ABERand: Effective Distributed Randomness on Decentralized Ciphertext-Policy Attribute-Based Encryption

Liang Zhang, Haibin Kan, Zening Chen, Ziqi Mao, and Jinjie Gao

***Abstract*—Distributed randomness is very useful for many applications, such as smart contract, proof-of-stake-based blockchain, elliptic curve generation and lottery. Randomness beacon protocols are proposed, which are aimed at continuously distributed randomness generation. However, a reliable source of distributed randomness is gained with difficulty because of Byzantine behavior, which may lead to bias for distributed randomness. These Byzantine behaviors include, but not limited to, the "last actor" problem, DoS attack and collusion attack. Various cryptography schemes have been used to generate distributed randomness. Current constructions face challenging obstacles due to high complexity and bias problems. Given these barriers, we propose a new protocol that is the first precept to utilize attribute-based encryption in a commit-and-reveal scheme for distributed randomness (ABERand). Compared to existing public distributed randomness protocols, ABERand possesses distinguished flexibility, security and efficiency. It is primarily because of trading space for time. More specifically, we resolve the "last actor" problem and make ABERand an intensive output randomness beacon with communication complexity $O(n^3)$, computation complexity $O(1)$, verification complexity $O(n)$ and communication complexity $O(n)$ of nodes adding/removing.**

***Index Terms*—distributed randomness, ciphertext-policy attribute-based encryption, space-for-time, commit-and-reveal, blockchain**

I. Introduction

How to generate public distributed random values among mutually distrustful nodes over a distributed network was first proposed by Blum [1] in 1981, thereby introducing coin-tossing protocols. Public randomness beacon, which aims to generate fresh, unpredictable and unbiased random values at certain intervals, was formalized by Rabin [2] in 1983. Many facts have indicated that the process of generating randomness should be more public and transparent to establish credibility and impartiality. For example, transparent machines with balls flying inside which seems to create complete chaos for the final result, are commonly used for national lotteries. However, a method like this is exposed to be unreliable.[1] The teams in NBA that missed the playoffs in the previous year will participate in a lottery to determine the draft order to sign top amateur players. Technologies related to blockchain [3], [4], such as consensus protocols [5], [6] and smart contracts [7], [8], have increased the demand for randomness beacons. These situations indicate that random values should not be predicted before being generated but must be publicized later. Recently, coin-tossing protocols and distributed randomness beacons have received increasing attention because randomness is both a vital component and application of blockchain. It is critical that the random number must be generated reliably.

The rise of blockchain has brought about a new perspective on how to gain and how to use distributed randomness [5], [6], [9], [10], [12]. Permissionless blockchain systems implement consensus protocol by picking one or a subset of participants to pack the new block and announce the network's latest state. Some blockchain systems select leaders via proof of work (PoW) [3], in which candidates solve difficult puzzles. However, the performance of PoW is low, and the computational cost of PoW is high. Proof-of-stake (PoS) [5], [11] protocols, which rely on virtual resources, needs an algorithm to elect a leader for packing the next block. Kiayias et al. [5] pointed out that leader election is a fundamental issue of PoS-based protocols since an adversary may manipulate the result if any entropy is introduced. The generation of manipulation resistant and unpredictable distributed randomness beacon is a solution of leader election for PoS-based protocols. It is of vital importance that distributed randomness beacons possess properties of availability, unpredictability, bias resistance and public verifiability [13]. In other words, public distributed randomnesses should be unpredictable before generation, should be unbiased at the time of generation, and should be publicly verifiable after generation, even if there are malicious or collusive participants. We follow the notions [13], [16] to describe the properties of a beacon:

Liveness/Availability. For a majority of honest participants, the protocol produces randomnesses output continuously.

Bias resistance. Each produced randomness is an unbiased and uniformly random value.

Unpredictability. No one learns anything about the final randomness until it is calculated.

Public verifiability. Any third party could verify the correctness of the final randomness with unforgeable proofs that produced in the protocol.

Flexibility is also an important indicator that nodes in

Liang Zhang, Haibin Kan, Zening Chen, Ziqi Mao, and Jinjie Gao are with Shanghai Key Laboratory of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai 200433, P. R. China, and Fudan-Zhongan Joint Laboratory of Blockchain and Information Security, Shanghai Engineering Research Center of Blockchain, Shanghai Institute of Intelligent Electronics & Systems, Shanghai Institute for Advanced Communication and Data Science, Shanghai, P.R. China, Email: hbkan@fudan.edu.cn

This work was supported by National Natural Science Foundation of China (Grant No. 61672166), the Plan of Shanghai Excellent Academic Leaders (Grant No. 16XD1400200), the Innovation Plan of Shanghai Science and Technology (Grant No. 16JC1402700) and Shanghai Leading Talent Programmes.

the group could join or leave because of either crashes or deliberation.

A. Background and Motivation

A substantial number of studies related to distributed randomness beacon protocols have been published both in academia and in the industry due to the rise of blockchain. Some protocols, such as national lotteries [14], NIST Randomness Beacon[2], and Lavarand[3], rely on third parties or physical sources of noises. Some other protocols abide by the commit-and-reveal protocol or commitment scheme[4] [17] principle. In the commit-and-reveal protocol, values are chosen and "locked" in the commit phase, and they are verified after the reveal phase. To illustrate the protocols from the perspective of cryptography, they are divided into PoW-based ones [9], [15], hash-based ones [17], [18], publicly verifiable secret sharing (PVSS)-based ones [5], [16], [19], and signature-based ones [20], [21].

PoW-based method. In the traditional method, financial data are regarded as a seed of public random numbers [22]. Heuristically, public random numbers can be obtained as a side-effect of Bitcoin's [9] proof-of-work-based consensus system. Compared to traditional financial data, the bitcoin market is open 24 hours a day, 7 days a week, and no trusted or centralized party is needed to maintain the system. The incentives of bitcoin consensus make every participant scramble to publish the nonce of a new block header. The nonce is therefore used to generate distributed randomness. However, if a miner gets more benefit than the incentives from manipulating the randomness by withholding attack [23], the final randomness will be biased. [9] focused on analyzing the financial cost when a miner withheld the nonce rather than providing an unbiased solution. Pierrot et al. [24] presented a detailed analysis of how a malicious entity could manipulate the random numbers on a public blockchain, even with a limited financial budget and computational power.

Hash-based method. Popov's research [25] introduced the principle of the hash-based distributed randomness protocol. To solve the "last actor" problem, wherein the last one has the option never to reveal his/her input, participants are divided into groups, and strong settings are assumed. More precisely, the settings are 1) at least one group contains only honest members, and 2) no group consists entirely of colluding parties. Iterated hash [17] is the basic idea to form a delay function [17], [18], [26], which is also called a slow-time function. A delay function relies on proof-of-Work blockchain [18] can get rid of the "last actor" problem.

PVSS-based method. In 1999, Schoenmakers [27] proposed the publicly verifiable secret sharing (PVSS) scheme. A dealer in PVSS is a temporary participant who distributes shares according to his/her secret. PVSS is a verifiable secret sharing scheme with the property that anyone can verify the validity of the shares distributed by the dealer. It is reported that Ouroboros [5] was the first to adopt PVSS as a provably secure proof-of-stake blockchain protocol. It relies on a combination of PVSS and other cryptographic primitives to obtain the last verifiable randomness. To achieve scalability, RandHound and RandHerd are systems with a failure probability of 0.08%, based on PVSS and collective signing [28]. Scrape [19] optimizes a variant of Schoenmakers's PVSS with an existing bulletin board. HydRand [13] was proposed as a standalone, self-contained protocol with a permissioned and fixed number of members system model.

Signature-based method. Algorand [20] builds a distributed ledger by combining a randomized Byzantine agreement protocol [29] and a randomness beacon based on Verifiable Random Function (VRF) [30]. VRF is used to produce unique signatures. However, strictly speaking, the produced randomness is not bias-resistant. Dfinity [21] is a blockchain project, and its core technique lies in a decentralized randomness beacon, which acts as a VRF. The innovative beacon is based on the threshold Boneh-Lynn-Shacham (tBLS) [31] signature and distributed key generation (DKG) [32], [33].

B. Our Contributions

We define a totally decentralized threshold multi-authority ciphertext-policy attribute-based encryption (MA CP-ABE, more details in II-D)) protocol based on the Rouselakis's construction [34] and use it as a commit-and-reveal scheme to construct ABERand as a public distributed randomness beacon. In a commit-and-reveal scheme with the property of enforcement 6, no adversaries could succeed to abort the protocol or bias the beacon in a meaningful way. Compared to PoW-based, hash-based and other public-key based approaches, ABERand solves the fatal "last actor" problem. Compared to PVSS-based and signature-based approaches that share secrets by broadcasting over the internet, ABERand hides secret sharing (Linear Secret Sharing Schemes, LSSS) in ciphertext, thus lowering complexity and enhancing flexibility. Moreover, compared to currently existing methods, ABERand lowers the computation and verification complexity on the whole. The whole network communication complexity is $O(n^3)$, permitting no more than $1/2$ malicious nodes. The single node computation complexity is only $O(n)$ and the verification complexity of final randomness is only $O(1)$ in ABERand, while the single node computation complexity is only $O(1)$ and the verification complexity of final randomness is only $O(n)$ in ABERand$'$. The network communication complexity is only $O(n)$ when nodes join or leave.

C. Organization

Section II introduces the notation and cryptographic primitives that will be used throughout the paper. Section III provides the model of our protocol. Section IV briefly gives the idea of a totally decentralized threshold MA CP-ABE in a commit-and-reveal scheme, followed by concrete construction of ABERand in Section V. In section VI, we evaluate the performance and properties of our protocols. Section VII, we analyze the security and possible attacks of the protocol. Finally, in section VIII, we conclude with directions for future work.

II. PRELIMINARIES

In this section, we provide a brief review of the background theory on access structures, linear secret-sharing schemes, bilinear maps, CP-ABE, multi-authority CP-ABE, eXclusive-OR to Generate Randomness and commit-and-reveal scheme.

A. Access Structures

Definition 1: [36] Let P = $\{P_1, P_2, ..., P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,P_2,...,P_n\}}$ is monotone if for any B and C, the following holds: if B $\in \mathbb{A}$ and C $\subseteq$ B, then C $\in \mathbb{A}$. An access structure is a collection $\mathbb{A}$ of nonempty subsets of $\{P_1, P_2, ..., P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,P_2,...P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in A are called the unauthorized sets.

In an ABE system, the roles of the parties are defined by attributes. An access structure in ABE contains the authorized sets of attributes. In our construction, we only consider monotone access structures, which means that when a user acquires more attributes, he will not lose his possible decryption privileges. There are different forms to describe access policies [38], such as minimal form access structures, monotone boolean formulas, threshold-gate access trees and monotone access trees. In this paper, we take advantage of the threshold-gate access trees.

B. Linear Secret Sharing Schemes

Definition 2: [42] Let p be a prime and Ω be the attribute universe. A secret sharing scheme Π with a domain of secrets $\mathbb{Z}_p$ realizing access structures on Ω is linear over $\mathbb{Z}_p$ if:

1. The shares of a secret z $\in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$.

2. For each access structure $\mathbb{A}$ on Ω, there exists a matrix a matrix A $\in \mathbb{Z}_p^{l\times n}$, called the share-generating matrix, and a function σ that labels the rows of A with attributes from Ω, i.e., σ: [l] $\longrightarrow \Omega$, which satisfy the following: during the generation of the shares, we consider the column vector υ = $(z, r_2, ..., r_n)^{\perp}$, where $r_2, ..., r_n \xleftarrow{R} \mathbb{Z}_p$. Then, the vector of l shares of the secret z according to Π is equal to λ = A$\upsilon \in \mathbb{Z}_p^{l\times 1}$. The share λ_j with j $\in$ [l] "belongs" to attribute σ(j).

We will be referring to the pair (A, σ) as the policy of the access structure $\mathbb{A}$.

As shown in [36], any monotone access structure can be achieved by a linear secret sharing scheme. Any monotone access structure can be converted into LSSS matrices. [38]

C. Bilinear Maps and Complexity Assumption

Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_1$ and e be a bilinear map, $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. The bilinear map e has the following properties [41]:

1. Bilinearity: for all μ, $\upsilon \in \mathbb{G}_1$ and a,b $\in \mathbb{Z}_p$, we have $e(\mu^a, \upsilon^b) = e(\mu, \upsilon)^{ab}$.

2. Nondegeneracy: $e(g, g) \neq 1$.

3. Computability: There is an efficient algorithm for computing $e(\mu, \upsilon)$ $\forall$ μ, $\upsilon \in \mathbb{G}_1$.

We call $\mathbb{G}_1$ a bilinear group if the group operation in $\mathbb{G}_1$ and the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ are both efficiently computable. The above definition considers the so-called symmetric groups, where the two arguments of the pairing belong to the same group. In general, there exist asymmetric bilinear groups, where $e' : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ and $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three different groups of prime order p. Several asymmetric instantiations of bilinear groups possess beneficial properties such as faster operations under the same security level and/or easier hashing to group elements.

Definition 3: (q-Decisional Parallel Bilinear Diffie-Hellman Exponent 2 Assumption). [34] Choose a bilinear group G of order p, and a non-degenerate bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. Let s,a,$b_1$,$b_2$,...,$b_q \in \mathbb{Z}_p$, $R \in \mathbb{G}_T$ choose at random. Let

$$D = (\mathbb{G}, p, e, g, g^s, \{g^{a^i}\}_{i\in[2q],i\neq q+1}, \{g^{b_j a^i}\}_{(i,j)\in[2q,q],i\neq q+1}, \{g^{s/b_i}\}_{i\in[q]}, \{g^{sa^i b_j/b_{j'}}\}_{(i,j,j')\in[q+1,q,q],j\neq j'})$$

No probabilistic polynomial-time algorithm A can distinguish the tuple $(D, e(g, g)^{sa^{q+1}})$ from the tuple (D, R) with more than a negligible advantage. The advantage of A is

$$\left| Pr[A(D, e(g, g)^{sa^{q+1}}) = 1] - Pr[A(D, R) = 1] \right|$$

where the probability is taken over the random choice of $s, a, b_1, b_2, ..., b_q \in \mathbb{Z}_p, R \in \mathbb{G}_T$, and the random bits consumed by A.

D. Multi-Authority CP-ABE

A great variety of ABE schemes have been proposed, and they are mainly divided into two categories: key-policy attribute-based encryption (KP-ABE), such as in [39], [40], and ciphertext-policy attribute-based encryption (CP-ABE), such as in [41]–[44]. In the KP-ABE scheme, an access structure is embedded in the secret key, and the ciphertext is associated with an attribute set. In the CP-ABE scheme, an attribute set is embedded in the secret key, and the ciphertext is associated with an access structure. In our situation, CP-ABE is our preferred choice. A nice feature of CP-ABE is that data can be encrypted without knowledge of the decryptors. Therefore, users can obtain keys for decryption sometime later after the data have been encrypted with some specific policy.

Most existing CP-ABE schemes [41]–[44] have only one authority that is responsible for attribute management and key distribution. This one-authority scenario can be problematic since the single authority can issue private keys to every user in the system. If the authority is malicious or compromised, the encrypted data are at risk. A multi-authority (MA) CP-ABE [34] system is comprised of the following five algorithms:

Global Setup(λ) $\longrightarrow$ GP. it takes in the security parameter λ and outputs global parameters GP for the whole system.

Authority Setup(GP, θ) $\longrightarrow$ SK_θ, PK_θ. Each authority θ takes GP as input to produce its secret key and public key pair, SK_θ, PK_θ.

Encrypt(M, (A, σ), GP, $\{PK_\theta\}$) $\longrightarrow$ CT. The algorithm takes in a message M, an access structure (A, σ), a set of

public keys, and the global parameters. It outputs a ciphertext CT.

KeyGen(GID, GP, u, SK_θ) $\longrightarrow$ $\mathrm{K}_{u,\mathrm{GID}}$. The algorithm takes in an identity GID, the global parameters, an attribute u belonging to the authority θ, and the secret key SK_θ for this authority. It produces a key $\mathrm{K}_{u,\mathrm{GID}}$ for this attribute and identity pair (GID, u).

Decrypt(CT, GP, $\{\mathrm{K}_{u,\mathrm{GID}}\}$) $\longrightarrow$ M. The decryption algorithm takes in the global parameters, the ciphertext, and a collection of keys corresponding to the attribute and identity pairs that all have the same fixed identity GID. Only and only if the collection of attributes u satisfies the access structure corresponding to the ciphertext, it outputs the message M.

Definition 4: [34] A multi-authority CP-ABE system is said to be correct if GP is obtained from the global setup algorithm, CT is obtained from the encryption algorithm on the message M, and $\mathrm{K}_{u,\mathrm{GID}}$ is a set of keys obtained from the key generation algorithm for the same identity GID and for a set of attributes satisfying the access structure of the ciphertext, Decrypt(CT, GP, $\mathrm{K}_{u,\mathrm{GID}}$) = M.

E. eXclusive-OR (XOR) to Generate Randomness

Given a set S of pseudorandom numbers in the range [0,b] that are uniformly distributed over the range and independent, the XORing calculation on them produces a new pseudorandom number in [0,b]. b is a power of 2.

Proof. It is easy to observe that the map $x \leftrightarrow x \oplus c$ is a bijection over the range [0,b] regardless of the value of the constant c. Thus, if x is uniformly distributed over the range, then so is $x \oplus c$. Since this is true for any constant c, it is also true even if c itself is a random variable, as long as it does not depend on x.

F. Commit-and-Reveal Scheme

A commit-and-reveal scheme or commitment scheme [45], is an efficient two-phase protocol in which one commits itself to a value or a statement. The committed result is commitment. The scheme has the following two properties:

Secrecy/Hiding: At the end of the first phase (or commit phase), no one else gains meaningful knowledge of the value or statement, even by all means.

Unambiguity/Binding: There is only one value or statement that is bound with the commitment even though one wants to cheat. The only corresponding value or statement, when uncovered, can be accepted as a legal "opening" of the commitment in the second phase (or reveal phase).

III. System and Threat model

We assume a partially synchronized model with pairwise connected but public channels network for broadcasting. An upper bound-time of the maximum delay of the network is Δt, and honest nodes could accomplish communication in this period. An epoch is a round to produce a distributed verifiable randomness, and we separate an epoch into several steps for the commit-and-reveal scheme. Nodes in each epoch are asynchronized, nodes in each step are asynchronized, but each step in an epoch should be synchronized. We target the number of participants in the system n in an epoch. f ($< n/2$) is the maximum number of malicious nodes that may crash or may cause Byzantine failures or disobey our protocol in an epoch. Malicious nodes, also called adversaries, can send their preset data through collaboration. A node is considered to be honest if it abides by the protocol; otherwise, it is considered to be faulty. We assume that the goal of the adversary is to bias or DoS-attack the protocol. The sender signs each message of the protocol. The counterparts only accept and act upon a message if it is correctly signed. If the adversary could disobey or abort the protocol by preventing the ciphertexts from being decrypted, then bias is introduced into the final result. We are considering adding or removing nodes in our model. There should be a reputation system, for practical usage, to prevent Sybil attack [37] that violates the $f < n/2$ assumption. And the reputation system is beyond discussion in this paper.

IV. Decentralized Threshold MA CP-ABE in a Commit-and-Reveal Scheme

From [38], we can know that threshold-gate access tree is equal to boolean formulas for an access structure. So a multi-authority CP-ABE is born to be a threshold MA CP-ABE. In some MA CP-ABE schemes, such as [34], [46], [47], multiple authorities jointly manage the attribute sets of the decryptors. [46] realizes a robust and verifiable multi-authority access control system in public cloud storage, but it has a certificate authority (CA). [34], [47] satisfy the scenario in which attributes are obtained from different authorities and guarantee that the secret key cannot be obtained by any authority alone. However, the attribute universe of [47] is restricted. A threshold MA CP-ABE scheme that is also somewhat decentralized [35]. We adapt MA CP-ABE [34] into a totally decentralized threshold MA CP-ABE protocol to produce public distributed randomness. By adapting Rouselakis's MA CP-ABE into a totally decentralized threshold MA CP-ABE, we mean his scheme is used in a decentralized way. We inherit his static model of security and complexity assumption [34].

A totally decentralized (t, n)-threshold multi-authority CP-ABE is a protocol that everyone is an authority, an encryptor and a decryptor. All participants have equal rights. It can achieve fine-grained access in various situations. Here we only consider a scenario in order that t-out-of-n could do "threshold decryption" 5. In this scenario, the participants use threshold access structures in which n attributes issued by n authorities are specified explicitly. To make it simple, an authority makes use of a publicly known value and his/her own identity as a unique attribute and issue the corresponding key to a decryptor. Concrete usage and construction in ABERand is described in V and V-C. In order to satisfy above scenario, the **KeyGen** algorithm should be invoked later than the **Encrypt** algorithm which ensures **Decrypt** algorithm is later than the **Encrypt** algorithm. In this way, we can make sure that an honest participant's ciphertext could act as a commitment.

Definition 5: threshold decryption. If an honest one of the n participants encrypts a message with the above threshold access structure. Later, any honest t-out-of-n could separately run **KeyGen** to generate keys which are the inputs of the **Decrypt** algorithm. No one needs to publicize their private key.

Definition 6: Enforcement. If a committer in a commit-and-reveal scheme is forced to open his/her commitment with high probability, we say the scheme has the property of enforcement. Enforcement is gained sometimes because of economic incentives, sometimes because of cryptographic primitives.

The decentralized threshold MA CP-ABE protocol works well in the commit-and-reveal scheme to produce public distributed randomness, because the **Encrypt** algorithm could "lock" values in the commit phase, the **KeyGen** algorithm and the **Decrypt** algorithm are done in the reveal phase. Apart from secrecy and unambiguity, "threshold decryption" guarantees the property of enforcement with an honest majority and a synchronized clock.

Proof of Secrecy. One's message is encrypted under the security model and complexity assumption [34]. Except for the possible keys generated by authorities, no other ways can be obtained to get the message. Even the collusion of limited malicious ones is impossible VII-A.

Proof of Unambiguity. A commitment (ciphertext) obtained from encryption of a stated value can be bound to more than one value is negligible. Furthermore, once a value is obtained by the decryption algorithm, it is convincing that the decrypted value is legal no matter whether the sender reveals it or not.

Proof of Enforcement. If a commitment (ciphertext) is well constructed in which the access structure of it is well-defined. An honest majority could always get the original stated value through threshold decryption.

The totally decentralized (t, n)-threshold multi-authority CP-ABE makes a commit-and-reveal scheme sound and "atomic", thus solving the "last actor" problem. To construct distributed randomness beacon means that the reveal phase should always be enforced correctly. The following sections will depict, in detail, how to put decentralized threshold MA CP-ABE into a commit-and-reveal scheme for the distributed randomness beacon.

V. ABERand Protocol

A. System Overview

Figure 1 is the flowchart that summarizes the overview of our protocol. During system setup, global parameters are calculated in a public way or by a delay function [26]. A participant generates his/her own private and public key pair according to global parameters. All participants exchange their public keys and store their secret keys privately. Each epoch of ABERand beacon is separated into 7 phases in detail. Each participant takes a random value as input and generates the ciphertext in the 1st phase. In the 2nd phase, participants share their ciphertexts. Then in the 3rd phase, participants generate keys for the current epoch and broadcast keys in the 4th phase. In the 5th phase, participants decrypt all valid ciphertexts and obtain a final random number by calculating eXclusive-OR of all messages in the 6th phase. Finally, the whole process can be verified publicly in the last phase. Agreement on the random number is automatic because of the correctness property of the CP-ABE scheme. It is obvious that a peer-to-peer network is utilized in the 2nd and 4th phases.

Same as CP-ABE [34], the threshold MA CP-ABE implementation have attributes in the form of "[attribute–id]@[authority–id]". To distinguish the attributes and to avoid collusion, the "attribute–id" and the "authority-id" are case-sensitive alphanumeric strings. The mapping T mentioned in Section V-C only extracts the part after the @ of the attribute string. For simplicity, all the attributes in ABERand follow the format of "k@nodeId", where k is the current epoch number, and "nodeId" is the authority's identity. Actually, attributes in ABERand are necessary for three aspects. Firstly, "[attribute–id]" represents the unique epoch number as a common-sense according to our protocol; Secondly, "[attribute–id]@[authority–id]" is used in access structure which makes threshold cryptography possible. Thirdly, attributes are designed in this way so that ABERand is a totally decentralized protocol. In traditional public-key encryption schemes without attributes, malicious participants have the opportunity to hide their private key which will bias or abort distributed randomness beacon.

B. Data Structure in Communication

In real application environments, the peer-to-peer network is very complicated due to the free joining and exiting of nodes and the existence of malicious nodes. To guarantee the transferred data is valid, a sender signs data. To describe the data structure, we simplify it as <message, ts, pk, sig>. The signing process is noninteractive.

C. Details of the Protocol

System Initialization. The system invokes the **Global Setup** algorithm in a public way and sets the initial round variable k to 1 (k increases by 1 at the end of every round). The global setup algorithm takes as input the security parameter λ and chooses a suitable bilinear group $\mathbb{G}$ of prime order p with generator g. It also defines a function H that maps global identities GID to elements of $\mathbb{G}$ and another function *F* that maps strings, interpreted as attributes, to elements of $\mathbb{G}$. Both of these functions will be modeled as random oracles in the security proof. T: U $\longrightarrow$ U_θ is a publicly computable mapping function that maps each attribute to a unique authority θ.

The initial number of participants is n; thus, the number of honest participants t should satisfy the demand $t \geq 1 + n/2$. Each of the n participants, with nodeId_θ as its identity, invokes the **Authority Setup** algorithm and acquires key pairs $\{\mathrm{PK}_\theta, \mathrm{SK}_\theta\}$. The authority setup algorithm chooses two random exponents, α_θ,$y_\theta \xleftarrow{R} \mathbb{Z}_p$, and calculates [34]

$$\mathrm{PK}_\theta = (e(g,g)^{\alpha_\theta}, g^{y_\theta}),\ \mathrm{SK}_\theta = (\alpha_\theta, y_\theta).$$

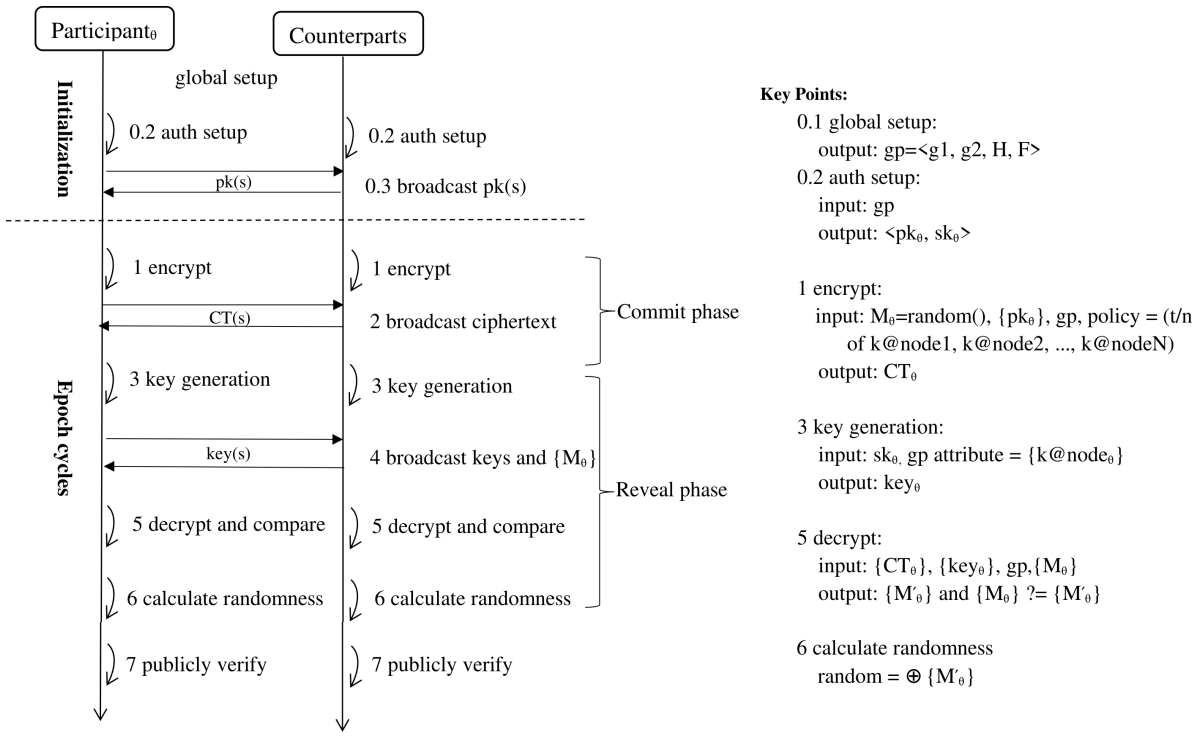

Fig. 1. System overview

All $\{\text{PK}_\theta\}_{for\ all\ \theta}$ are public, while $\{\text{SK}_\theta\}_{for\ all\ \theta}$ are private. We denote Δt as the upper bound overhead of one round of communication when sharing data among participants.

Therefore, the global parameters of ABERand are GP = $\{k,\ \mathbb{G}_1,\ g1,\ \mathbb{G}_2,\ g2,\ \text{H},\ F,\ \text{T},\ t,\ n,\ \Delta\text{t},\ \{PK_\theta\}_{for\ all\ \theta},\ \{nodeId_\theta\}_{for\ all\ \theta}\}$. $g1$, $g2$ are the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

Encrypt and Broadcast Ciphertext. The **Encrypt** algorithm takes as input a message M, an access structure (A, σ) with A $\in \text{Z}^{l\times n}$, the public keys of the relevant authorities, and the global parameters. The function ρ : [l] as $\rho(\cdot)$ = T($\sigma(\cdot)$) to realize the mapping of rows to authorities. The algorithm first creates vectors $\boldsymbol{\upsilon} = (z, \upsilon_2, \cdots, \upsilon_n)^\perp$ and $\boldsymbol{\omega} = (0, \omega_2, \cdots, \omega_n)^\perp$, where $z, \upsilon_2, \cdots, \upsilon_n, \omega_2, \cdots, \omega_n \xleftarrow{R} \mathbb{Z}_p$. We let λ_x denote the share of z corresponding to row x, i.e., $\lambda_x = \boldsymbol{A}_x\boldsymbol{\upsilon}$, and ω_x denote the share of 0, i.e., $\omega_x = \boldsymbol{A}_x\boldsymbol{\omega}$, where $\boldsymbol{A}_x$ is the x-th row of A. For each row x of A, it chooses a random $t_x \xleftarrow{R} \mathbb{Z}_p$. The ciphertext is computed as: [34]

$$C_0 = Me(g,g)^z,$$
$$\{C_{1,x} = e(g,g)^{\lambda_x} e(g,g)^{\alpha_{\rho(x)} t_x},\ C_{2,x} = g^{-t_x},$$
$$C_{3,x} = g^{y_{\rho(x)} t_x} g^{\omega_x},\ C_{4,x} = F(\sigma(x))\}$$

Each epoch starts when all participants begin to generate their own random numbers. We use t_0 to denote the current time. The random number will be the input of the encrypt algorithm. The access structure for the random number is constructed as "t-of-n $\{k$@nodeId$_\theta\}_{for\ all\ \theta}$" so that the output ciphertext can be decrypted when t-out-of-n participants combine their secret keys with attribute k@nodeId$_\theta$ in the future. After generating ciphertext CT_θ privately, all participants broadcast their ciphertexts in the form of the data structure mentioned in section V-B to each other via the peer-to-peer network in the following period Δt.

Generate and Broadcast Keys. The **KeyGen** algorithm takes as input a global identifier GID, the identifier $_\theta$ of the authority, the attribute u, and the authority's secret key and the global parameters. It should be the case that $u \in \text{T}^{-1}(\theta)$, i.e., the attribute is controlled by the specific authority. The algorithm first chooses a random $t \xleftarrow{R} \mathbb{Z}_p$, and then it outputs the secret key: [34]

$$SK_{\text{GID},u} = \{K_{\text{GID},u} = g^{\alpha_\theta} H(\text{GID})^{y_\theta} F(u)^t,$$
$$K'_{\text{GID},u} = g^t\}$$

From t_0 to t_1, all participants have encrypted their random values and shared them. At the time of t_1, participant$_\theta$/authority$_\theta$ starts to generate his/her key with the attribute "k@nodeId$_\theta$" for a known "virtual decryptor" GID. Then, in the next Δt, $SK_{\text{GID},u}$ are exchanged in the form of the data structure mentioned in section V-B among the participants; subsequently, t_2 (= t_1 + Δt). At the moment t_2, as long as more than t-out-of-n of the total nodes behave honestly and have shared their real keys, they can decrypt the valid ciphertexts generated at t_0.

To accelerate the verification and to check out whether the key is valid, we demand he/she to send two extra fields along

with the key, namely "affliated_ct" and "M". "affliated_ct" is a small affiliated ciphertext whose original message is "k" and whose access structure is "1-of-1 k@node$_\theta$". The key in this way could be proved to be valid when used to decrypt the affiliated ciphertext. "M" is the original random number M_θ generated at t_0, so that anyone could verify the decryption with computation complexity $O(1)$.

Decrypt ciphertexts. Let (A, σ) be the access structure of the ciphertext. If the decryptor has the secret keys $\{\mathrm{K}_{\mathrm{GID},\sigma(x)}, K^{'}_{\mathrm{GID},\sigma(x)}\}$ for a subset of rows $\boldsymbol{A}_x$ of A such that (1,0,$\cdots$,0) is in the span of these rows, then for each such row x, he/she invokes the **Decrypt** algorithm by computing the following: [34]

$$C_{1,x} \cdot e(K_{\mathrm{GID},\sigma(x)},\ C_{2,x}) \cdot e(H(\mathrm{GID}),\ C_{3,x}) \cdot e(K^{'}_{\mathrm{GID},\sigma(x)},\ C_{4,x}) = e(g,\ g)^{\lambda_x} e(H(\mathrm{GID}),\ g)^{\omega_x}$$

The decryptor then calculates constants $c_x \in \mathbb{Z}_p$ such that $\sum_x c_x \boldsymbol{A}_x$ = (1,0,$\cdots$,0) and computes:

$$\prod_x (e(g,g)^{\lambda_x} e(H(\mathrm{GID}),g)^{\omega_x})^{c_x} \ = \ e(g,g)^z$$

This is true because $\lambda_x = \boldsymbol{A}_x \boldsymbol{\upsilon}$ and $\omega_x = \boldsymbol{A}_x \boldsymbol{\omega}$, where $(1,0,\cdots,0)\boldsymbol{\upsilon}$ = z and $(1,0,\cdots,0)\boldsymbol{\omega}$ = 0. The message can then be obtained as follows:

$$\mathrm{M} = C_0 / e(g,g)^z$$

After collecting at least t honest participants' keys, every participant can run the **Decrypt** algorithm. Since key_θ is associated with attribute "k@node$_\theta$", with at least t reliably generated keys, the decrypt algorithm runs soundly for everyone.

Generate Random Number. By now, i.e., t_2, the random values $\{M'_\theta\}$ independently generated by participants are obtained with the decrypt algorithm. All participants calculate exclusive-or for these random values and will obtain the same random value for the current epoch of ABERand, i.e., res = $\oplus\{M'_\theta\}$.

All the broadcasted messages are recorded for public verification. The public verification could be done instantly just by comparing res with $\oplus\{M_\theta\}$.

Nodes Join/Leave. ABERand is a flexible protocol that permits participants to join and leave freely. When a participant joins, he/she runs the authority setup algorithm according to the global parameters, keeps the secret key, and broadcasts the public key. Meanwhile, the algorithm collects the counterparts' public keys. Then, the t and n in the global parameters are updated to $t \leftarrow 1+(n+1)/2$, $n \leftarrow n+1$. The new participant will participate in the next epoch to distribute his/her random value, which will finally be used in the XOR calculation. When a node in the system leaves, if the time of leaving is in $t_0 \sim t_1$, his/her random value will be not included in current epoch's XOR calculation; if the time of leaving is in $t_1 \sim t_2$, it will have no impact on the final randomness of the current epoch. Then set $t \leftarrow 1+(n-1)/2$, $n \leftarrow n-1$.

Optimization. As mentioned in the above subsections, the generating process of distributed randomness is divided into a series of steps, i.e., $t_0 \sim t_1$, and $t_1 \sim t_2$. Although each step is dependent on the previous steps, two continuous randomness occurrences are independent. Specifically, with pipelining optimization, ABERand can produce randomness with any small interval. Pipelining does not reduce the time that it takes to generate a random number, but it increases the number of steps that can be processed simultaneously and eliminates the delay between the generation of two random values, thus increasing throughput. Figure 2 shows how to optimize the randomness beacon in a pipelining way. In the pipelining design, every random number is delivered by the interval Δt' which is much smaller than Δt, instead of 2Δt.

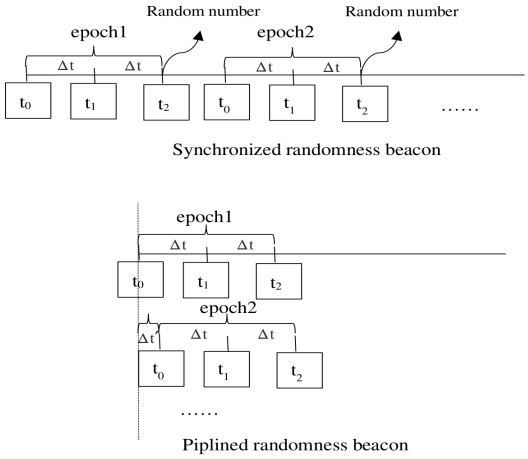

Fig. 2. Optimize randomness beacon in a pipelining way

Pseudocode. Algorithm 1 "ABERand" demonstrates how one node participant in generating distributed verifiable randomness beacon.

VI. Implementation and Evaluation

Framework. We implemented decentralized threshold MA CP-ABE in Charm [48], which is a framework for constructing cryptographic schemes and protocols. It is written in Python language. The Charm framework relies on the gmp (GNU multiple precision) arithmetic library and the PBC (pairing-based cryptography) library written in C language. We implemented the peer-to-peer network in which "ip:port" was used to represent a node. All our experiments were executed on 8 cores of an Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00 GHz with 32 GB RAM running Linux Ubuntu 4.10.0 and Python 3.6.7.

Implementation Details. We chose supersingular symmetric EC group "SS512", which is an asymmetric group in our experiment. Namely, we had three groups, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, and the pairing e was a function from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$. The GID was set to the epoch k as a virtual decryptor. We initiated from 2 to 24 nodes to test the randomness beacon efficiency, in which nodes were full connected. The nodes in the network mainly responded to "NODECONNECTED", which represented a new connection event, and "NODEMESSAGE", which informed the node that new data are arriving.

Algorithm 1 ABERand on a node$_\theta$

procedure ABERAND
  public_key$_\theta$, *private_key*$_\theta$ = *AuthSetup()*
  broadcast *public_key*$_\theta$
  for all $public\_key \in all\ received$ *public_key* **do**
    pass $\leftarrow$ check signature of $public\_key$
    if *pass* **then**
      $public\_key$.append($public\_key$)
  while true **do**
    CT$_\theta$ $\leftarrow$ *Encrypt(RANDOM, policy, public_keys)*
    broadcast *CT*$_\theta$
    for all $ct \in all\ received$ *CTs* **do**
      pass $\leftarrow$ check signature of ct
      if *pass* **then**
        CTs.append(ct)
    KEY$_\theta$ $\leftarrow$ *KeyGen(public_keys, attributes)*
    broadcast *KEY*$_\theta$
    for all $key \in all\ received$ *KEY* **do**
      pass $\leftarrow$ check signature of key
      if *pass* **then**
        $KEYs$.append(key)
    for all $CT \in CTs$ **do**
      RANDOM $\leftarrow$ *Decrypt(CT, KEYs)*
      RANDOMs append *RANDOM*
    random $\leftarrow \oplus\{RANDOMs\}$
    broadcast and record *random*

A. Properties of ABERand

Liveness/Availability: In the commit-and-reveal system with decentralized MA CP-ABE, the protocol successfully produces the final random output accurately. Since there are a sufficient number of honest parties, the (t, n)-threshold decryption ensures the enforcement 6 and guarantees the finality of an epoch. With an honest majority, all parties join in a commit-and-reveal game epoch after epoch, thus ensuring the availability of ABERand.

Bias resistance: Hash or public-key encryption can also be used in commit-and-reveal schemes in a distributed system to lock value in the commit phase. However, a rushing adversary who tries to speak the last may bias the result by revealing or withholding its original value in the reveal phase. On the contrary, adversaries cannot bias the value of the random output with the property of enforcement 6 and property of XOR calulation II-E. If f ($< n/2$) malicious parties try to modify or abort the protocol, they can only gain parts of the inputs for the XOR operation. If crashed participants fail to participate or malicious adversaries deliberately broadcast bad ciphertext in the commit phase, the original random of them are omitted. No matter the values of malicious adversaries are gained in advance or abandoned in the reveal phase, the final random output represents an unbiased, uniformly random value with at least one honest random input.

Unpredictability: There is an upper bound-time δt of the network. In a commit-and-reveal scheme, for a committer, its committed value can be known only after the reveal phase. All committed values are locked and tamper-resistant. Once the commitment is open, all values are obtained irreversibly, including the ones of the adversaries. Only after authorities generate keys in the reveal phase can nodes start to decrypt and get the original value. No party can learn anything about the final random output before the generated keys are broadcast. In our commit-and-reveal scheme, the property of unambiguity makes the original values definite, and the property of privacy scheme makes the original values unpredictable. What's more, everyone could checkout whether another one is a malicious one or not according to the decryption result.

Public verifiability: Public distributed randomness beacon construction must be able to garner public belief in its fairness. When having got the final randomness, any third party must be able to verify that it is correctly generated with $> t + 1$ correct keys. The broadcasted data are public, which can be used to replay the protocol and verify the result. If an adversary can forge the verification process, the adversary must have owned t-out-of-n of the secret keys in the system, violating the assumption that at most f ($< n/2$) nodes are controlled by the adversary. Therefore, any party could verify the legality of the random output with computation complexity $O(1)$ in ABERand and $O(n)$ in ABERand$'$.

Flexibility: ABERand considers the situation that nodes join or leave. In a public-key cryptography system, such as Bitcoin and Ethereum, nodes join or leave without affecting others whose communication complexity is $O(1)$. In PVSS or DKG ones, nodes have to negotiate with each other to re-run PVSSS or DKG, which is at least communication complexity of $O(n^2)$. While in our protocol, nodes are decentralized with respective long-term key pair. Other nodes only need to be notified with the node which is joining or leaving. The join/leave event that happens in the connected network costs communication complexity of $O(n)$, and other nodes hardly need to do anything except for updating t and n.

B. Performance Analysis

For a distributed randomness beacon, the frequency of producing a random value is critical. We mostly focus on the time consumption in our experiment and analyze the bottleneck parts that prevent our protocol from speeding up. We first provide a chart depicting the overhead of the decentralized threshold multi-authority CP-ABE scheme. Then, we implement the scheme in a real peer-to-peer network environment and provide charts describing the overhead of each step in our protocol. Finally, we analyze the experimental data and draw conclusions.

Figure 3 shows that the encryption and key generation time cost increase linearly with the number of attribute authorities. Figure 4 shows the decryption time cost comparison between $t = n/2$ and $t = n * 2/3$. Figure 5 shows that the ciphertext size increases linearly with the number of attribute authorities. All of the above are tested on one core of the CPU group. Figure 6 shows the histogram of the comparison of the time cost of different steps in ABERand. We can conclude that

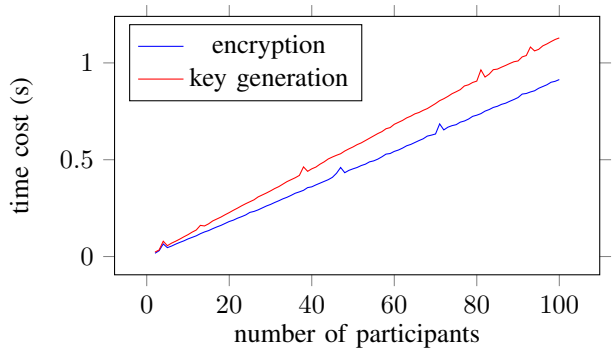

Fig. 3. Encryption and key generation cost

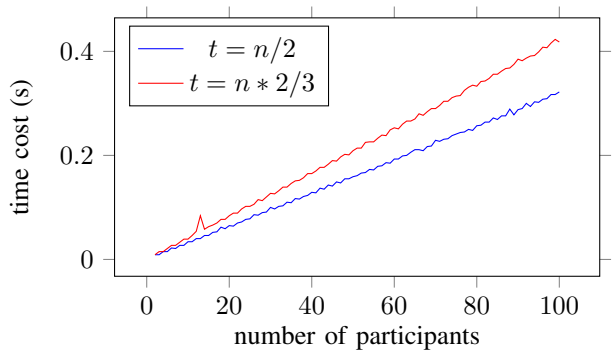

Fig. 4. Decryption cost

the "Dec" is the most time-consuming step in ABERand if ciphertexts are decrypted one after one. However, these decryptions can be done simultaneously if a node has enough computation resources.

When the group size is 100, a node in an epoch produces ∼8MB data which could be further compressed. To get the final beacon value, it takes ∼0.9s to encrypt, ∼1.1s to generate key and ∼0.3s × 100 to decrypt. To optimize, the decryptions could be done in parallel. Because of limited system resources, deviation exists when conducting experiments in a fully connected network. To test ABERand, we prepare 2 running environments, i.e., 2 to 24 nodes run on one physical machine and on 6 physical machines. All 8 cores of the CPU groups of a machine are used in the above two situations.

Figure 7 provides the results to compare the frequency of the distributed randomness in the 2 different environments. It is evident that when the number of participants is $\leq$ 8 (8 is the number of cores of a machine), the overhead of ABERand in the two situations is nearly the same. It also implies that performance is much better if one machine runs only one node. Figure 8 shows the average CPU usage of a machine in the 2 different environments.

Since original random number is along with key in "Generate and Broadcast Keys" phase, see more in V-C. An exciting optimization is that the original random number is directly used for the final randomness and decryption is only used for verification. Thus, the computation complexity is O(1),

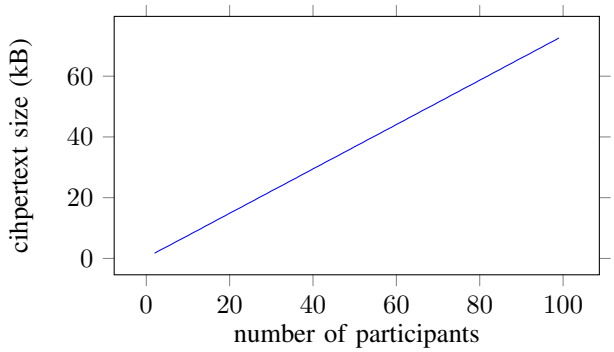

Fig. 5. Ciphertext sizes against number of AAs

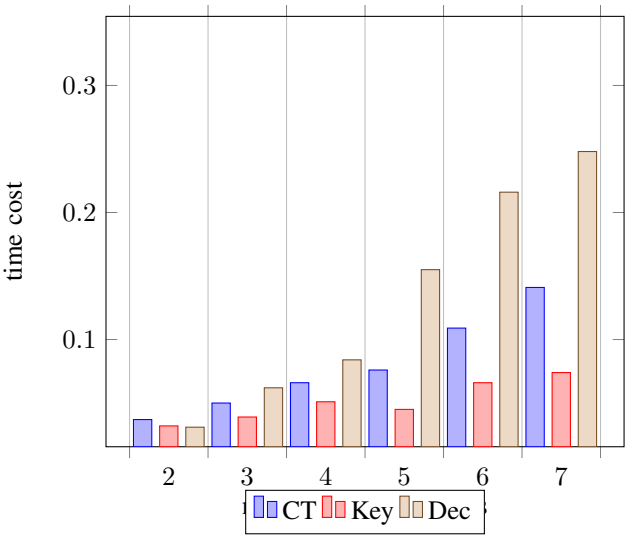

Fig. 6. Cost of different steps with different nodes
CT: cost of generating and broadcasting cihpertexts;
Key: cost of generating and broadcasting keys;
Dec: cost of decryptions.

and the verification complexity is O(n). We call this protocol ABERand′. Figure 9 and figure 10 depict the performance of ABERand′, and the results show great improvement compared to figure 7 and figure 8.

For a participant, ABERand receives $n-1$ ciphertexts and $n-1$ keys and performs n times of decryptions. In the broadcast environment, one node receives data from $n-1$ counterparts, and a ciphertext size is $O(n)$, so the total communication overhead is (n^3), and the one node computation overhead is $O(n)$. Decryptions are dense for a node, but could be calculated in parallel. With multiple nodes on one physical machine, there is no much difference using pipelining methods in our testing experiment.

C. Comparison

With Hash-Based or PoW-Based. Hash-Based method is a basic idea to implement the commit-and-reveal protocol to resist tampering. However, it has the "last actor" problem. When a participant receives all counterparts' original ran-

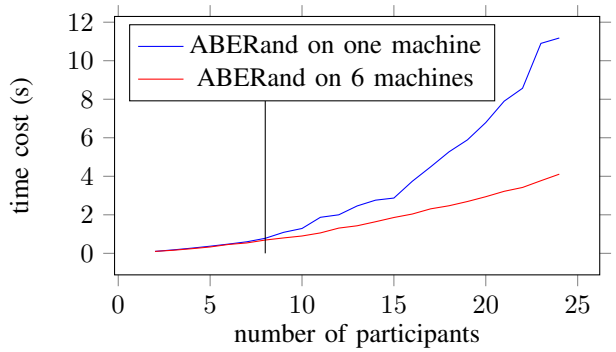

Fig. 7. Total cost of ABERand randomness

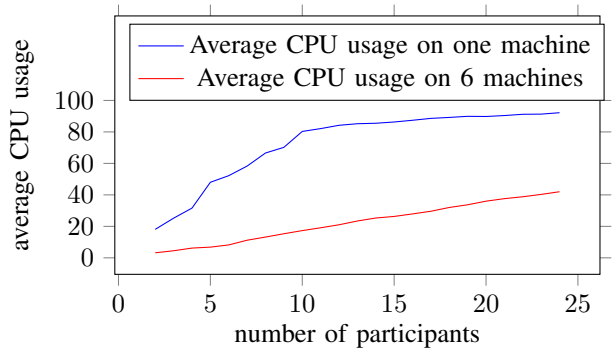

Fig. 8. CPU usage of ABERand randomness

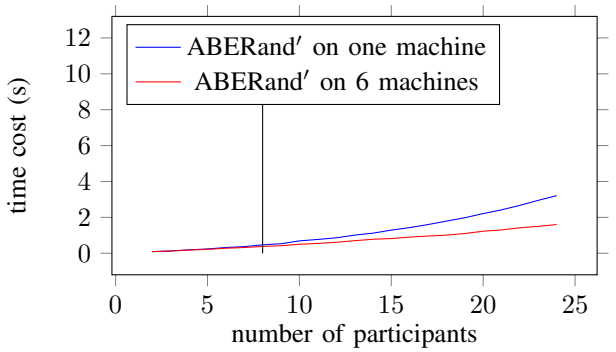

Fig. 9. Total cost of ABERand′ randomness

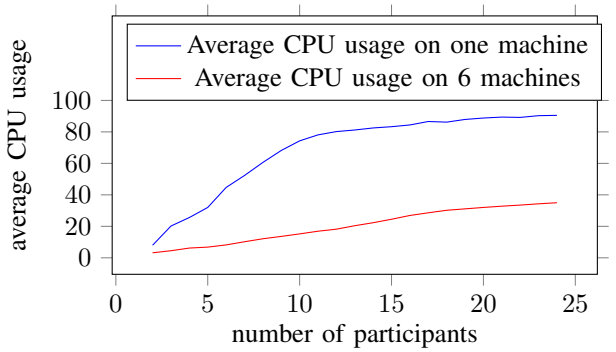

Fig. 10. CPU usage of ABERand′ randomness

doms, he/she could compute the final beacon results. If the result is little of an advantage to him/her, he/she could abort the commit-and-reveal protocol, which is fatal to distributed randomness beacons. ABERand achieves the commit-and-reveal protocol which has no "last actor" problem because of threshold cryptography. Iterated hash [17] or verifiable delay function [26] suffers from the problem of how to set a specific time to delay. It has to rely on a proof-of-work blockchain to produce a distributed randomness beacon, which may bias the result as described in the previous part. The consensus algorithm and a standalone distributed randomness are somewhat chicken and egg situation. ABERand is a standalone protocol that could enhance the ability of a blockchain rather than rely on it.

With PVSS-Based. PVSS-based projects [5], [16], [19] are mainly pragmatic and complex. In PVSS-Based methods, computation and verification are intensive and costly. The communication complexity of most of them are $O(n^3)$. RandHound and RandHerd [16] use PVSS and other cryptographic primitives, such as BFT and Collective Signature [28] to support scalability. They lowered communication complexity to $O(c^2 \cdot n)$ and $O(c^2 \cdot logn)$ where c is the group size. However, they have the "self-DoS" problem and a high faulty probability of 0.08%. Scrape [19] lowered PVSS to communication complexity of $O(n^2)$, which uses a public bulletin board. Hydrand [13], which makes use of Scrape's protocol, lowered communication complexity to $O(n^2)$ by running one PVSS in a round. It has to wait for $f+1$ rounds to produce a distributed randomness in Hydrand, and all nodes have to be "online" which is hardly possible in a practical peer-to-peer system. Hydrand has a strong assumption that the number of participants is fixed and it ignores the "flexibility" property of a beacon. ABERand permits nodes join or leave with communication complexity of $O(n)$ and could produce randomness simultaneously with parallel running epochs. In addition, ABERand has no DoS attack and solves the "last actor" problem which both may lead bias to a beacon. In a word, ABERand is more flexible, bias-resistant and effective.

With BLS-Based. Dfinity [21] combines DKG and BLS to produce distributed randomness. DKG requires the communication complexity of at least $O(n^2)$ to share secrets securely. When f evil nodes exist, f "complaints" [33] will be broadcasted which will delay the protocol, and the worst communication complexity of running DKG is $O(n^2 \cdot f)$. For this reason, in our opinion, to lower the communication cost and improve efficiency, participants are divided into small groups by threshold relay protocol [21]. Groups in Dfinity[5] are fixed in each epoch, which brings the faulty probability of 10^{-17}. Parameters, such as group size, epoch interval, may further affect the availability when members leave since it will take long (e.g., one week) to update a group. DKG of Dfinity uses Joint-Feldman verifiable secret sharing [33], which was proved to be insecure [33] in some situations. The security of such a system (DKG + tBLS) needs to be more

Protocol	Availability/ Faulty prop.	Comm. Complexity	Unpredi-ctability	Bias-resistant	Comp. complexity	Add/remove participant	Verification Comp.	Trusted dealer	DKG	Byzantine nodes
Algorand	1e-12	O(cn)	√	×	O(c)	$O(n^2)$	O(1)	×	×	< n/3
Dfinity	1e-17	O(cn)	√	√	O(c)	$O(n^2)$	O(1)	×	√	< n/2
Ouroboros	√	$O(n^3)$	√	√	$O(n^3)$	$O(n^3)$	$O(n^3)$	×	×	< n/2
Scrape	√	$O(n^3)$	√	√	$O(n^2)$	$O(n^2)$	$O(n^2)$	×	×	< n/2
RandHound	0.08%	$O(c^2 n)$	√	√	$O(c^2 n)$	$O(n^2)$	$O(c^2 n)$	×	×	< n/3
RandHerd	0.08%	$O(c^2 \log n)$	√	√	$O(c^2 \log n)$	$O(n^2)$	O(1)	√	×	< n/3
PoW	√	O(n)	√	×	very high	O(1)	O(1)	×	×	< n/2
Hydrand	√	$O(n^2)$	√	√	O(n)	very high	O(n)	×	×	< n/3
ABERand	√	$O(n^3)$	√	√	O(n)	O(n)	O(1)	×	×	< n/2
ABERand′	√	$O(n^3)$	√	√	O(1)	O(n)	O(n)	×	×	< n/2

Fig. 11. Comparisons based on Hydrand's [13]

The "Availability/Faulty prop." column shows the result whether it is bias-resistant. The "Comm. Complexity" column gives the total network communication cost. The "Unpredictability" column shows whether the final result of an epoch can be predicted. The "Comp. complexity", "Add/remove participant" and "Verification Comp." columns show the computation cost in one node with one processor. The "Trusted dealer" and "DKG" columns show whether a trusted dealer and DKG is needed, respectively. The "Byzantine nodes" column gives the permitted maximum amount of malicious nodes.

discussed. Compared to BLS-based protocol, ABERand is a standalone randomness beacon independent of any blockchain or public bulletin board. Multiple epochs in ABERand could generate randomness simultaneously. Most importantly, no DKG protocol is needed, and a participator's public key and private key pair are once and for all.

When considering adding/removing participants, the participants in PVSS-based or BLS-based methods have to re-run the PVSS or DKG protocol, both of which need secret sharing and cost at least $O(n^2)$. ABERand uses a decentralized threshold MA CP-ABE scheme whose ciphertext already contains secret sharing (LSSS). When the CP-ABE scheme is combined with a commit-and-reveal scheme, a ciphertext not only contains "shares" [27] but also acts as "commitment". Figure 11 gives a detailed comparison of various distributed randomness beacon mentioned in section I.

In a word, our construction is more simpler, flexible, bias-resistant and effective, because CP-ABE uses space to exchange time to lower complexity by hiding secret sharing in ciphertexts.

VII. Possible Attacks and Security

Possible attacks on ABERand are analyzed below, which proves that ABERand is a secure and sound protocol.

A. Collusion Attack

To prevent collusion attacks, we prove that ABERand is secure in two aspects.

On the one hand, from the perspective of the MA CP-ABE scheme [34], the global identities are "tied" together with the various attributes that belong to a specific user so that they cannot be successfully combined with another user's attributes in the decryption. More specifically, the encryption algorithm blinds the message M with $e(g_1, g_1)^s$, where g_1 is a generator of the subgroup G_{p_1}, and s is a randomly chosen value in $\mathbb{Z}_N$. The value s is then split into shares λ_x according to the LSSS matrix, and the value 0 is split into shares ω_x. The decryptor must recover the blinding factor $e(g_1, g_1)^s$ by pairing their keys for the attribute and identity pairs (i, GID) with the ciphertext elements to obtain the shares of s. In doing so, the decryptor will introduce terms of the form $e(g_1, H(\mathrm{GID}))^{\omega_x}$. If the decryptor has a satisfying set of keys with the same identity GID, these additional terms will be canceled from the final result since the ω_xs are shares of 0. If two users with different identities GID and GID′ attempt to collude and combine their keys, then there will be some terms of the form $e(g_1, H(\mathrm{GID}))^{\omega_x}$ and some terms of the form $e(g_1, H(\mathrm{GID}'))^{\omega_{x'}}$, and these will not cancel with each other, thereby preventing the recovery of $e(g_1, g_1)^s$. In summary, keys in different epochs of ABERand are one-shot and could not be combined.

On the other hand, according to the ABERand protocol specification, the number of malicious parties $f < n/2$. Therefore, in the "worst case", these malicious parties collude with each other in advance to decrypt the ciphertexts. Honest parties abide by the protocol and set the threshold t, which is higher than $n/2$. Since the threshold $t > f$, f malicious parties fail to decrypt ciphertexts even if they share their keys in advance, thus preventing collusion when producing distributed randomness.

B. Attacks on ABERand

An epoch of ABERand has four stages in general when considering the interaction among nodes. Participants join or leave at the 1st stage; Participants encrypt a random value and broadcast ciphertexts of their randoms at the 2nd stage; Participants generate and broadcast keys at the 3rd stage;

Participants compute the final randomness at the 4th stage. There is a timeout Δt for the broadcasting stage.

Join/Leave Attack: When a participant joins after the 1st stage, other nodes increase t and n by 1 and the new joint one will participate in the next epoch. If a participant leaves at the before the 2nd stage, it contributes none to the final randomness of the current epoch; If it leaves at later stages, its original randomness will be obtained through the decryption by the honest ones. It neither bias the result to any one's favor because of XOR operation II-E nor aborts the whole process because of the enforcement of our protocol 6.

Access Structure Attack: Malicious participants may use a different access structure in the 2nd stage, thus producing an illegal ciphertext. No matter whether the ciphertext could be decrypted by more than $f+1$ honest ones, it will be abandoned for the current epoch of the beacon.

KeyGen Attack: In the 3rd stage, since participant θ sends a ciphertext whose original message is "k" and whose access structure is "1-of-1 k@node$_\theta$". Others could check out whether the key is valid through decrypting the ciphertext. If the key is failed to decrypt the associated ciphertext, it will not be used in the decryption step. The original random value of this participant is abandoned without affecting the final result because of II-E. In addition, the one who provides a failure key will be exposed to everyone. With an honest majority, the whole process will go on.

Message(Ciphertext) Attack: In broadcasting stages, messages can be verified to be untampered with public key and signature in the messages. Nodes are connected so that no malicious one could cheat them by broadcasting different ciphertexts. Any malicious participant could send a forged ciphertext, thus leading to failure of decryption at the 3rd stage. It is a situation the same as Join/Leave Attack in the 1st stage.

Public Key Attack: If a malicious participant joins and broadcasts a fake public key, which is similar to the situation that a participant loses his/her private key. Without a private key, the participant will not produce the correct key in the 3rd stage. It is a situation the same as Join/Leave Attack in the 1st stage.

Randomness Attack: Malicious participants may try colluding to decrypt the honest parties$'$ ciphertext in advance; however, they will fail because f is less than $n/2$. Moreover, they cannot stop honest ones from obtaining randomness in each epoch under the commit-and-reveal scheme using decentralized threshold MA CP-ABE.

DoS Attack: Malicious participants at any stage, could deny obeying the protocol. If they initiate the DoS attack at the 1st and 2nd stage, it is a situation the same as Join/Leave Attack; If they attack at later stages, it is a situation the same as Message Attack or KeyGen Attack.

Sybil Attack: As mentioned in "system and threat model" section III, nodes are authenticated to participate and generate distributed randomness. With "nodeId" as each node's public unique identity, all nodes are registered and recognized in our protocol. No adversary could create a majority of colluding nodes ($f >= n/2$) in our setting to start a Sybil attack.

VIII. Concluding remarks

In this paper, we adapt Rouselakis's MA CP-ABE to the decentralized (t, n)-threshold MA CP-ABE approach, in which t-out-of-n participants generate keys for decryption, and propose a flexible distributed randomness beacon, named ABERand. The decentralized (t, n)-threshold MA CP-ABE approach is used as a commit-and-reveal scheme. ABERand extends the commit-and-reveal scheme to obtain property of enforcement 6. CP-ABE hides secret sharing in ciphertext which results in lower complexity and high flexibility.

To further lower the communication cost and to improve the performance in a real production environment, sharding [49] is a direction from an engineering point and constant size ciphertext ABE [50], [51] could be adopted from a theoretical view for future job.

Notes

[1]http://en.wikipedia.org/wiki/1980_Pennsylvania_Lottery_scandal
[2]https://www.nist.gov/programs-projects/nist-randomness-beacon
[3]https://en.wikipedia.org/wiki/Lavarand
[4]https://en.wikipedia.org/wiki/Commitment_scheme
[5]https://dfinity.org/pdf-viewer/pdfs/viewer?file=../library/threshold-relay-blockchain-stanford.pdf

References

[1] M. Blum. Coin flipping by telephone. In Allen Gersho, editor, CRYPTO '81, volume ECE Report 82-04, pages 11–15. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981.

[2] M. O. Rabin. Transaction protection by beacons. J. Comput. Syst. Sci., 27(2):256–267, 1983.

[3] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[4] Garay J., Kiayias A., Leonardos N. (2015) The Bitcoin Backbone Protocol: Analysis and Applications. In: Oswald E., Fischlin M. (eds) Advances in Cryptology - EUROCRYPT 2015. EUROCRYPT 2015. Lecture Notes in Computer Science, vol 9057. Springer, Berlin, Heidelberg.

[5] Kiayias, A., Konstantinou, I., Russell, A., David, B.M., Oliynykov, R. (2016). A Provably Secure Proof-of-Stake Blockchain Protocol. IACR Cryptology ePrint Archive, 2016, 889.

[6] S. Azouvi, P. McCorry, and S. Meiklejohn. Winning the caucus race: Continuous leader election via public randomness. arXiv preprint arXiv:1801.07965, 2018.

[7] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. white paper, 2014.

[8] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In 2016 IEEE Symposium on Security and Privacy, pages 839–858. IEEE Computer Society Press, May 2016.

[9] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. On bitcoin as a public randomness source. Cryptology ePrint Archive, Report 2015/1015, 2015. http://eprint.iacr.org/2015/1015.

[10] M. Milutinovic, W. He, H. Wu, M. Kanwal, "Proof of luck: An efficient blockchain consensus protocol", Proc. 1st Workshop Syst. Softw. Trusted Execution (SysTEX), pp. 1-6, 2016.

[11] King, S., Nadal, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. 2012

[12] Albert Kwon, David Lu and Srinivas Devadas. XRD: Scalable Messaging System with Cryptographic Privacy. Cryptology ePrint Archive, 2019. http://arxiv.org/abs/1901.04368.

[13] Schindler, Philipp, Aljosha Judmayer, Nicholas Stifter and Edgar R. Weippl. HydRand : Efficient Continuous Distributed Randomness. (2019).

[14] Thomas Baignères and Cécile Delerablée and Matthieu Finiasz and Louis Goubin and Tancrède Lepoint and Matthieu Rivain. Trap Me If You Can – Million Dollar Curve. (2015)

[15] I. Bentov, A. Gabizon, and D. Zuckerman. Bitcoin beacon. https://arxiv.org/pdf/1605.04559v2, 2016. Accessed: 2016-06-06.

[16] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford. Scalable Bias-Resistant Distributed Randomness. In Security and Privacy (SP), 2017 IEEE Symposium on, pages 444–460. IEEE, 2017. Accessed: 2017-08-20.

[17] A. K. Lenstra and B. Wesolowski. A random zoo: sloth, unicorn, and trx. Cryptology ePrint Archive, Report 2015/366, 2015.

[18] B. Bunz, S. Goldfeder, and J. Bonneau. Proofs-of-delay and randomness beacons in Ethereum. In S&B '17: Proceedings of the 1st IEEE Security & Privacy on the Blockchain Workshop, April 2017. Accessed: 2017-08-21.

[19] I. Cascudo and B. David. Scrape: Scalable randomness attested by public entities. http://eprint.iacr.org/2017/216, 2017. Accessed: 2017-03-24.

[20] Yossi Gilad and Rotem Hemo and Silvio Micali and Georgios Vlachos and Nickolai Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. (2017)

[21] Timo Hanke, Mahnush Movahedi, and Dominic Williams. DFINITY Technology Overview Series, Consensus System. CoRR, Vol. abs/1805.04548 (2018). arxiv: 1805.04548 http://arxiv.org/abs/1805.04548

[22] Jeremy Clark and Urs Hengartner. On the use of financial data as a random beacon. Proceedings of the 2010 international conference on Electronic voting technology/workshop on trustworthy elections, p.1-8, August 09-10, 2010, Washington, DC

[23] S. Bag, S. Ruj and K. Sakurai, "Bitcoin Block Withholding Attack: Analysis and Mitigation," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 8, pp. 1967-1978, Aug. 2017. doi: 10.1109/TIFS.2016.2623588

[24] C. Pierrot and B. Wesolowski (2016). Malleability of the blockchain's entropy. Cryptography and Communications, 10, 211-233.

[25] S. Popov (2016). On a decentralized trustless pseudo-random number generation algorithm. IACR Cryptology ePrint Archive, 2016, 228.

[26] Dan Boneh, Joseph Bonneau, Benedikt Bunz, and Ben Fisch. Verifiable delay functions.In CRYPTO 2018, 2018.

[27] Schoenmakers B. (1999) A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting. In: Wiener M. (eds) Advances in Cryptology — CRYPTO' 99. CRYPTO 1999. Lecture Notes in Computer Science, vol 1666. Springer, Berlin, Heidelberg

[28] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford. Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning. In 37th IEEE Symposium on Security and Privacy, May 2016.

[29] S. Micali. (2017). Byzantine Agreement , Made Trivial.

[30] Micali, Silvio, Michael O. Rabin and Salil P. Vadhan. "Verifiable Random Functions." FOCS (1999).

[31] Boneh D., Lynn B., Shacham H. (2001) Short Signatures from the Weil Pairing. In: Boyd C. (eds) Advances in Cryptology — ASIACRYPT 2001. ASIACRYPT 2001. Lecture Notes in Computer Science, vol 2248. Springer, Berlin, Heidelberg

[32] Pedersen T.P. (1991) A Threshold Cryptosystem without a Trusted Party. In: Davies D.W. (eds) Advances in Cryptology — EUROCRYPT '91. EUROCRYPT 1991. Lecture Notes in Computer Science, vol 547. Springer, Berlin, Heidelberg

[33] Gennaro R., Jarecki S., Krawczyk H., Rabin T. (1999) Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In: Stern J. (eds) Advances in Cryptology — EUROCRYPT '99. EUROCRYPT 1999. Lecture Notes in Computer Science, vol 1592. Springer, Berlin, Heidelberg

[34] Rouselakis Yannis, Waters Brent. (2015). Efficient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption. IACR Cryptology ePrint Archive. 2015. 315-332. 10.1007978-3-662-47854-7_19.

[35] H. Lin, Z. Cao, X. Liang, and J. Shao. Secure threshold multi authority attribute based encryption without a central authority. In INDOCRYPT, pages 426–436, 2008.

[36] A. Beimel. Secure schemes for secret sharing and key distribution. Ph.D. dissertation, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[37] J. R. Douceur. The Sybil attack. In 1st International Workshop on Peer-to-Peer Systems (IPTPS), Mar. 2002.

[38] Liu, Z. J. et al. Efficient Generation of Linear Secret Sharing Scheme Matrices from Threshold Access Trees. (2014).

[39] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. in Proc. 13th ACM Conf. Comput. Commun. Security, 2006, pp. 89–98.

[40] N. Attrapadung, B. Libert, and E. Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. in Proc. 14th Int. Conf. Practice Theory Public Key Cryptography, 2011, pp. 90–108.

[41] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. in Proc. IEEE Symp. Security Privacy, 2007, pp. 321–334.

[42] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. in Proc. 14th Int. Conf. Practice Theory Public Key Cryptography, 2011, pp. 53–70.

[43] V. Goyal, A. Jain, O. Pandey, and A. Sahai. Bounded ciphertext policy attribute based encryption. in Proc. 35th Int. Colloquium Automata, Lang. Programm., 2008, pp. 579–591.

[44] R. Bobba, H. Khurana, and M. Prabhakaran. Attribute-sets: A practically motivated enhancement to attribute-based encryption. in Proc. 14th Eur. Symp. Res. Comput. Security, 2009, pp. 587–604.

[45] Oded Goldreich. Foundations of Cryptography, volume II. Cambridge University Press, 2004.

[46] Li W, Xue K, Xue Y, Hong J. (2015). TMACS: a robust and verifiable threshold multi-authority access control system in public cloud storage. IEEE Trans Inf Forensics Secur 10(1):55–68

[47] Lewko A, Waters B (2011). Decentralizing attribute-based encryption. In: Advances in cryptology–EUROCRYPT 2011, Springer, NewYork, pp 568–588

[48] Joseph A. Akinyele, Matthew Green, and Avi Rubin. Charm: A framework for rapidly prototyping cryptosystems. Cryptology ePrint Archive, Report 2011/617, 2011. http://eprint.iacr.org/.

[49] Loi Luu , Viswesh Narayanan , Chaodong Zheng , Kunal Baweja , Seth Gilbert , Prateek Saxena, A Secure Sharding Protocol For Open Blockchains, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, October 24-28, 2016, Vienna, Austria [doi: 10.1145/2976749.2978389]

[50] Doshi Nishant, and DeveshJinwala, "Constant ciphertext length in multi-authority ciphertext policy attribute based encryption.", In Computer and Communication Technology (ICCCT), 2011 2nd International Conference on, pp. 451-456, 2011.

[51] Zhang, Y., Li, J., Yan, H.: Constant size ciphertext distributed CP-ABE scheme with privacy protection and fully hiding access structure. IEEE Access 7, 47982–47990 (2019)