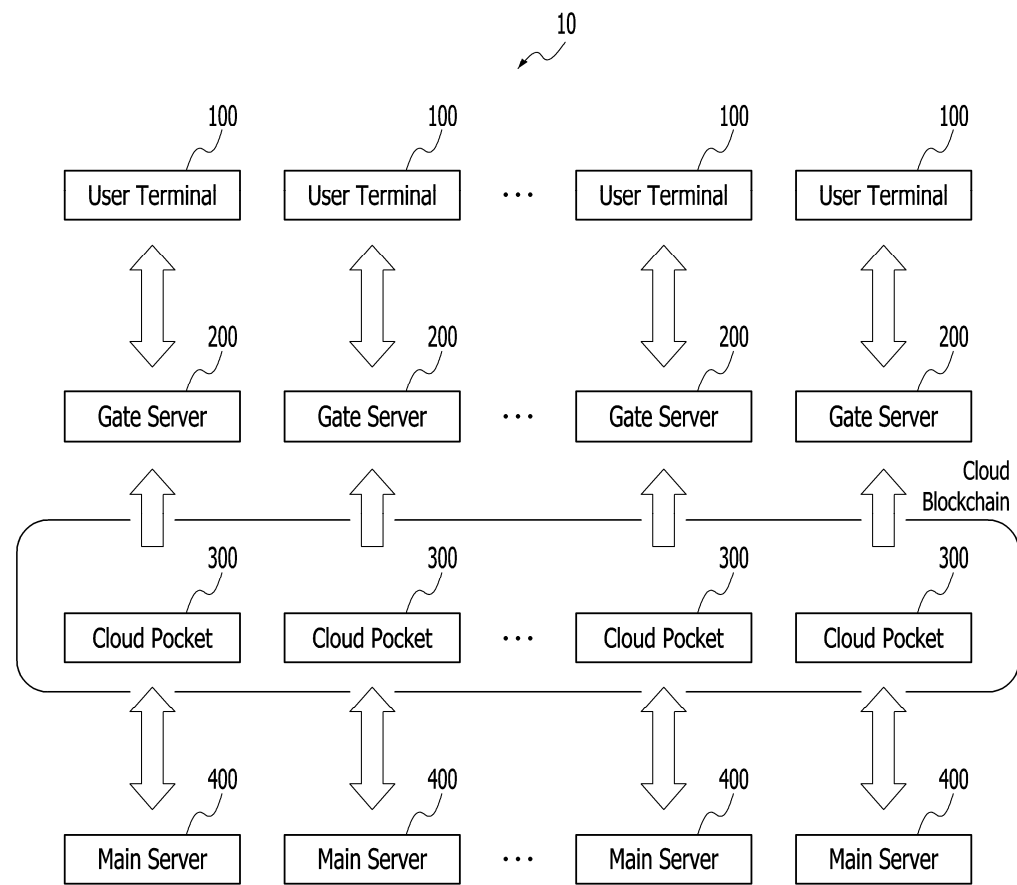
 registering, by the cloud pocket, the vault ID in the table of the gate server after accessing the gate server at the preset interval; and

confirming, by the user terminal, the vault ID registered in the table of the gate server.

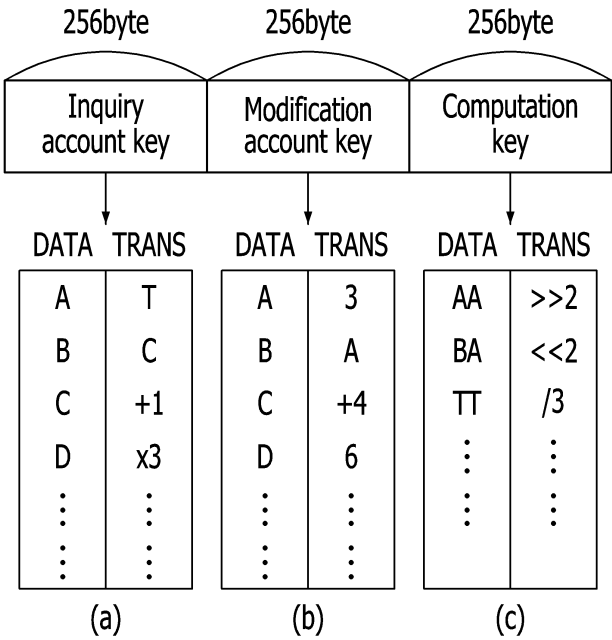
ABSTRACT

The present disclosure discloses a cloud blockchain system and a method for controlling the same. A user terminal encrypts original data using an authentication key to generate an encrypted block, and registers request information including the encrypted block and a timestamp on a gate server. A cloud pocket accesses the gate server to verify the request information, restores the original data by decrypting the encrypted block, and stores the restored data as a new block in a blockchain or modifies existing data. By configuring the blockchain in a network-connected cloud pocket rather than on a local device, no information is stored on the personal device, and only an encrypted block protocol exists between the user and the cloud pocket. A plurality of cloud pockets are interconnected via a P2P network to enable currency functions through real-time processing.

[FIG. 1]



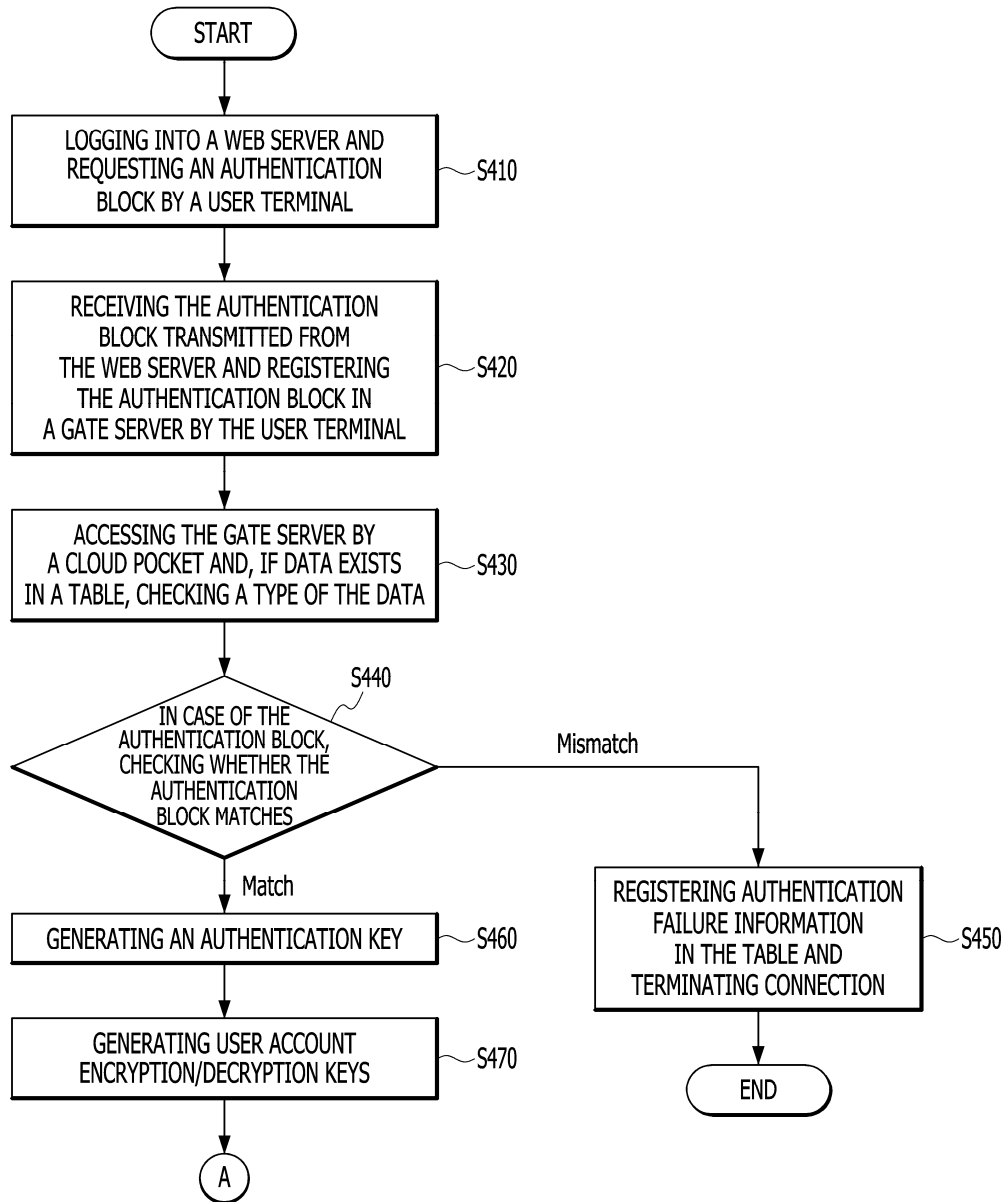
[FIG. 2]



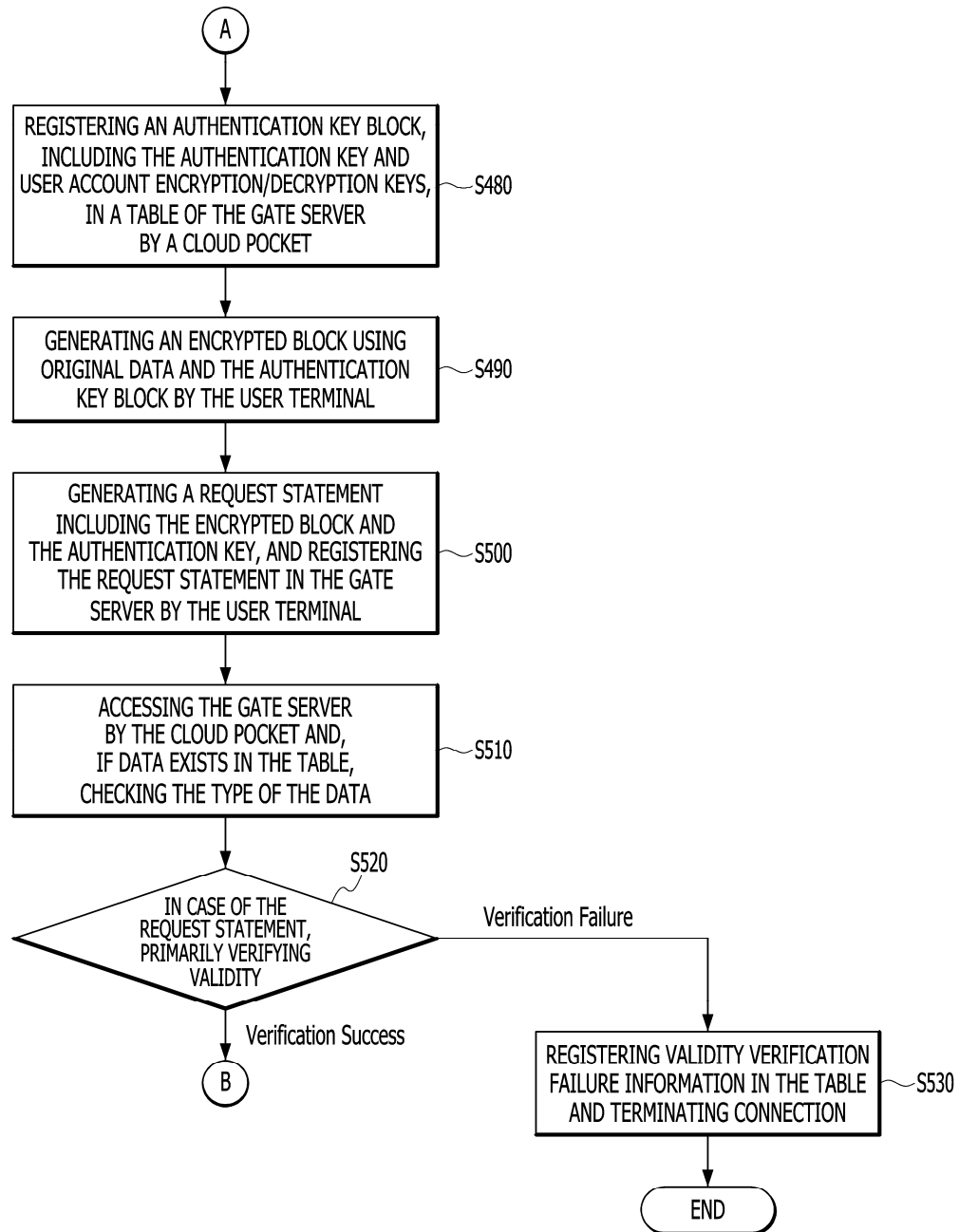
[FIG. 3]

Vault ID	Table
1	member
2	bbs
3	info
4	content
...	...

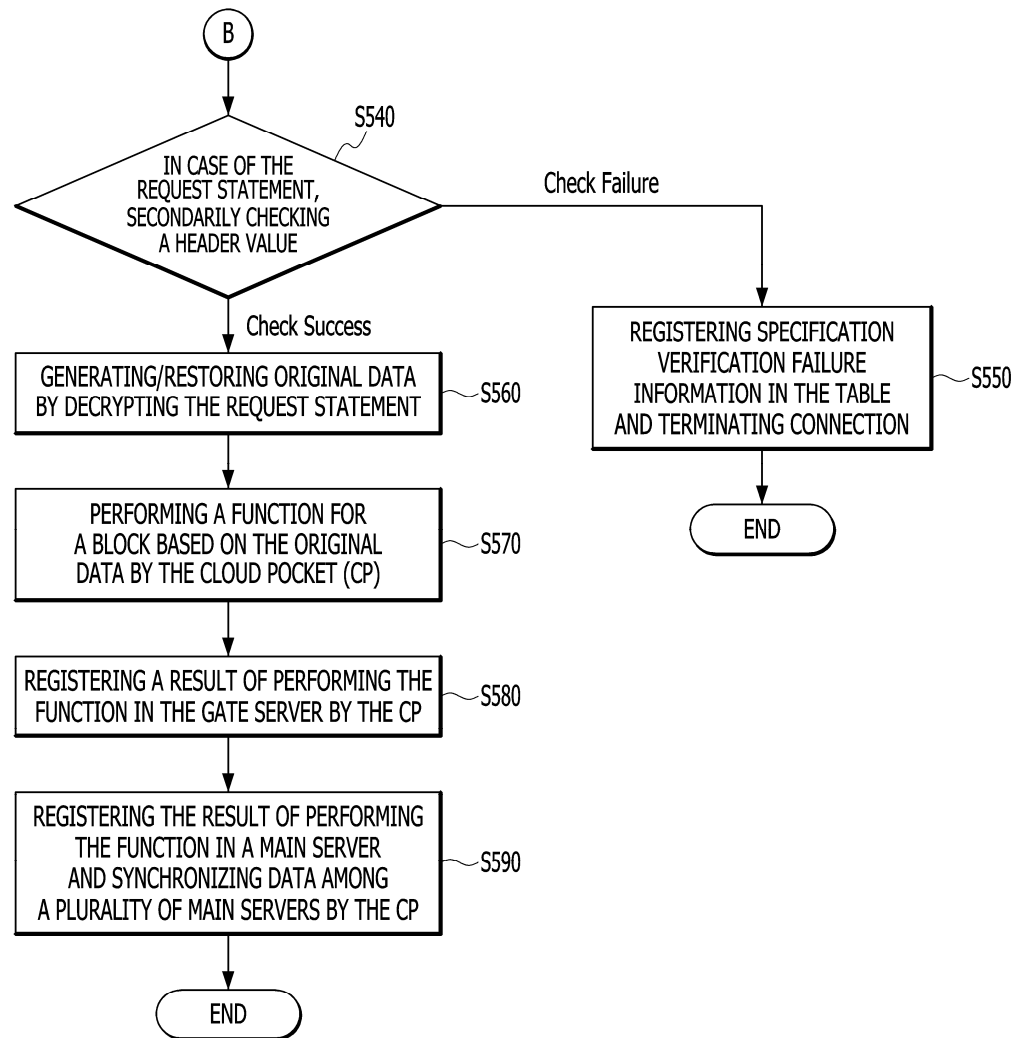
[FIG. 4]



[FIG. 5]



[FIG. 6]



PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of

Docket No: Q318247

Sang Yeob LEE

Appln. No.: Not Yet Assigned

Group Art Unit: Not Yet Assigned

Confirmation No.: Not Yet Assigned

Examiner: Not Yet Assigned

Filed: March 12, 2026

For: CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD

**INFORMATION DISCLOSURE STATEMENT
UNDER 37 C.F.R. §§ 1.97 and 1.98**

MAIL STOP AMENDMENT

Commissioner for Patents

P.O. Box 1450

Alexandria, VA 22313-1450

Sir:

In accordance with the duty of disclosure under 37 C.F.R. § 1.56, Applicant hereby notifies the U.S. Patent and Trademark Office of the documents which are listed on the attached PTO/SB/08 (modified) form and/or listed herein and which the Examiner may deem material to patentability of the claims of the above-identified application.

One copy of each of the listed documents is submitted herewith, except for the following: U.S. patents and/or U.S. patent publications; and co-pending non-provisional U.S. applications filed after June 30, 2003.

The submission of the listed documents is not intended as an admission that any such document constitutes prior art against the claims of the present application. Applicant does not waive any right to take any action that would be appropriate to antedate or otherwise remove any listed document as a competent reference against the claims of the present application.

INFORMATION DISCLOSURE STATEMENT

UNDER 37 C.F.R. §§ 1.97 and 1.98

Appln. No.: Not Yet Assigned

Attorney Docket No.: Q318247

The USPTO is directed and authorized to charge all required fees, except for the Issue Fee and the Publication Fee, to Deposit Account No. 19-4880. Please also credit any overpayments to said Deposit Account.

Respectfully submitted,

/Hyunseok Park/

SUGHRUE MION, PLLC

Telephone: 202.293.7060

Facsimile: 202.293.7860

WASHINGTON OFFICE

23373

CUSTOMER NUMBER

Hyunseok Park

Registration No. 74,833

Date: March 12, 2026

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

Application Number	Not Yet Assigned
Confirmation Number	Not Yet Assigned
Filing Date	March 12, 2026
First Named Inventor	Sang Yeob LEE
Art Unit	Not Yet Assigned
Examiner Name	Not Yet Assigned
Attorney Docket Number	Q318247

U.S. PATENTS

Examiner Initials*	Cite No	Patent Number	Kind Code ¹	Issue Date	Name of Inventor of cited Document	Pages, Columns, Lines where Relevant Passages or Relevant Figures Appear

U.S. PATENT APPLICATION PUBLICATIONS

Examiner Initials*	Cite No	Publication Number	Kind Code ¹	Publication Date	Name of Inventor or Applicant of cited Document	Pages, Columns, Lines where Relevant Passages or Relevant Figures Appear

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No	Foreign Document Number ³	Country Code ²	Kind Code ⁴	Publication Date	Name of Patentee or Applicant of cited Document	Pages, Columns, Lines where Relevant Passages or Relevant Figures Appear	T ⁵
	1.	10-2015-0107796	KR	A	2015-09-23	INTEL CORP		English Abstract of US 2014/281531, Cited in KR Office Action and Notice of Allowance
	2.	10-1284353	KR	B1	2013-07-08	PARK SUN YOUNG		English Abstract, Cited in KR Office Action and Notice of Allowance
	3.	10-1727525	KR	B1	2017-04-17	SCALECHAIN INC		English Abstract, Cited in Specification on page 2
	4.	10-2062688	KR	A	2020-02-11	SAMSUNG ELECTRONICS CO LTD		English Abstract of US 2013/336486, Cited in KR Office Action and Notice of Allowance
	5.	10-2307863	KR	B1	2021-10-01	SK Planet Co., Ltd.		English Abstract, Cited in KR Office Action and Notice of Allowance
	6.	10-2461050	KR	B1	2022-10-28	LEE SOO HEE		English Abstract, Cited in KR Office Action and Notice of Allowance

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

Application Number	Not Yet Assigned
Confirmation Number	Not Yet Assigned
Filing Date	March 12, 2026
First Named Inventor	Sang Yeob LEE
Art Unit	Not Yet Assigned
Examiner Name	Not Yet Assigned
Attorney Docket Number	Q318247

NON-PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ⁵
	7.	Korean Office Action for 10-2025-0034578, dated June 5, 2025	English Machine Translation
	8.	Korean Notice of Allowance for 10-2025-0034578, dated September 25, 2025	English Machine Translation

EXAMINER SIGNATURE

Examiner Signature		Date Considered	
*EXAMINER: Initial if reference considered and whether or not citation is in conformance with MPEP 609. Draw line through a citation if not in conformance and not considered. Include copy of this form with next communication to applicant.			
¹ See Kind Codes of USPTO Patent Documents at www.USPTO.GOV or MPEP 901.04. ² Enter office that issued the document, by the two-letter code (WIPO Standard ST.3). ³ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁴ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁵ Applicant is to place a check mark here if English language translation is attached.			

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

Application Number	Not Yet Assigned
Confirmation Number	Not Yet Assigned
Filing Date	March 12, 2026
First Named Inventor	Sang Yeob LEE
Art Unit	Not Yet Assigned
Examiner Name	Not Yet Assigned
Attorney Docket Number	Q318247

TIMING STATEMENT

Please see 37 CFR 1.97(e) to make the appropriate selection(s):

- ☐ That each item of information contained in the information disclosure statement was first cited in any communication from a foreign patent office in a counterpart foreign application not more than three months prior to the filing of the information disclosure statement. See 37 CFR 1.97(e)(1).

OR

- ☐ That no item of information contained in the information disclosure statement was cited in a communication from a foreign patent office in a counterpart foreign application, and, to the knowledge of the person signing the certification after making reasonable inquiry, no item of information contained in the information disclosure statement was known to any individual designated in 37 CFR 1.56(c) more than three months prior to the filing of the information disclosure statement. See 37 CFR 1.97(e)(2).
- ☐ See attached timing statement under 37 CFR 1.97(e).
- ☒ A timing statement under 37 CFR 1.97(e) is not submitted herewith.

TIMING FEE

- ☐ The fee set forth in 37 CFR 1.17(p) has been submitted herewith.

COPIES

- ☐ An identification of an earlier application pursuant to 37 CFR 1.98(d)(1) is attached.

SIZE FEE ASSERTION

Please see 37 CFR 1.17(v) and the IDS Auto-Load Instructions for completing this form to make the appropriate selection of an assertion under 37 CFR 1.98. For the information disclosure statement (IDS) submitted herewith, the applicant or patent owner certifies the following with respect to the cumulative number of applicant-provided or patent-owner provided items of information submitted to date including those in the accompanying IDS (select only one):

- ☒ No IDS size fee is required under 37 CFR 1.17(v) at this time.
- ☐ The IDS is accompanied by the IDS size fee under 37 CFR 1.17(v)(1).
- ☐ The IDS is accompanied by the IDS size fee under 37 CFR 1.17(v)(2).
- ☐ The IDS is accompanied by the IDS size fee under 37 CFR 1.17(v)(3).
- ☐ Previous Payment Amount _____ Date Submitted _____

The fee set forth in 37 CFR 1.17(v) has been submitted herewith.

DEPOSIT ACCOUNT

- ☒ The USPTO is directed and authorized to charge all required fees, except for the Issue Fee and the Publication Fee, to Deposit Account No. 19-4880. Please also credit any overpayments to said Deposit Account.

WASHINGTON OFFICE

23373

CUSTOMER NUMBER

SIGNATURE

A signature of the applicant or representative is required in accordance with 37 CFR 1.33 and 11.18. Please see 37 CFR 1.4(d) for the form of the signature.

Signature	/Hyunseok Park/	Date (YYYY-MM-DD)	2026-03-12
Name/Print	Hyunseok Park	Registration Number	74,833



UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC PAYMENT RECEIPT

APPLICATION #
19/564,935

RECEIPT DATE / TIME
03/12/2026 03:34:54 PM Z ET

ATTORNEY DOCKET #
Q318247

Title of Invention

CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD

Application Information

APPLICATION TYPE	Utility - Nonprovisional Application under 35 USC 111(a)	PATENT #	-
CONFIRMATION #	3313	FILED BY	Hiroko Derello
PATENT CENTER #	74831260	AUTHORIZED BY	Hyunseok Park
CUSTOMER #	23373	FILING DATE	-
CORRESPONDENCE ADDRESS	-	FIRST NAMED INVENTOR	Sang Yeob LEE

Payment Information

PAYMENT METHOD CARD / 2251	PAYMENT TRANSACTION ID E20263BF35257906	PAYMENT AUTHORIZED BY Hiroko Derello
PRE-AUTHORIZED ACCOUNT 194880	PRE-AUTHORIZED CATEGORY 37 CFR 1.17 (Patent application and reexamination processing fees)	

FEE CODE	DESCRIPTION	ITEM PRICE(\$)	QUANTITY	ITEM TOTAL(\$)
3011	BASIC FILING FEE - UTILITY (PAPER FILING ALSO REQUIRES NON-ELECTRONIC FILING FEE UNDER 1.16(T))	70.00	1	70.00
3311	EXAMINATION OF ORIGINAL PATENT APPLICATION	176.00	1	176.00
3111	UTILITY PATENT APPL. SEARCH FEE	154.00	1	154.00
TOTAL AMOUNT:				\$400.00

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized

by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.



UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #
19/564,935

RECEIPT DATE / TIME
03/12/2026 04:06:35 PM Z ET

ATTORNEY DOCKET #
Q318247

Title of Invention

CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD

Application Information

APPLICATION TYPE	Utility - Nonprovisional Application under 35 USC 111(a)	PATENT #	-
CONFIRMATION #	3313	FILED BY	Hiroko Derello
PATENT CENTER #	74834132	FILING DATE	-
CUSTOMER #	23373	FIRST NAMED INVENTOR	Sang Yeob LEE
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	Hyunseok Park

Documents

TOTAL DOCUMENTS: 2

DOCUMENT	PAGES	DESCRIPTION	SIZE (KB)
Q318247PPHRequestasfiled.pdf	2	Petition to make Special under Patent Prosecution Highway	149 KB
Q318247PPHSupportingDocuments.pdf	20	Petition to make Special under Patent Prosecution Highway	307 KB

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
Q318247PPHRequestasfiled.pdf	810D2F60661C33790858A316F598FA2FB90EED8BC6510151E2EB41E9F483675DA1209EAFAB9FD28DE3D44A301E6E9FDB4AC8899400B55A92C0FAEFD21C747258
Q318247PPHSupportingDocum	F0D4A90C422DADDD957DEF1E64B2F62DDD95158C6D2B407

ents.pdf

3C2DA4944E6EF647DCBFDFEC77801D17CAB14B315EB54D41
EBDADD07084C96E5EAFF35645D26EC43F

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

**REQUEST FOR PARTICIPATION IN THE GLOBAL/IP5
PATENT PROSECUTION HIGHWAY (PPH) PILOT PROGRAM IN THE USPTO**

Application No.:	19/564,935	Confirmation No.:	3313
Filing Date:	March 12, 2026	Attorney Docket No.:	Q318247
First Named Inventor:	Sang Yeob LEE		
Title of Invention:	CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD		

THIS REQUEST FOR PARTICIPATION IN THE PPH PILOT PROGRAM ALONG WITH THE REQUIRED DOCUMENTS MUST BE SUBMITTED VIA THE USPTO PATENT ELECTRONIC FILING SYSTEM. INFORMATION REGARDING THE PATENT ELECTRONIC FILING SYSTEM IS AVAILABLE AT <https://www.uspto.gov/ebc>

APPLICANT HEREBY REQUESTS PARTICIPATION IN THE PATENT PROSECUTION HIGHWAY (PPH) PILOT PROGRAM AND PETITIONS TO MAKE THE ABOVE-IDENTIFIED APPLICATION SPECIAL UNDER THE PPH PILOT PROGRAM.

***For national stage applications under 35 U.S.C. 371 only:** This is an express request to begin national stage processing under 35 U.S.C. 371(f). **Note:** National stage processing will not commence prior to 30 months from the priority date absent an express request to begin national stage processing under 35 U.S.C. 371(f) and fulfillment of the requirements under 35 U.S.C. 371(c)(1), (2), and (4) for payment of the basic national fee, copy of the International Application and English translation thereof (if required), and the oath or declaration of the inventor(s).*

Office of earlier examination (OEE): Korea (Korean Intellectual Property Office)

OEE application number: 10-2025-0034578

Both the OEE application and the above-identified U.S. application have the following earliest date (filing or priority date): March 18, 2025

Type of OEE work product relied upon: Notice of refusal to grant a patent (where such notice indicates allowable claims), Decision to grant a patent

Mailing date of OEE work product: June 5, 2025, September 25, 2025

Supporting Documents

1. OEE Work Product and Translation

A copy of the OEE work product and translation if not already in English:

- ☒ Attached ☐ Previously Submitted ☐ Not required because the decision to grant a patent was the first office action
- ☐ Applicant requests the USPTO to attempt to obtain the OEE work product from the Dossier Access System or PATENTSCOPE

NOTE: If the applicant requests the USPTO to obtain the OEE work product electronically and such attempt is unsuccessful, the applicant will be required to supply the document. Accordingly, to avoid dismissal of the initial PPH request and potential denial of participation in the PPH program, the applicant should verify that the OEE work product is actually available via the Dossier Access System or PATENTSCOPE before requesting retrieval. If the applicant is unable to verify availability, then the applicant should submit the document with the PPH request.

2. References Cited in OEE Work Product

An information disclosure statement (IDS) listing the references cited in the OEE work product and document copies (except U.S. patents and U.S. published patent applications):

- ☐ Attached ☒ Previously Submitted ☐ Not required because no references were cited in the OEE work product

REQUEST FOR PARTICIPATION IN THE GLOBAL/IP5 PPH PILOT PROGRAM IN THE USPTO

(continued)

Application No.:	19/564,935	First Named Inventor:	Sang Yeob LEE
------------------	------------	-----------------------	---------------

3. Claims Correspondence Certification Statement

All of the claims in this application sufficiently correspond to the patentable/allowable claims in the OEE application.

4. Claims Correspondence Table

Claims in U.S. Application	Patentable/Allowable Claims in OEE Application	Explanation Regarding the Correspondence
1	1	Substantially Same
2	3	Substantially Same
3	4	Substantially Same
4	5	Substantially Same
5	6	Substantially Same
6	7	Substantially Same

5. Payment of Fees:

- ☒ The USPTO is directed and authorized to charge all required fees, except for the Issue Fee and the Publication Fee to Deposit Account No. 19-4880. Please also credit any overpayments to said Deposit Account.
- ☐ Credit Card.

SEND ALL CORRESPONDENCE TO:
The address associated with Customer Number:

WASHINGTON OFFICE

23373

CUSTOMER NUMBER

Signature /Hyunseok Park/

Date March 12, 2026

Name (print or type) Hyunseok Park

Registration Number 74,833



UNITED STATES PATENT AND TRADEMARK OFFICE

UNDER SECRETARY OF COMMERCE FOR INTELLECTUAL PROPERTY AND
DIRECTOR OF THE UNITED STATES PATENT AND TRADEMARK OFFICE

MARCH 16, 2026

PTAS

SUGHRUE MION PLLC
2000 PENNSYLVANIA AVENUE, N.W.
SUITE 9000
WASHINGTON, DC 20006

509715892

UNITED STATES PATENT AND TRADEMARK OFFICE NOTICE OF RECORDATION OF ASSIGNMENT DOCUMENT

THE ENCLOSED DOCUMENT HAS BEEN RECORDED BY THE ASSIGNMENT RECORDATION BRANCH OF THE U.S. PATENT AND TRADEMARK OFFICE. A COMPLETE COPY IS AVAILABLE AT THE ASSIGNMENT SEARCH ROOM ON THE REEL AND FRAME NUMBER REFERENCED BELOW.

PLEASE REVIEW ALL INFORMATION CONTAINED ON THIS NOTICE. THE INFORMATION CONTAINED ON THIS RECORDATION NOTICE REFLECTS THE DATA PRESENT IN THE PATENT AND TRADEMARK ASSIGNMENT SYSTEM. IF YOU SHOULD FIND ANY ERRORS OR HAVE QUESTIONS CONCERNING THIS NOTICE, YOU MAY CONTACT THE ASSIGNMENT RECORDATION BRANCH AT 571-272-3350. PLEASE SEND REQUEST FOR CORRECTION TO: U.S. PATENT AND TRADEMARK OFFICE, MAIL STOP: ASSIGNMENT RECORDATION BRANCH, P.O. BOX 1450, ALEXANDRIA, VA 22313.

RECORDATION DATE: 03/12/2026

REEL/FRAME: 74061/0615
NUMBER OF PAGES: 4

BRIEF: ASSIGNMENT OF ASSIGNOR'S INTEREST

DOCKET NUMBER: Q318247

ASSIGNOR:
LEE, SANG YEGB

DOC DATE: 02/13/2026

ASSIGNEE:

KIM, BO MI
34-1, 516-RO 97BEON-GIL
SEOGWIPO-SI, JEJU-DO, KOREA, REPUBLIC
OF, 63604

APPLICATION NUMBER: 19564935

FILING DATE:

PATENT NUMBER:

ISSUE DATE:

TITLE: CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD

ASSIGNMENT RECORDATION BRANCH
PUBLIC RECORDS DIVISION

CERTIFICATION OF MICRO ENTITY STATUS
(GROSS INCOME BASIS)

Application Number:	Not Yet Assigned	Confirmation No.:	Not Yet Assigned
First Named Inventor:	Sang Yeob LEE	Patent Number (if applicable):	
Filing Date:	March 12, 2026	Attorney Docket No.:	Q318247
Title of Invention:	CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD		

The applicant hereby certifies the following -

- (1) **SMALL ENTITY REQUIREMENT** – The applicant qualifies as a small entity as defined in 37 CFR 1.27.
- (2) **APPLICATION FILING LIMIT** – Neither the applicant nor the inventor nor a joint inventor has been named as the inventor or a joint inventor on more than four previously filed U.S. patent applications, excluding provisional applications and international applications under the Patent Cooperation Treaty (PCT) for which the basic national fee under 37 CFR 1.492(a) was not paid, and also excluding patent applications for which the applicant has assigned all ownership rights, or is obligated to assign all ownership rights, as a result of the applicant's previous employment.
- (3) **GROSS INCOME LIMIT ON APPLICANTS AND INVENTORS** – Neither the applicant nor the inventor nor a joint inventor, in the calendar year preceding the calendar year in which the applicable fee is being paid, had a gross income, as defined in section 61(a) of the Internal Revenue Code of 1986 (26 U.S.C. 61(a)), exceeding the "Maximum Qualifying Gross Income" reported on the USPTO Web site at http://www.uspto.gov/patents/law/micro_entity.jsp which is equal to three times the median household income for that preceding calendar year, as most recently reported by the Bureau of the Census.
- (4) **GROSS INCOME LIMIT ON PARTIES WITH AN "OWNERSHIP INTEREST"** – Neither the applicant nor the inventor nor a joint inventor has assigned, granted, or conveyed, nor is under an obligation by contract or law to assign, grant, or convey, a license or other ownership interest in the application concerned to an entity that, in the calendar year preceding the calendar year in which the applicable fee is being paid, had a gross income, as defined in section 61(a) of the Internal Revenue Code of 1986, exceeding the "Maximum Qualifying Gross Income" reported on the USPTO Web site at http://www.uspto.gov/patents/law/micro_entity.jsp which is equal to three times the median household income for that preceding calendar year, as most recently reported by the Bureau of the Census.

SEND ALL CORRESPONDENCE TO:
The address associated with Customer Number:

WASHINGTON OFFICE

23373

CUSTOMER NUMBER

SIGNATURE by an authorized party set forth in 37 CFR 1.33(b)

Signature	/Hyunseok Park/				
Name	Hyunseok Park				
Date	March 12, 2026	Telephone	202.293.7060	Registration No.	74,833
☐	There is more than one inventor and I am one of the inventors who are jointly identified as the applicant. The required additional certification form(s) signed by the other joint inventor(s) are included with this form.				

TRANSMITTAL FOR POWER OF ATTORNEY TO ONE OR MORE REGISTERED PRACTITIONERS

NOTE: This form is to be submitted with the Power of Attorney by Applicant form (PTO/AIA/82B) to identify the application to which the Power of Attorney is directed, in accordance with 37 CFR 1.5, unless the application number and filing date are identified in the Power of Attorney by Applicant form. If neither form PTO/AIA/82A nor form PTO/AIA82B identifies the application to which the Power of Attorney is directed, the Power of Attorney will not be recognized in the application.

Application Number	Not Yet Assigned
Filing Date	March 12, 2026
First Named Inventor	Sang Yeob LEE
Title	CLOUD BLOCKCHAIN SYSTEM AND ITS CONTROL METHOD
Art Unit	Not Yet Assigned
Examiner Name	Not Yet Assigned
Attorney Docket Number	Q318247

SIGNATURE of Applicant or Patent Practitioner

Signature	/Hyunseok Park/	Date (Optional)	
Name	Hyunseok Park	Registration Number	74,833

Title (if Applicant is a juristic entity)	
---	--

Applicant Name (if Applicant is a juristic entity)	
--	--

NOTE: This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4(d) for signature requirements and certifications. If more than one applicant, use multiple forms.

☒ *Total of 1 forms are submitted.

POWER OF ATTORNEY BY APPLICANT

I hereby revoke all previous powers of attorney given in the application identified in the attached transmittal letter.

I hereby appoint:

Patent Practitioner(s) associated with the following Customer Number:

WASHINGTON OFFICE

23373

CUSTOMER NUMBER

as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the application referenced in the attached transmittal letter (Form PTO/AIA/82A).

Please recognize or change the correspondence address for the application identified in the attached transmittal letter to:

The address associated with the above-mentioned Customer Number.

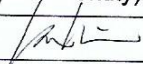
I am the Applicant (if the Applicant is a juristic entity, list the Applicant name in the box):

Bo Mi KIM

- ☐ Inventor or Joint Inventor (title not required below)
- ☐ Legal Representative of a Deceased or Legally Incapacitated Inventor (title not required below)
- ☒ Assignee or Person to Whom the Inventor is Under an Obligation to Assign (provide signer's title if applicant is a juristic entity)
- ☐ Person Who Otherwise Shows Sufficient Proprietary Interest (e.g., a Petition under 37 CFR 1.46(b)(2) was granted in the application or is concurrently being filed with this document)(provide signer's title if applicant is a juristic entity)

SIGNATURE of Applicant for Patent

The undersigned (whose title is supplied below) is authorized to act on behalf of the applicant (e.g., where the applicant is a juristic entity).

Signature		Date (Optional)	February 13, 2026
Name	Bo Mi KIM		
Title			

NOTE: Signature - This form must be signed by the applicant in accordance with 37 CFR 1.33. See 37 CFR 1.4 for signature requirements and certifications. If more than one applicant, use multiple forms.

☒ Total of 1 forms are submitted.