Digital Asset Management with Distributed Permission over Blockchain and Attribute-based Access Control

Yan Zhu, Yao Qin, Zhiyuan Zhou, Xiaoxu Song, Guowei Liu, William Cheng-Chung Chu

1School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China
2Beijing Municipal Commission of Economy and Informatization, Beijing 100101, China
3Department of Computer Science, Tunghai University Taichung, Taiwan
*Email: zhuyan@ustb.edu.cn

Abstract—Digital asset management (DAM) has increasing benefits in booming global Internet economy, but it is still a great challenge for providing an effective way to manage, store, ingest, organize and retrieve digital asset. To do it, we present a new digital asset management platform, called DAM-Chain, with Transaction-based Access Control (TBAC) which integrates the distribution ABAC model and the blockchain technology. In this platform, the ABAC provides flexible and diverse authorization mechanisms for digital asset escrowed into blockchain while the blockchain’s transactions serve as verifiable and traceable medium of access request procedure. We also present four types of transactions to describe the TBAC access control procedure, and provide the algorithms of these transactions corresponding to subject registration, object escrowing and publication, access request and grant. By maximizing the strengths of both ABAC and blockchain, this platform can support flexible and diverse permission management, as well as verifiable and transparent access authorization process in an open decentralized environment.

Index Terms—digital asset management; blockchain; access control; transaction; attribute-based

I. INTRODUCTION

Digital asset management (DAM) provides a way to organise companies valuable files, called digital assets, in a way that makes them quick-find and easy to access. In an information economy, everyone can gain a huge advantage over their competitors by handling their digital assets more effectively than others. We all process digital assets, whether as documents, audible content, motion picture, and other relevant digital data that are currently in circulation, or files sent to us via email, all of which bring us wealth. Therefore, effective digital asset management has taken on increased importance.

A DAM system represents an intertwined structure incorporating both software and hardware and/or other services in order to manage, store, ingest, organize and retrieve digital assets. Moreover, it provides the unbroken maintenance of the ownership of a digitized object while permitting access to those who have obtained rights to that access. The digital asset management offers many advantages and benefits, which consist of

1) the ability to dynamically distribute assets to internal and external teams;
2) a place to quickly find and retrieve assets;
3) the ability for the owner to control who else may view, use, or modify the assets; and
4) a mechanism to keep track of assets’ history, so as to reuse them for maximizing their value.

DAM has increasing benefits in booming global Internet economy, but it is still a great challenge for constructing an effective DAM. Fortunately, blockchain technology may be one of the effective ways to solve this problem. Exactly, blockchain is a digital decentralized ledger that keeps a record of all transactions that take place across a peer-to-peer network. The features of blockchain, including decentralization, tamper-proof, and traceability, stem from its cryptographically secure data structure [1] that takes a number of records and puts them in a block (like collating them on to a sheet). As shown in Fig. 1, each block is then chained to the next block, using a cryptographic hash. In addition, the Merkle hash tree, as a type of binary tree, is introduced to guarantee that the transaction details are validated, and thus cannot be altered later on. In virtue of decentralized nature and above-mentioned properties, the blockchain allow mutually distrustful parties to transact securely without trusted third parties.

Blockchain technology should be a preeminent solution for DAM because it protects the ownership of digital assets and protects the information from being misused. Firstly, it is not easy to modify digital assets stored on blockchain because the assets are stored with a hash. In fact, it is distributed to a large number of validators which makes it extremely hard to tamper any data in blockchain. Secondly, the validity of a
DAM transaction is determined by consensus of the various computers or nodes in the network instead of having a central trusted or official record. Moreover, blockchain can be used to verify the transfer of ownership of digital assets or even just the right to access and use such assets. In summary, digital asset management could leverage security properties of blockchains, which include:

- Impossibility of counterfeit,
- Immutability,
- Disintermediation and ease of transfer, and
- Transparency and ease of auditing.

Therefore, there is no denying in the fact that Blockchain has immense potential to become an important component of any DAM strategy.

Although blockchain provides some unique properties for distributing and managing digital asset, it is still unable to meet the requirements of flexibility, diversity, and dynamicity for permission management. For example, if requesters want to get access to a certain digital asset, they must have the permission of whoever manages it and provide some declarations (or at least hints about how to transact on the required digital asset). More concretely, in the healthcare system the patient’s medical records must follow strict access constraints to prevent the leakage of sensitive information. Or, the distribution of copyrighted media must be consistent with copyright requirements, such as, some movies can only be played in a specific region or a certain device. To overcome this problem, it is a direct and effective method to introduce access control technology into DAM.

In the last two years, several new researches have addressed the problem of integrating access control and blockchain technology. For example, Ouaddah et al. proposed a framework, called FairAccess, for access control in Internet-of-Things (IoT) on the blockchain [2]. This framework used organization-based access control (OrBAC) to enable users to own and control their data. However, the granularity of OrBAC is not enough to realize digital asset management considering that its organization structure may remain relatively fixed over time. Xu et al. also proposed a distributed ledger based access control (DL-BAC) scheme for Web applications [3]. This scheme adapt access control list (ACL) to realize a lightweight decision-making and granting access. For the future work, they plan to extend the privacy enhanced DL-BAC to support more complex access policies including role-based access control (RBAC) and so on. To address the same problem, Azaria et al. used the blockchain to implement a decentralized medical record access and permission management [4] with authentication, confidentiality and accountability. Xia et al. also proposed a blockchain-based data sharing framework [5] which allows access to only invited, and hence verified users. As a result of this design, further accountability is guaranteed as all users are already known and a log of their actions is kept by the blockchain. In general, these researches have enlightening significance for our research.

To implement a flexible, diverse and dynamic access control in DAM, in this paper we intent to introduce attribute-based access control (ABAC) [6], [7] into the blockchain. ABAC provides an efficient approach to accommodate a wide breadth of access control policies and simplify permission management. Recently, several practical frameworks, such as Extensible Access Control Markup Language (XACML) [8] and Next Generation Access Control (NGAC) [9], have been proposed to provide a standardized way for expressing and enforcing vastly diverse access control policies on various types of data services. These frameworks possess two major features:

- **Diversification**: the ability to extract attribute values from multi-source, various places, different types, e.g., GPS location, time, visitor traffic, or threat level;
- **Dynamicity**: the ability to provide runtime support for acquiring the policy and attributes associated with a subject, object, action, or environment in which access requests occur.

In other word, access control logic of ABAC engine can determine who should have access to what resources under what circumstances and by taking what actions. Hence, the ABAC is particularly useful for an open environment, such as blockchain. However, we should note that there exist still several actual challenges that need to be solved.

One of the core challenges to be addressed is how to integrate access control mechanisms in ABAC into blockchain. An effective solution for this problem is to construct various blockchain’s transactions corresponding to the steps of access control procedure, e.g., store, ingest, organize and retrieve, for digital asset managed by DAM. By means of this approach, a registered user is allowed to publish his/her own digital assets to all members in blockchain, and then the platform can ensure the member’s access abided by common ABAC’s rules. Furthermore, these transactions recorded in the blockchain achieve the goal of ownership claiming, asset escrowing, transparency and verification of all steps.

**Our Contribution.** In this paper we focus on a new generation of digital asset management platform in a decentralized organization or alliance. This platform can support flexible and diverse permission management, as well as verifiable and transparent access process. Exactly, we not only provide a complete new access control framework and procedure, but also conduct researches on concrete implementation techniques. Our contributions are listed as follows:

- We present a new digital asset management platform, called DAM-Chain, with Transaction-based Access Control (TBAC) which integrates the distribution ABAC model and the blockchain technology. In this platform, the ABAC provides flexible and diverse authorization mechanisms for digital asset escrowed into blockchain, and the transactions in blockchain serve as verifiable and traceable medium of access request procedure.
- We present four types of transactions to describe the TBAC access control procedure, and provide the algorithms of these transactions corresponding to subject registration, object escrowing and publication, access request and grant. Moreover, the Bitcoin-type crypto-
graphic scripts are designed to guarantee authorization relationship among these transactions.

**Organization.** The rest of the paper is organized as follows. We describe our TBAC framework and various transactions in Section II and IV. The paper concludes in Section VI.

## II. System Framework

Digital asset management (DAM) has increasing benefits in booming global Internet economy, but it is still a great challenge for providing an effective way to manage, store, ingest, organize and retrieve digital assets. To achieve our goal, we introduce the distributed ABAC model into the exiting blockchain for utilizing the policies to restrict user’s access. Moreover, we intend to design new transaction-enabled mechanisms to guarantee the enforcement of these access policies correctly. These mechanisms would provide a good solution for enhancing the creditability of policy decision-making, as well as attribute diversification and dynamicity. This new solution should be sufficiently general to provide the following functions:

- **Asset security:** by using transactions, it needs to implement adequate authorization protocols to identify ownership and permit transfer or issuance of assets.
- **Secure asset issuance:** the shared objects are escrowed into blockchain by using a verifiable transaction, and then the access rules and policies will be used to prevent unauthorized entities from accessing to the object in the open environment.
- **Distributed Permission:** multiple authorization centers and all relevant parts can make a comprehensiveness decision of access request by providing the different attributes and certifications.

Blockchain and ABAC model also provide some good properties to implement our goal, e.g., the blockchain’s transactions inherently provide authenticity, integrity, traceability and somewhat anonymous, and the ABAC model is featured as flexibility, dynamicity and diversity for resource management.

### A. Standard ABAC Model

We illustrate the standard ABAC model given by NIST, which is the foundation of subsequent research. Within the NIST’s ABAC model [6], there exist four kinds of entities, i.e., subject, object, action, and environment. The characteristics of these entities are defined as attributes. According to these attributes, access policy extracted from common rules is used to determine whether a subject requests to perform operations on objects should be allowed under a specific environment.

The ABAC’s reference architecture is shown in Fig. 2. This architecture includes four service nodes such as policy enforcement point (PEP), policy decision point (PDP), policy information point (PIP), and policy administration point (PAP). Also, PDP and PEP functionality can be distributed or centralized, and they constitute so-called authorization service (AS). For an access request, the workflow of this architecture is described as follows:

1. The PEP intercepts the access request from an authenticated subject and sends the request to the PDP.
2. The PDP makes access decision according to access policy generated by PAP and the attributes of subject, object, and environment obtained by querying the PIP.
3. The final decision result given by the PDP is sent to the PEP, and then the PEP fulfills (either permits or denies) the access request according to the decision of PDP.

![Fig. 2. The NIST’s ABAC model.](image)

The above architecture is also composed of two repositories and many environment perception modules. Two repositories store and manage common access rules and entity attributes, respectively. The environment perception module can acquire the environment information at the current time relevant to a request, in which these information may include the current time, location, threat level, device’s type, etc.

### B. Threat Model

In this work, we mainly consider the threats from the semi-trusted cloud server and malicious client in an open and large-scale resource sharing. Here, we first assume cloud itself is semi-trusted, which means it honestly follows protocols and does not pollute data integrity actively as a malicious adversary, but it may try to find out as much secret information of stored data as possible. Malicious clients may try to access the data without permission by data owner. The goal of our system is to guarantee that only authorized client can access the data and conversely unauthorized clients or cloud will learn nothing by using access control and resource encryption.

In addition, we assume that our TBAC is deployed in an open and untrusted environment. All components are decentralized and distributed across the complete platform. Moreover, the attacker can eavesdrop and counterfeit the communication between any two components in platform. However, we do not assume that the attacker can corrupt any functional components. To these assumptions, the security of TBAC focuses mainly on three aspects: transaction security, authorization security, and decision-making security. We dive into the details as follows.

## III. Our TBAC Platform

We present a new digital asset management platform, called DAM-Chain, with transaction-based access control (TBAC) platform on blockchain and cloud storage to realize our design...
goal. This platform uses the blockchain as the core of resource distribution and sharing while introducing ABAC model to implement flexible resource access authorization.

A. Our TBAC Construction

The TBAC platform is perfectly engaged in “off-chain” storage, where blockchain as object directory is adopted for the retrieval service of resources, and the DDSs are actually used to deposit them by using cloud computing environment. The advantage of this storage way is to reduce the overhead of user’s management data. Considering the privacy of shared objects, they will be stored into the DDSs in an encrypted way. The encryption process is implemented by the object’s owner before submitting into the DDS.

As shown in the Fig. 3, the new TBAC consists of a blockchain and some additional modules under ABAC model. These additional modules are described as follows.

- **Data Depository Server (DDS)**: stores the owner’s escrow objects, and provides access service for the shared objects.
- **Authorization Service Unit (ASU)**: evaluates and enforces access decisions in response to the request from a subject requesting access to a protected object.
- **Policy Depository Server (PDS)**: manages and retrieves common access rules, and produces the access policy from the rules according to the user’s request.
- **Attribute Grant Unit (AGU)**: serves as the acquisition source of attribute values required for policy evaluation.

The presented TBAC platform is consistent with the NIST’s ABAC model [6], so that each module contains a certain functional point in the ABAC model. Specifically, the AGUs, as function extension of the PIPs, are dispersed into the whole blockchain network, e.g., environment attribute values could come from a client-side equipment where the user sends an access request. Moreover, the attribute, acquired from the AGU, would be issued in a way of verifiable security Token, called attribute Token, in order to prevent a forgery of attribute.

The PDS can be considered as a specific implementation of the PAP in the ABAC model. It is responsible for managing the common access rules which can be combined together to express policy according to the user’s access request.

The ASU, including PEP and PDP, is main execution agent to monitor, evaluate, and enforce the decision in terms of the user’s access require. PEP and PDP functionality may be put together or separated from each other. For example, the overall ASU can be deployed on the client side. However, this setting, which requires high communication and computation overheads, is not suitable for lightweight devices. Instead, we would prefer to put the PDP into the blockchain’s node, but to lay the PEP into the client’s application in order to improve the performance.

B. The Workflow of TBAC

The TBAC platform provides the entire process management of resource distribution and sharing, including subject register, resource publish, permission management of access acquire. In TBAC, any user must register to the platform by a certain AGU. For a registered user, the AGU manages the user’s subject attributes and submits the user’s information (encapsulated into subject registration transaction, SRT) into the blockchain.

While intending to share resource, the owner sends the encrypted object and its attribute information to a DDS, and then the DDS submits the object’s information (encapsulated into object escrow transaction, OET) into the blockchain. By means of this approach, the DDS implements the object escrow and the ownership claiming.

We next turn our attention to the process of access request. After a subject applies for access to an encrypted object, the PEP of ASU interprets and transfers this request (called access request transaction, ART) to the PDP and all blockchain nodes, and then waits for the decision result. The workflow of policy decision-making is described as follows.

1) **Generating Policy**: upon receiving the request, the PDP inquires of PDS about its access policy. In response, the PDS produces the policy relevant to the request from the PAP stored in the blockchain.

Fig. 3. The framework of TBAC model.
policy repository, and then generates and sends it and the corresponding cryptographic policy to the PDP.

2) **Acquainting Tokens**: upon receiving the responsive policy, the PDP selects the necessary attributes that satisfy the policy, and then queries these attributes' tokens to the relevant AGUs. In response, the AGU generates and returns the attribute Tokens if the queried attributes are valid at the current time.

3) **Making Decision**: the PDP makes the access control decision based on the received cryptographic policy and the attribute Tokens. If the decision result is True, the PDP confirms this result to blockchain by sending its ART signature to all nodes. It finally sends the Token (called **Authorized Token**) of decision result to the PEP.

4) **Decrypting Object**: for an authorized decision, the PEP firstly requests the DDS to pass the target object back. The DDS returns the encrypted object and appends its signature into the corresponding ART Transaction. The PEP next uses the authorized Token to decrypt the returned object, and then confirms this access to blockchain by sending its ART signature to all nodes.

5) **Confirming Transaction**: when all blockchain nodes receive the confirmation signature from the PEP, the blockchain platform starts the consensus protocol to confirm that the ART transaction is complete, and records the complete ART (called **access granted transaction, AGT**) into the blockchain. Finally, the PEP fulfills the operation requested by the subject.

The only thing that is necessary for the subject is to authenticate his identity before entering the platform. Moreover, it is unnecessary for subject to retain any key corresponding to resource decryption. This means there is no need for the user to perform any cryptographic operation in addition to applying for an access request.

In the TBAC platform, an attribute Token is a one-time security proof for the authorized attribute issued by the AGU. We consider the Token as a public verifiable “ticket” to prove that the attribute is what they claim to be. This ticket usually contains the information of the issuer, attributes, issue time, as well as a tag, where the tag is a simple signature that verifies the authenticity of all other information by using the issuer’s public key.

C. **Access Rules and Policies in ABAC**

In TBAC, we construct the PDS by using XACML that defines a policy specification language and reference architecture for ABAC implementation. Let $S = \{s, o, a, e\}$ denote a set that consists of subject, resource, action, and environment attribute. Here, attribute instances are specified as name-value pairs, where attribute name denotes the property or characteristic associated with certain attribute in $S$. For example, in a medical setting, $Role(s) = doctor$ denotes the attribute name $Role$ associated with a subject $s$ is doctor, similarly, $Ward(o) = pediatrics$, $ResourceID(o) = medical - records$, $Time(e) = 12:11$, and so on.

In TBAC, an access policy is composed of a target and a set of rules. A target defines a simple Boolean condition that, if satisfied by the attributes, establishes the need for subsequent evaluation by a PDP. We use the form of “attribute category : attribute name : attribute value” to express attribute instance, where symbol ‘:’ is used as separator. For example, “subject : role : doctor” denotes $Role(s) = doctor$. Considering a target may contain multiple attribute instance, we provide a way of reconciling these individual decisions, called "one-and-only", to restrict that each attribute must satisfy one condition. The target of this example applies to "All read or write accesses to medical records by a doctor or intern", that is, $target(s, o, a) := role(s) \in \{doctor, intern\} \land resourceID(o) = medical - records \land actionID(a) \in \{read, write\}$.

In addition to a target, a rule includes a series of Boolean conditions that if evaluated True have an effect of either “Permit” or “Deny”. The conditions of a rule are typically more complex and may include functions involving logical operators (e.g., and, or) and relation operations (e.g., $\leq, \geq, =$) for the comparison of attribute values. For example, the first rule denotes that any access request will be denied if the ward assigned by subject is not the same ward where the patient is located, i.e., $rule1(s, o) := WardAssignment(s) \neq WardLocation(o)$.

Similarly, the second rule denotes $rule2(s, a) := role(s) = intern \land actionID(a) = write$, and the third rule also is $rule3(s, o) := role(s) = doctor \land patientstatus(o) = critical$. Finally, we discuss the several standard combining methods that combine multiple rules into a single policy, including the following:

- **Permit-overrides**: if any decision evaluates to Permit, then the result is Permit, otherwise the result is Deny.
- **Deny-overrides**: if any decision evaluates to Deny, or no decision evaluates to Permit, then the result is Deny. If all decisions evaluate to Permit, the result is Permit.

In this example, we apply for the permit-overrides to integrate rule1, rule2 and rule3 together, that is,

$$policy(s, o, a, e) := target(s, o, a) \land (\neg(rule1(s, o) \lor rule2(s, a)) \lor rule3(s, o))$$

denotes the access is denied if only the conditions stated in rule1 or rule2 apply except for the access under a critical situation.

IV. **TBAC Transactions**

The TBAC platform employs transactions to ensure that only transactions conforming to the requisite policies are authorized and registered into the blockchain. Basically, a transaction is a data structure that encodes a transfer of access request and authorization process among participants in the platform. Everything in the blockchain is designed to ensure that transactions can be created, propagated on the network, validated, and finally added to the blockchain (as the global ledger). In the proposed TBAC platform, there exists four main
transactions to enforce fine-grained access policies. These four transactions are described as blow.

A. Subject Registration Transaction

The SRT is used to record the information of one or more subjects, and must be signed and issued by the AGU who manages the subject’s attribute information. In Algorithm 1 we show a high-level example of how a SRT transaction is constructed. The public keys of subjects are converted into Pay-to-Public-Key scriptPubKey scripts (explained below), while the output scripts are accompanied by signature of the corresponding AGU’s private key.

Algorithm 1 Procedure for creating SRT.

Input: Subject information; Subject attributes; Authentication information; AGU information;
Output: Subject registration transaction;
1: /*List all subjects and their information, attributes and authentication information*/
2: for all subject S do
3: public-attr ← get S’s attributes information;
4: pub, priv ← get S’s keypair;
5: addr ← get pub’s hash;
6: scriptPK ← (OP_PUSHDATA(33),pub,OP_CHECKSIG);
7: Add (addr, tx_index, public-attr, scriptPK) to list of subject;
8: end for;
9: /*Specify AGU’s information*/
10: AGU ← AGU’s network address;
11: pub,priv ← get AGU’s key pair;
12: sig ← sign_prv(All except scriptSig in SRT);
13: scriptSig ← (OP_PUSHDATA(72), sig);
14: Transaction ← (version, AGU, subject, time, tx_index, scriptSig);
15: return Transaction;

An instance of the executing result of Algorithm 1, where the SRT contains four types of information: 1) subject information, e.g., subject’s wallet address (called addr) and this transaction’s index and sequence number (expressed as tx_index#sequence); 2) subject attributes, e.g., the list of public attributes (public-attribute); 3) authentication information, e.g., the script of public key (scriptPubKey); and 4) AGU information, e.g., AGU’s network address (AGU) and the script of its signature (scriptSig).

Similar to Bitcoin, the wallet is used to store the user’s public/private key pair. When the user is registered into the platform, the wallet software is installed and a public/private key pair with an elliptic curve digital signature algorithm (ECDSA) is generated. The public key is then hashed and this hash value serves as the wallet address ("addr") that is transferred from/to in TBAC.

The above-mentioned scripts are sequences of instructions called opcodes that get executed by all entities in our platform. In particular, our TBAC platform makes use of Bitcoin’s scripting language, that is stack-based and without loops. Moreover, TBAC employs five kinds of scripts. We here give all the types as follows:

- **Type I: pay-to-public-key** contains two components:
  - scriptPubKey: <pubKey>, OP_CHECKSIG
  - scriptSig: <sig>

- **Type II: pay-to-public-key-hash** contains two components:
  - scriptPubKey: OP_DUP, OP_HASH160, <pubKeyHash>, OP_EQUALVERIFY, OP_CHECKSIG
  - scriptSig: <sig>, <pubkey>

- **Type III: pay-to-script-hash** contains two components:
  - scriptPubKey: OP_HASH160, <scriptHash>, OP_EQUAL
  - scriptSig: <sig>, <script>

- **Type IV: multiple signature** contains two components:
  - scriptPubKey: M, <pubKey A>, <pubKey B>, <pubKey C>, N, OP_CHECKMULTISIG
  - scriptSig: OP_0, <sig B>, <sig C>

- **Type V: OP_Return** contains two components:
  - scriptPubKey: OP_RETURN, <data>
  - scriptSig: NULL

For example, the subject’s script takes the structure of Type I to store the user’s public-key into “scriptPubKey”, in which OP_PUSHDATA(33) is to push the subsequent 33-byte public key into the stack, and “OP_CHECKSIG” expects two values on the stack to be verified. Similarly, the AGU’s script “scriptSig” takes the structure of Type I to push the 72-byte AGU’s signature into the stack by using OP_PUSHDATA(72). The execution process of scripts refers to [10].

B. Object Escrow Transaction

The OET, as escrow credential and ownership claim, is used to record various information of protected objects. Algorithm 2 describes the procedure of an OET transaction, where the OET requires the signatures from both the owner and the DDS who is the actual object’s depository. Thanks to openness of blockchain, the OETs are publicly accessible to all members in the whole platform, so that any member can retrieve the required objects conveniently.

The structure of OET produced by Algorithm 2 can be divided into two parts: one is the owner’s profile and the other is the escrowed objects’ profile. This kind of structure describes the “one-to-many” relationships between the owner and the escrowed objects.

The owner’s profile contains two types of information: 1) the owner’s information, i.e., the owner’s SRT index and sequence number (owner_tx_index), which is used to find owner’s information stored in the corresponding SRT; and 2) ownership claim, i.e., the script of the owner’s signature (scriptSig). As mentioned above, the script of TBAC currently utilizes two different scriptSig/scriptPubKey pairs which can be cryptographically linked together to enforce the agreement. This kind of agreement can be assessed by using the public-key (scriptPubKey) to verify its holder’s signature (scriptSig).
The script `scriptSig` stores the owner’s ECDSA signature over the transaction itself. This signature, verified by the `scriptPubKey` in SRT related to `owner_tx_index`, proves that the objects in this transaction were possessed by the subject in the SRT.

### Algorithm 2: Procedure for creating OET

**Input:** Profile of owner; Profile of escrowed objects;  
**Output:** Object escrow transaction;  
1. /*Specify owner*/  
2. owner_tx_index ← the owner’s SRT index#sequence  
3. for all escrowed object O do  
   4. name ← get O’s identity;  
   5. attribute ← get O’s attributes;  
   6. DDS ← get DDS’s URL address;  
   7. pub, priv ← get DDS’s keypair;  
   8. `sign` ← `signpriv`(O’s information and attributes);  
   9. scriptSig ← OP_PUSHDATA(72) `sign`;  
   10. Add (name, DDS, tx_index, attribute, scriptSig) to list of escrow;  
11. end for;  
12. pub, priv ← get owner’s key pair by owner’s SRT;  
13. `sig` ← `signpriv`(All except `scriptSig` in OET);  
14. scriptSig ← OP_PUSHDATA(72), `sig`;  
15. Transaction ← (version, owner_tx_index, escrow, time, tx_index, scriptSig);  
16. return Transaction;

The profile of escrowed object consists of three parts: 1) object information, e.g., the object’s name (`name`), the DDS’s address (expressed by URL), and its transaction index (`tx_index`); 2) object attributes, e.g., the list of attributes (`attribute`); and 3) escrow credential, i.e., the DDS’s signature (`scriptSig`). This signature, authenticated by the PKI public-key certificate obtained from the URL of DDS¹, proves the object was escrowed by the owner of the certificate.

### C. Access Request Transaction

The ART is a credible ticket to take note about object’s access procedure in the form of transaction logs. It contains all necessary information of access request generated by PEP, and these information will be used by the subsequent decision-making for the access request. As shown in Fig. 4, we describes the procedure of an ART transaction. Considering that the ART contains several unauthorized signatures (expressed by `empty`), it is not allowed to append into the blockchain as a valid access permission. In ABAC, access request is generally used to specify what `subject` wants to have access to what `object` with what kinds of actions. For this reason, the ART ought to contain three segments (`subject`, `object`, `action`) of an access request. As shown in Fig. 4, the `subject` segment consists of the subject’s SRT index (`tx_index`) and the subject’s signature (`scriptSig`), which can be verified by the `scriptPubKey` in the indexed SRT. In the same way, the `object` segment consists of the object’s OET index (`tx_index`) and an empty DDS’s signature (`DDS_Sig`), which will be verified by the PKI certificate issued by the DDS. The `action` segment is the list of all authorized operations, each of which is expressed as “company:technology#action”.

![Fig. 4. Example of access request transaction.](image-url)

In addition, the ART contains the information that is used to provide dynamic authentication from relevant authorized entities in TBAC platform. These authentication information consists of the PEP/PDP address (expressed by URL) and their signatures (PEP_Sig and PDP_Sig), as well as access request time and period of validity. As mentioned in Section III-B, the initial signatures of DDS, PEP, and PDP are empty, but they will be signed according to the order of PDP, DDS and PEP for a valid request.

### D. Access Granted Transaction

The AGT is the final form of ART after all empty signatures are fulfilled, and then it will be stored into blockchain as an access log once all of signatures are validated by the block generator. For the sake of clarity, we next describe the authentication relationship between the above-mentioned transactions (including SRT, OET and AGT) and different entities (PDP, PEP, DDS, AGU, etc.) from an AGT-centered viewpoint.

The AGT is submitted to the blockchain until the four signatures in AGT are checked. Among subject’s scriptSig, DDS_Sig, PEP_Sig and PEP_Sig, only the scriptSig is written in scripting language. So that, the signature of subject is verified by performing function `executeScript` which sequentially executes the `scriptSig` and `scriptPubKey` written in scripting language, and then the `sigVerify` function is called respectively to verify the signature of the DDS, the PDP, and the PEP through using public-key certificates issued by the CA.

¹Several entities, including DDS, PDP, PEP and AGU, utilize PKI certificate to issue their public keys. The SRT may be another option for issuing their public key.
V. SYSTEM ANALYSIS

A complete authentication system can be generated from complex authorization or permission relationship among transactions and platform’s entities. Algorithm 3 is used to implement this procedure of AGT verification.

Algorithm 3 Procedure to verify AGT

Input: Access granted transaction;
Output: valid or invalid;
1: /*Verify each signature in the transaction*/
2: scriptPubKey ← extract from SRT according to subject’s tx_index;
3: if executeScript(scriptSig, scriptPubKey)==false then
4:    return invalid;
5: end if
6: DDS_Pub ← get DDS’s public key from CA;
7: if sigVerify(DDS_Pub, DDS_Sig)==false then
8:    return invalid;
9: end if
10: PDP_Pub ← get PDP’s public key from CA;
11: if sigVerify(PDP_Pub, DDS_Sig)==false then
12:    return invalid;
13: end if
14: PEP_Pub ← get PEP’s public key from CA;
15: if sigVerify(PEP_Pub, DDS_Sig)==false then
16:    return invalid;
17: end if
18: return valid;

For an access request, the subject segment of AGT is used to point to a requester’s SRT that contains the public key of the requester. Such that, the requester’s signature in the AGT can be verified by using this public key. For the accessed object, the object segment in the AGT points to the OET which contains the address of the corresponding DDS. The public key certificate of DDS can be obtained according to this address. Therefore, any one can use the DDS signatures, stored in both the object in AGT and the escrow in OET, to validate the DDS authorization. The owner segment in OET can also help us to find out the SRT of object’s owner through the tx_index, and the owner claims ownership of the object using the signature signed by his private key. Finally, the PDP_Sig and PEP_Sig in AGT are considered as access permissions issued by the PDP and the PEP. In addition, the AGU in SRT stores its own address which is convenient for acquire the subject’s attributes.

According to the above description, our TBAC platform implements these functions as follows:

- **Access Openness**: the platform supports data sharing and exchanging with unlimited number of users in the public network environment.
- **Authorized Trusteeship**: after signing the object escrow transaction, the owners do not need to participate in subsequent access authorization.
- **Rule Generality**: the rules in TBAC can support the general security restrictions (such as read-up/write-down), and they can change as required.

- **Policy Dynamics**: for the current request, the policy is produced dynamically from the security rules according to the current state of the request.
- **Access Traceability**: the process of digital asset sharing and exchanging is traceable via transactions, and their access authorizations are cryptographically verifiable.

VI. CONCLUSIONS

In this paper we take transaction as a bridge to integrate ABAC and blockchain into a new platform for resource distribution and sharing. Our proposed platform supports flexible and diverse permission management, as well as verifiable and transparent access authorization process. We hope our research could provide useful reference for globe resource sharing.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (Grant No. 61472032), NSFC-Genertec Joint Fund For Basic Research (Grant No. U1636104) and Joint Research Fund for Overseas Chinese Scholars and Scholars in Hong Kong and Macao (Grant No. 61628201).

REFERENCES