

# DSMIX: A Dynamic Self-organizing Mix Anonymous System

Renpeng Zou\*, Xixiang Lv\*,

\*School of Cyber Engineering, Xidian University, Xian 710071, China

**Abstract**—Increasing awareness of privacy-preserving has led to a strong focus on anonymous systems protecting anonymity. By studying early schemes, we summarize some intractable problems of anonymous systems. Centralization setting is a universal problem since most anonymous system rely on central proxies or presetting nodes to forward and mix messages, which compromises users' privacy in some way. Besides, availability becomes another important factor limiting the development of anonymous system due to the large requirement of additional additional resources (i.e. bandwidth and storage) and high latency. Moreover, existing anonymous systems may suffer from different attacks including abominable Man-in-the-Middle (MitM) attacks, Distributed Denial-of-service (DDoS) attacks and so on. In this context, we first come up with a BlockChain-based Mix-Net (BCMN) protocol and theoretically demonstrate its security and anonymity. Then we construct a concrete dynamic self-organizing BlockChain-based MIX anonymous system (BCMIX). In the system, users and mix nodes utilize the blockchain transactions and their addresses to negotiate keys with each other, which can resist the MitM attacks. In addition, we design an IP sharding algorithm to mitigate Sybil attacks. To evaluate the BCMIX system, we leverage the distribution of mining pools in the real world to test the system's performance and ability to resistant attacks. Compared with other systems, BCMIX provides better resilience to known attacks, while achieving low latency anonymous communication without significant bandwidth or storage resources.

**Index Terms**—Anonymous systems, blockchain, anonymity, self-organizing, mix network attacks.

## I. INTRODUCTION

KEEPING communication private has become increasing important in an era of mass surveillance and carriers-sponsored attacks. Recently, many events about private data leakage in Online Social Networks (OSNs) [1], mobile network service (Uber, Didi Chuxing) [2] and telephone communications [3] have occurred frequently. Thus it is often the case that two parties want to communicate anonymously, which means to exchange messages while hiding the fact that they are in conversation.

In this context, the anonymous communication technology emerges as a critical topic. Aiming to preserve communication privacy within the shared public network environment, anonymous communication mainly focus on how to hide the identities or address information of one side or both sides in communications. Since the seminal work by Chaum [4] for anonymous communication, more than seventy anonymous systems have been proposed, based on different anonymous mechanism [5]. Generally, anonymous systems can be divided into the following sub-types, mix re-encryption, multicast/broadcast, mix multi-layer encryption and peer-to-peer.

There are, however, concerns about the lack of efficiency and security in anonymous systems [6], [7], as explained below:

- **Centralization.** Most of anonymous systems, i.e. Anonymizer [8], LPWA [9] and cMix [10], rely on central proxies or preset nodes to hide the address information, then still use these proxies to blind and forward messages. The centralized anonymous systems bring privacy leakage risks to users since the service providers can control all proxies and mix node to infer the users' identities. What's worse, the public proxies or preset nodes are easily exposed to attackers. For instance, an attacker can launch distributed denial-of-service attacks to block one or more proxies and thus crash the system.
- **Availability.** Efficiency and additional resources are the main factors affecting the availability of anonymous systems. High-latency anonymous systems such as OneSwarm [11] and A3 [12], though provide high anonymity, are not well suited for practical utilization because of the poor efficiency in terms of intolerable latency [11]. In order to hide the identities of the recipients, multicast/broadcast and peer-to-peer based anonymous systems consume more bandwidth resources to cover the normal traffics [13]. In spite of effectiveness, users may not be willing to contribute a lot of bandwidth, which hinders the development of such anonymous systems. In addition, some anonymous systems, such as cMix, adopt additional inspection schemes to identify the malicious nodes, which would place an additional burden on users and decrease message utilization.
- **Security.** Along the research line about security, anonymous systems based on different mechanism may suffer from different security issues. Some anonymous systems based on MIX technologies rely on fixed cascade nodes to mix and forward messages. Such systems are vulnerable to collusion-tagging attacks which is hard to detect. Re-routing or proxy forwarding based anonymous systems, such as Tor [14] and Tarzan [15], deliver messages through nodes or proxies randomly selected from clusters, hence an eavesdropper can perform traffic analysis attacks and destroy the anonymity. As for multicast/broadcast based anonymous system, an attacker can masquerade as the benign recipients to intercept the messages. With respect to the P2P based anonymous systems, an attacker can create multiple identities to launch a Sybil attack, which allows the attacker to analyze the forwarding

path and impersonate the recipient to receive the messages. Moreover, in anonymous systems applying key exchange schemes, an attacker can employ Man-in-the-Middle (MitM) attacks [16] to undermine the security of these systems.

#### A. Solutions and Contributions

Motivated by these identified limitations, we combine blockchain technology with mix network, and design a dynamic self-organizing blockchain-based mix anonymous system. We expect to dispose of the following challenges.

**Challenge 1. Designing a decentralized self-organizing anonymous system.** The first challenge we seek to address is the centralization issues. As we mention above, the central proxies or preset nodes might pry into users' private data and reveal the true identities. Therefore, we intend to construct a decentralized anonymous system in which the mix nodes are dynamic and self-organizing.

**Solution 1.** When it comes to decentralization, the most popular technology is blockchain, an emerging decentralized architecture and distributed computing paradigm underlying Bitcoin [17] and other cryptocurrencies. Leveraging the dynamic and decentralized properties of blockchain miners, we devise voting algorithms to elect mix nodes from blockchain miners. This trustless and distributed design not only prevents privacy leakage from service providers, but also mitigates single point failure and DDoS attacks.

**Challenge 2. Designing a user friendly anonymous system.** Another challenge is to construct an anonymous system with high availability. In fact, users are often reluctant to consume more additional resources (i.e. bandwidth and computing resources) and wait for a long time. Thus we are drove to build a high available anonymous system with less additional resources.

**Solution 2.** Our proposed solution is inspired by cMix. Similar to cMix, we also split the time-consuming, complicated public key operations with the real time phase. The difference is that cMix adopts fixed mix nodes with a stable joint public key while our approach leverages the dynamic blockchain miners to compete for mix nodes, which brings a problem that the elected mix nodes (miners) have to negotiate a joint public key in each round. To avoid interacting with other mix nodes in each round, we firstly propose a revised additive homomorphism mix-net protocol. Then we combine the protocol with Bitcoin account schemes, such that the elected mix nodes can calculate the system public key in a non-interactive manner. It is worth noting that no additional resources are required in BCMIX other than the miners' computing power for solving Bitcoin puzzles.

**Challenge 3. Designing a secure anonymous system.** The most intractable challenge is building a secure anonymous system. Based on different principles, anonymous systems are assailable to disparate attacks including internal attacks and external attacks. The internal attacks , i.e. traffic analysis attacks, DDoS attacks and so on, are caused by the design principles of systems while the external attacks, such as MitM attacks and Sybil attacks, arise from the cryptographic

protocols or other schemes utilized in anonymous systems. In the case, we intend to construct an anonymous system which can resist the attacks mentioned above.

**Solution 3.** Through the former schemes we found mix technology can defend against most kind of attacks on anonymous systems except bring the centralization problem and tagging attacks. To mitigate the weakness, our first consideration is combining blockchain and mix technology, as mentioned in Solution 1. Unfortunately, the introduction of blockchain raises the Sybil attacks into the anonymous system. By researching the properties of Sybil attacks we structure PoW voting and IP sharding algorithms to mitigate the impact of Sybil attacks. For MitM attacks, we design a transaction-based key exchange scheme which makes use of the Bitcoin's transaction propagation mechanism to break the single-channel control of attackers. The detailed illustration is provided in Section V and Section VI.

To summarize, the contributions of this paper are as follows.

- We propose a blockchain-based mix-net protocol (BCMN) whose security and anonymity are demonstrated theoretically. Especially, we elect miners as mix nodes via special algorithms to avoid the centralized settings.
- We construct a dynamic self-organizing blockchain-based mix anonymous system (upon the proposed BCMN protocol) and discuss how the proposal can satisfy the security requirements.
- We demonstrate the feasibility and effectiveness of the proposed BCMIX by developing the system in an analog network with the miner distribution data in the real world. Compared with existing systems, our system performs well and provides stronger security.

#### B. Related Works

Generally, anonymous communication systems can be divided into the following sub-types, mix re-encryption, multicast/broadcast, mix multi-layer encryption and peer-to-peer. Mix re-encryption based anonymous systems [18], [19] leverage cryptography technologies to dispose messages and hide the users' identities, which can resist traffic analysis attacks. But with the utilizing of public key cryptography, mix re-encryption based anonymous systems is expensive and easy to waste resources. To settle the problem, Chaum et.al [10] proposed a anonymous system called cMix in 2017. Through a precomputation, the core cMix protocol eliminates all expensive real time public-key operations at the senders, recipients and mix nodes, thereby decreasing real time cryptographic latency and lowering computational costs for clients. The core real time phase performs only a few fast modular multiplications. The authors claim that cMix can detect the malicious nodes by utilizing Random Partial Checking (RPC) and commitment scheme.

Multicast/broadcast based anonymous communication systems [20], [21] achieve anonymity through one-to-many communications among hosts. This method is expensive and inefficient for non-broadcast networks. In the case of large scale networks, an attacker can easily masquerade as the recipient to intercept the message, which further increases the computation and communication overhead required for authentication.

As for mix multi-layer encryption based anonymous systems [22] [14] [23], one or more proxies are selected from the cluster, and forward the messages from the former nodes and then the messages in a confusing order. The technology can achieve low-latency communications under the premise of ensuring efficiency, but is vulnerable to analyzing attacks such as traffic analysis attacks and sniper attacks [24].

The rapid development of peer-to-peer (P2P) networks drives the research of anonymous communication technology in P2P network environment [25]. In P2P based anonymous systems [26], [27] nodes enjoy anonymous services, and provide anonymous services for other nodes in their spare time. Because the P2P network itself has a high degree of self-organization and disorder, and the number of members is large, the P2P network can also provide a high degree of anonymity when the attacker has a huge attack resource. However, because of its openness and anonymity, the attacker can control a large number of zombie nodes to launch witch attacks, and can disguise as normal nodes for traffic analysis, thereby destroying the system's anonymity without being noticed.

### C. Roadmap

The rest of this paper is organized as follows. In Section II, we review some preliminaries and propose an attacks against cMix. In Section III, we illustrate the security model and requirements. Next, we detail the BCMN protocol and the proposed BCMIX in Section IV. Then we evaluate the performance and demonstrate security respectively in Section V and Section VI. Finally, this paper is concluded in Section VII.

## II. PRELIMINARIES AND PROPOSED ATTACKS ON CMIX

In this section we briefly review the relevant notations and definitions that are used in this paper. Then we describe some attacks against cMix.

### A. Elliptic Curve based Cryptographic Primitives

In this work, we adopt elliptic curves over prime finite field  $\mathbb{F}_p$ . The elliptic curve  $y^2 = x^3 + ax + b$  over  $\mathbb{F}_p$  could be represented as  $E_p(a, b)$ .

**EC-Elgamal.** The analog of ElGamal crypto system based on ECC, which is known as EC-Elgamal, was first introduced in [28]. It consists of the following algorithms.

- **Setup( $1^{\mathcal{K}}$ )**. The algorithm takes as input the security parameter  $\mathcal{K}$ , and outputs the elliptic curve  $E_p(a, b)$  with base point  $G$ .
- **KeyGen**. For the elliptic curve  $E_p(a, b)$  with base point  $G$ , pick  $k \xleftarrow{R} \mathbb{F}_p$  and compute  $K = kG$ . Set  $PK = K$  and  $SK = k$ .
- **Enc( $m, PK, r$ )**. For the plaintext point  $m$ , pick  $r \xleftarrow{R} \mathbb{F}_p$ . Compute  $C_1 = rG$ ,  $C_2 = m + rK$ . Set ciphertext points  $C = (C_1, C_2)$ .
- **Dec( $C, SK$ )**. For the ciphertext  $C$ , compute  $m' = C_2 - kC_1$ .

**Definition 1.** Suppose  $p$  is a prime and  $E_p(a, b)$  is an elliptic curve. For the two points  $G$  and  $Q$  on the elliptic curve, they satisfy  $Q = kG$ . It can be proved that it is easier to calculate  $Q$  from  $k$  and  $G$ . However, it is difficult to calculate  $k$  from  $Q$  and  $G$  [29].

The security of ECC is based on Elliptic Curve Discrete Logarithm Problem (ECDLP) which is consider to be computationally infeasible to solve.

**ECDH Key Exchange.** Elliptic Curve Diffie-Hellman (ECDH) key exchange is the elliptic cuive analogue of the classical Diffie-Hellman key exchange operating in  $\mathbb{Z}_p^*$ . We describe two communicating parties, usually called Alice and Bob, establish a shared secret key in secure communication channel as follows. We assume that Alice and Bob use the same set of domain parameters  $D := (p, a, b, G, n, h)$  for their computations.

- Alice generates an ephemeral key pair  $(k_A, Q_A)$ , i.e. he/she generates a random number  $k_A$  in  $[1, n - 1]$  and then performs a scalar multiplication to get the corresponding public key  $Q_A = k_A \cdot G$ . Then Alice sends  $Q_A$  to Bob.
- Bob generates an ephemeral key pair  $(k_B, Q_B)$  with  $(Q_B = k_B \cdot G)$  in the same way and sends  $Q_B$  to Alice.
- After Alice receives  $Q_B$ , he/she performs a scalar multiplication to obtain the shared secret  $S = k_A \cdot Q_B$ .
- After Bob receives  $Q_A$  from Alice, he/she obtains the shared secret through computation of  $S = k_B \cdot Q_A$

The security of the ECDH protocol relies on the intractability of (computational) Elliptic Curve Diffie-Hellman Problem (ECDHP). That is, given an elliptic curve  $E_p(a, b)$ , a base point  $G \in E(\mathbb{F}_p)$ , and two points  $Q_A = k_A \cdot G$  and  $Q_B = k_B \cdot G$ , find the point  $S = k_A \cdot k_B \cdot G$  without knowledge of  $k_A, k_B$ . It is clear that an algorithm for solving a generic ECDLP instance would allow one to solve the ECDHP as well.

### B. Verifiable Random Function.

A Verifiable Random Function (VRF) [30] is the public-key version of a keyed cryptographic hash. In this application, a Prover holds the VRF secret key and uses the VRF hashing to construct a hash-based data structure on the input data. Due to the nature of the VRF, only the Prover can answer queries about whether or not some data is stored in the data structure. Anyone who knows the public VRF key can verify that the Prover has answered the queries correctly. A VRF is a triplet of algorithms  $VRF := (\text{Gen}, \text{Eval}, \text{Vfy})$  providing the following functionalities.

- **Gen( $1^{\mathcal{K}}$ )**. The key generation algorithm is a probabilistic algorithm that takes as input the security parameter  $\mathcal{K}$  and outputs a key pair  $(vk, vsk)$ . We say that  $vsk$  is the secret key and  $vk$  is the verification key.
- **Eval( $vsk, X$ )**. The deterministic algorithm on input the secret key  $vsk$  and  $X \in \{0, 1\}^k$  and outputs a function value  $Y \in \mathcal{Y}$ , where  $\mathcal{Y}$  is a finite set, and a proof  $\pi$ . We write  $V_{vsk}(X)$  to denote the function value  $Y$  computed by Eval on input  $(vsk, X)$ .

- $\text{Ver}(vk, X, Y, \pi)$ . The verification algorithm takes as input  $(vk, X, Y, \pi)$  and outputs a bit  $b \in \{0, 1\}$  indicating whether or not  $\pi$  is a valid proof.

**Blockchain Basics.** We review some basic components of a proof-of-work blockchain [31]. We define a transaction  $tx := (\text{inputs}, \text{outputs}, \text{sig})$ , where  $\text{inputs}$  and  $\text{outputs}$  are the inputs and outputs of a UTXO-based model,  $\text{sig}$  is the signature signed by the transaction sender. A block is a triple of the form  $B := (s, x, txs, ctr)$ ,  $s \in \{0, 1\}^{\mathcal{K}}$ ,  $x \in \{0, 1\}^*$ ,  $ctr \in \mathbb{N}$ , where  $s$  is the state of the previous block,  $x$  is the data and  $ctr$  is the proof of work of the block. A block  $B$  is valid iff

$$\text{validBlock}^D(B) := H(ctr, T(s, x, txs)) < D.$$

Here,  $H : \{0, 1\}^* \rightarrow \{0, 1\}^{\mathcal{K}}$  and  $T : \{0, 1\}^* \rightarrow \{0, 1\}^{\mathcal{K}}$  are cryptographic hash functions, and the parameter  $D \in \mathbb{N}$  is the difficulty level of the block.

A chain is simply a chain of blocks, that we call  $\mathcal{C}$ . The rightmost block is called the head of the chain, denoted by  $\text{Head}(\mathcal{C})$ . Any chain  $\mathcal{C}$  with a head  $\text{Head}(\mathcal{C}) := (s, x, ctr)$  can be extended to a new longer chain  $\mathcal{C}' := \mathcal{C}||B'$  by attaching a block  $B' := (s', x', ctr')$  such that  $s' = H(ctr, G(s, x))$ ; the head of the new chain  $\mathcal{C}'$  is  $\text{Head}(\mathcal{C}') := B'$ . We let  $\mathcal{C} := \varepsilon$  to express a chain  $\mathcal{C}$  is empty. The function  $\text{len}(\mathcal{C})$  denotes the length of a chain  $\mathcal{C}$ .

For a chain  $\mathcal{C}$  of length  $n$  and any  $q > 0$ , we denote by  $\mathcal{C}^{\lceil q}$  the chain resulting from removing the  $q$  rightmost blocks of  $\mathcal{C}$ , and analogously we denote by  $\mathcal{C}^{\lceil q}$  the chain resulting in removing the  $q$  leftmost blocks of  $\mathcal{C}$ ; note that if  $q \geq n$  then  $\mathcal{C}^{\lceil q} := \varepsilon$  and  $\mathcal{C}^{\lceil q} := \varepsilon$ . If  $\mathcal{C}$  is a prefix of  $\mathcal{C}'$  we write  $\mathcal{C} \prec \mathcal{C}'$ . We also leverage  $slot$  which is defined in [1]. A slot is the continuous amount of divided time. Each slot  $slot_l$  is indexed for  $l \in \{1, 2, 3, \dots\}$ . We assume that users have a synchronised clock that indicates the current time down to the smallest discrete unit.

**Blockchain protocol.** With the illustrations of the basic components, we describe the blockchain protocol [31]  $\Gamma = (\Gamma.\text{KeyGen}, \Gamma.\text{Update}, \Gamma.\text{Validate}, \Gamma.\text{Broadcast})$  as follows.

- $\{pk, sk\} \leftarrow \Gamma.\text{KeyGen}$ : The algorithm generates the key pair  $(pk, sk)$  of the blockchain nodes.
- $\{\mathcal{C}', \perp\} \leftarrow \Gamma.\text{Update}$ : This algorithm returns a longer and valid chain  $\mathcal{C}$  in the network (if it exists), otherwise returns  $\perp$ .
- $\{0, 1\} \leftarrow \Gamma.\text{Validate}$ : The validity check algorithm takes as inputs a transaction  $tx$ , a block  $B$  or a chain  $\mathcal{C}$  and returns 1 iff the transaction, the block or the chain is valid according to a public set of rules.
- $\Gamma.\text{Broadcast}$ : The algorithm takes as inputs some  $txs$  and broadcasts it to all the nodes of the blockchain system.

The security of a PoW blockchain protocol  $\Gamma$  is characterized by three properties, namely: *Chain Growth*, *Chain Quality* and *Common Prefix* [31].

**Chain growth.** The chain property quantifies the number of blocks that are added to the blockchain during any given number of slots.

**Definition 2. (Chain Growth).** Consider the chains  $\mathcal{C}_1, \mathcal{C}_2$  possessed by two honest parties at the onset of two slots  $slot_1, slot_2$ ,

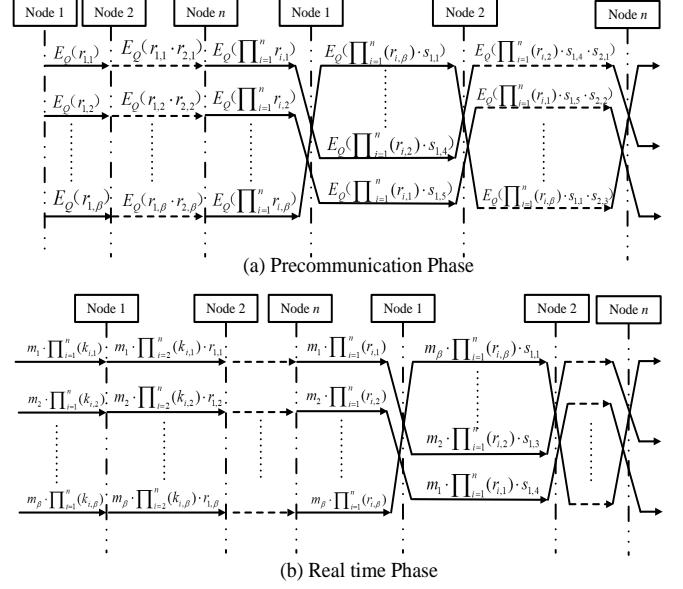


Fig. 1. Workflow of cMix.

$slot_2$ , with  $slot_2$  at least  $s$  slots ahead of  $slot_1$ . Then it holds that  $\text{len}(\mathcal{C}_2) - \text{len}(\mathcal{C}_1) \geq \tau \cdot s$ , for  $s \in \mathbb{N}$  and  $0 < \tau \leq 1$ , where  $\tau$  is the speed coefficient.

**Chain Quality.** The chain quality property informally states that the ratio of adversarial blocks in any segment of a chain held by a honest party is no more than a fraction  $\mu$ , where  $\mu$  is the fraction of resources controlled by the adversary.

**Definition 3. (Chain Quality).** Consider a portion of length  $\ell$ -blocks of a chain possessed by an honest party during any given slot intervals, for  $\ell \in \mathbb{N}$ . Then, the ratio of adversarial blocks in this  $\ell$  segment of the chain is at most  $\mu$ , where  $0 < \mu \leq 1$  is the chain quality coefficient.

**Common Prefix.** The common prefix property informally says that if we take the chains of two honest nodes at different times slots, the shortest chain is a prefix of the longest chain.

**Definition 4. (Common Prefix).** The chains  $\mathcal{C}_1, \mathcal{C}_2$  possessed by two honest parties at the onset of the slots  $slot_1 < slot_2$  are such that  $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$ , where  $\mathcal{C}_1^{\lceil k}$  denotes the chain obtained by removing the last  $k$  blocks from  $\mathcal{C}_1$ , where  $k \in \mathbb{N}$  is the common prefix parameter.

### C. cMix Anonymous System

The cMix protocol by Chaum et al. [10] is a new mix-net protocol which aims to provide an anonymous communication tool for users at large scales. In contrast with existing mix-net systems, cMix provides significant performance and security upgrades. Figure 1 briefly describes the workflow of cMix. The protocol contains two participants:  $Users := (U_1, \dots, U_\beta)$  and  $mix\ nodes := (N_1, \dots, N_n)$ . Each node  $N_i$  holds a tuple of the form  $(\pi_i, r_{i,j}, s_{i,j}, K_{i,t}, E(\cdot), D(\cdot))$ , where  $\pi_i$  is a random permutation,  $r_{i,j}, s_{i,j} \in \mathbb{G}(i \neq j)$  is the random elements in cyclic group  $\mathbb{G}$ ,  $K_{i,t} \in \mathbb{G}$  denotes the shared group element between node  $N_i$  and user  $U_t$ ,  $E(\cdot)$  and  $D(\cdot)$  denotes

the encryption and decryption algorithm of Elgamal. The detailed description of cMix protocol is provided in Appendix A.

We now present a collision tagging attack on cMix protocol. The attacker have to compromise the last node  $N_l$  and any mix node  $N_i$ . To launch the attack, only small changes are needed to the protocol:

- **Precomputation Phase- Step 3:** The mix node  $N_i$  calculate the decryption share  $\mathcal{D}(\vec{C}_1)$  with the vector  $(1, \dots, t^{-1}, \dots, 1)$  and commit to the  $\mathcal{D}(\vec{C}_1)$ .
- **Real time Phase- Step 1:** The mix node  $N_i$  adds tag  $(1, \dots, t, \dots, 1)$  to  $\vec{m}$ , and sends  $\vec{m}$  to the next mix node.
- **Real time Phase- Step 3:** The last node publish the output of the mixing step  $\Pi_h(\vec{m} \times \vec{R}_h) \times \vec{T}_h$ . Afterwards, all mix nodes release their decryption shares  $\mathcal{D}_i(\vec{C}_1)$  and the message component of the ciphertext  $\vec{C}_2$ . The mix node  $N_i$  waits for other mix nodes to publish their decryption first, then  $N_i$  collude with  $N_l$  to obtain the message package with the tagged messages. After that,  $N_i$  publish  $\mathcal{D}(\vec{C}_1)$ .

The mix node  $N_i$  and  $N_l$  get the location of the tag message in advance, and the mixed messages are the same from the senders' perspective. Thus the attacker can break the anonymity of cMix.

### III. SECURITY MODEL AND REQUIREMENTS

In this section, we present our proposed system model for BCMIX and the related security requirements. The communication methods among them include transactions and Transport Layer Security (TLS), where the former is an on-chain communication (i.e., publishing a transaction using P2P communications) and the latter is an off-chain communication (i.e., establishing a secure communication channels among mix nodes).

#### A. System Model

There are three entities in our proposed BCMIX, that is, Miners, Mix nodes and Senders (see Figure 2).

- **Miners:** These entities validate new transactions and record them on the global ledger. Simultaneously, the entities compete to solve a difficult mathematical puzzle based on a cryptographic hash algorithm. In BCMIX, miners are eligible to become mix nodes through PoW algorithm competition. Miners disclose their addresses in the form of  $(address_M, pk_M)$  where  $address_M$  is the blockchain address and  $pk_M$  is the blockchain public key deriving the related address.
- **Mix nodes:** These entities are selected from miners through the PoW and VRF algorithm. After being elected as mix nodes successfully, these entities firstly negotiate keys with senders in the set up phase. Then they execute the precomputation and real time phase to encrypt and mix the messages during the duty period and pass the message down. Besides, they should commit to their computations and send special transactions to blockchain network for subsequent auditing.

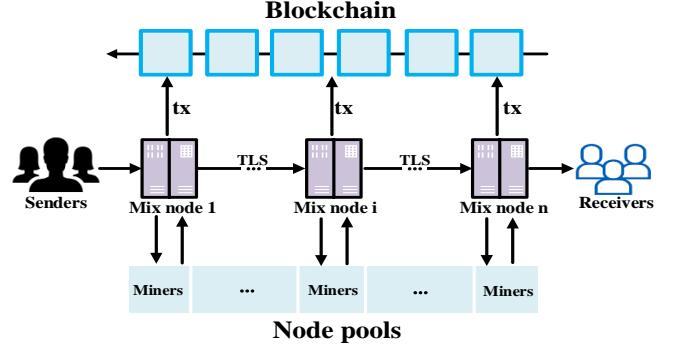


Fig. 2. Architecture of blockchain-based mix anonymous system. Before anonymity phase, we elect miners in node pools to serve as mix nodes via designed algorithms. During anonymity phase, mix nodes mix and forward messages from users through Transport Layer Security (TLS). Meanwhile, mix nodes sends special transactions to blockchain for auditing.

- **Senders:** These entities refers to BCMIX users, who hold the respective accounts  $address_U, pk_U$ . Before accessing to the anonymous service, senders send transactions to mix nodes and negotiate the corresponding keys with mix nodes. Then in the real time phase, senders blind messages with the shared keys and send the message to the first mix nodes.

#### B. Threat Model

BCMIX assumes authenticated communication channels among all mix nodes. Therefore, we consider a malicious adversary (a.k.a Byzantine), who can delay, drop, eavesdrop, forward, and delete messages between mix nodes, but not modify, replay, or inject new ones, without detection. For any communication not among mix nodes, we assume the adversary can delay, drop, re-order, eavesdrop, modify, or inject messages at any point of the network. BCMIX accepts one message per user per batch, starting the pre-computation once the batch reaches  $\beta$  messages.

The adversary can also create a lot of accounts and compromise an arbitrary numbers of users. In addition, we assume the adversary can control more than 50% of the system computing powers. However, such adversary is not able to read the contents of the messages. We assume the security of the used cryptographic primitives, including a secure hash function and a secure signature scheme.

#### C. Security Requirements

According to the existing literature [5], [32], [33], BCMIX needs to satisfy the following fundamental security requirements.

- **Resistance to Sybil Attacks.** BCMIX should minimize the possibility of an attacker being successfully selected as multiple mix nodes at the same time.
- **Resistance to Collision Tagging Attacks.** BCMIX should prevent collision attackers from performing tagging attacks to link a message to a certain sender.

- **Sender Anonymity.** BCMIX provide sender anonymity for users. That is, every mix node performs mixing operations on messages, destroying the associations between senders and receivers. Thus an attacker can not associate the export messages with a certain sender.
- **Resistance to MitM attacks.** BCMIX should prevent an attacker from replacing the shared keys between senders and mix nodes.
- **Single Point of Failure Resilience.** In case of single point of failure (e.g. a mix node is under denial of service attacks or the mix node is crashed), BCMIX should detect the failure and guarantee the system keep operating.
- **Resistance to Other Attacks.** BCMIX should resist common attacks on mix-net system such as replay attacks, traffic-analysis attacks and so on.

#### IV. PROPOSED BCMIX SYSTEM

In this section, we will present our construction of BCMIX system. We first introduce an additive homomorphism mix-net protocol and then we propose the blockchain based mix-net protocol. Thereafter, we describe the concrete BCMIX system.

##### A. Additive Homomorphism Mix-net Protocol

To solve the MitM attacks of key agreement process and the dependency on trusted entities, we replace Elgamal in cMix [10] with EC-Elgamal and propose an additive homomorphism mix-net protocol  $\Delta$  to integrate the mix-net protocol with the blockchain.

Our mix-net protocol contains two participants,  $Senders := (U_1, \dots, U_\beta)$  and  $mix\,nodes := (N_1, \dots, N_n)$ , where the mix nodes are selected from the blockchain miners. Mix nodes negotiate keys with senders by means of a special transaction  $tx_{KE}$  and calculate the system public key through their address pair  $(address_N^i, pk_N^i)$ , where  $pk_M^i = Q_i$ . (We will describe these two processes in the next part). We suppose a authenticated communication among mix nodes and we denote it as  $\Delta.TLS$ . The proposed mix-net protocol is a tuple of algorithms  $(Setup, Precom, RealTime)$ . The notations and the processes of the protocol  $\Delta$  are presented in Table I and Algorithm 1.

##### B. Basic Components of the Proposed Blockchain Protocol

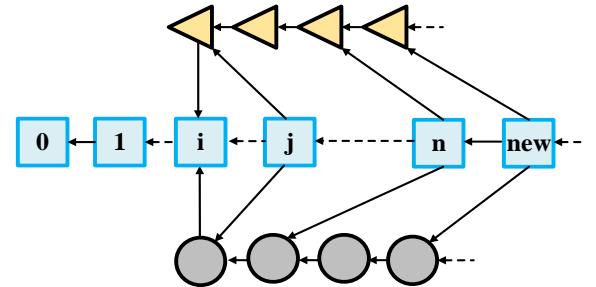
We build our blockchain protocol  $\Gamma^*$  by extending and modifying the aforementioned protocol  $\Gamma$ . We first define the basic components in our blockchain protocol.

**Transaction.** Our blockchain contains three types of transaction, namely normal transaction  $tx_N$ , key-exchange transaction  $tx_{KE}$  and commitment transaction  $tx_{COM}$ . The normal transaction  $tx_N$  is the same definition of the transaction that in protocol  $\Gamma$ . We define a key-exchange transaction  $tx_{KE} := (KE, pk, inputs, outputs, sig)$  and a commitment transaction  $tx_{COM} := (COM, commitment, sig)$ , where  $KE$ ,  $COM$  denotes the transaction type,  $pk$  is the blockchain public key of the sender,  $commitment$  is the commitment value issued by mix nodes and  $inputs, outputs, sig$  are the same as the above definition. A key-exchange transaction is used to negotiate

TABLE I  
NOTATIONS OF THE REVISED MIX-NET PROTOCOL

Symbol	Description
$d_i$	the secret share for mix node $N_i$ of the secret key $d$ , $d_i \in \mathbb{F}_p$ ;
$Q_i$	the public key of mix node $N_i$ , $Q_i = d_i G$ ;
$Q$	the public key of the system, $Q = \sum_i Q_i$ ;
$\mathcal{E}_Q()$	$\mathcal{E}_Q(m) = (x \cdot G, m + x \cdot Q)$ , $x \in \mathbb{F}_p$ ;
$\mathcal{D}_{d_i}()$	the decryption share of mix node $N_i$ , $\mathcal{D}_{d_i}(\sum_i x_i \cdot G) = (\sum_i x_i) d_i \cdot G$ ;
$r_{i,a}$	random values (freshly generated for each round) of mix node $N_i$ for groove $a$ ;
$s_{i,a}$	random values (freshly generated for each round) of mix node $N_i$ for groove $a$ ;
$\pi_i$	a random permutation of the $\beta$ grooves used by mix node $N_i$ ;
$\Pi_i$	the permutation performed by BCMix through mix node $N_i$ ;
$k_{i,j}$	a group element shared between mix node $N_i$ and the sending user for groove $j$ . These values are used as keys to blind messages;
$\mathbf{k}_i$	the vector of derived secret keys shared between mix node $N_i$ and all users in a batch, i.e. $\mathbf{k}_i = (k_{i,1}, \dots, k_{i,\beta})$ ;
$K_j$	the product of all shared keys for the sending user of slot $j$ , i.e. $K_j = \prod_{i=1}^n k_{i,j}$ ;
$M_j$	the message sent by user $j$ . Like other values in the system, these values are group elements;
$R_i$	the product of all local random $r$ values through mix node $N_i$ , $R_i = \prod_{j=1}^i r_j$ ;
$S_i$	the product of all local random $s$ values through mix node $N_i$ , $S_i = \begin{cases} s_1 & i = 1 \\ \pi_i(S_{i-1}) \times s_i & 1 < i \leq n \end{cases}$ .

□ Main block    ● Commit block    ▲ Key exchange block



$\mathcal{C}_1$  and  $\mathcal{C}_1^{\Gamma(w+1)} \neq \mathcal{C}_1$ , then we call  $\mathcal{C}_M^w$  is a stable main chain and  $B_M^i$  where  $i \in [1, w]$  are stable main blocks.

Note that the  $B_{KE}$  and the  $B_{COM}$  do not contain proof of work, thus an adversary can manipulate the contents of these blocks. To avoid the problem we specify that the latest main block  $B_M^{latest}$  always contains the states of  $B_{KE}^{latest}$  and  $B_{COM}^{latest}$ . We further give out the definition of valid blocks.

**Definition 6.** We say blocks  $B_N$ ,  $B_{KE}$  and  $B_{COM}$  are valid iff

- The transactions contained in  $B_N$ ,  $B_{KE}$  and  $B_{COM}$  are valid;
- For any two main blocks  $B_N^i := (s_{ke}^p, s_{com}^q, x^i, tx_N^i s, ctr^i)$  and  $B_N^j := (s_{ke}^m, s_{com}^n, x^j, tx_N^j s, ctr^j)$ , where  $i < j$ ,  $p \geq m$  and  $q \geq n$  hold;
- $H(ctr, T(s_{ke}, s_{com}, x, txs)) < D$ .

**Blockchain based mix-net protocol.** Below we propose a blockchain based mix-net protocol  $\Gamma^*$  which combines the basic blockchain protocol  $\Gamma$  with the proposed mix-net protocol  $\Delta$ . The protocol  $\Gamma$  has copies of all the basic functionalities exposed by  $\Gamma$  and  $\Delta$  through the interfaces described above, and adds additional algorithms including VRF, IP Sharding in order to resist Sybil attacks. We describe the protocol  $\Gamma^* := (\text{Setup}, \text{Update}, \text{Vote}, \text{Mix}, \text{Verify}, \text{Broadcast})$  as follows.

- **Setup.** System parameters involved in our construction are  $\{E_p(a, b), \mathbb{G}, G, \mathcal{K}, H, T\}$ , where  $E_p(a, b)$  is a non-singular elliptic curve,  $\mathbb{G}$  is a cyclic group which consists of all points on  $E_p(a, b)$ , as well as the point at infinity  $\mathcal{O}$ ,  $G$  is the base point of  $E_p(a, b)$ ,  $\mathcal{K}$  is the security parameter and  $H : \{0, 1\}^* \rightarrow \{0, 1\}^{\mathcal{K}}$ ,  $T : \{0, 1\}^* \rightarrow \{0, 1\}^{\mathcal{K}}$  are cryptographic hash functions. Entities invoke the algorithm to generate the blockchain key pair  $(pk, sk) := (qG, q)$ , the address  $(address, pk)$  and the VRF key pair  $(vk, vsk)$ . Here  $q \in \mathbb{F}_p$  is selected by entities.
- **Update.** This algorithm first invoke  $\Gamma.\text{validate}$  to validate the new transactions, chains and blocks. Then returns a longer valid chain  $\mathcal{C} := (\mathcal{C}_M, \mathcal{C}_{KE}, \mathcal{C}_{COM})$  in the network (if it exists), otherwise returns  $\perp$ .
- **PoWVote.** The algorithm take as inputs the data  $x$ , the state  $s$  and a difficulty level  $D' < D$ , where  $D$  is the system difficulty level, and then computes  $H(ctr', T(s, x))$  where  $ctr'$  is a random string. Thereafter, the algorithm estimate whether  $H_i(ctr', T(s, x)) < D'$ . If so the algorithm outputs 1 and add the miner into a set  $\mathcal{M}$ , otherwise returns 0.
- **IPSharding.** This algorithm takes as input IP prefix of miners in set  $\mathcal{M}$ , and outputs  $n$  node pools (Miners in each node pool have the same IP prefix  $IP/z$  where  $z$  is a integer and  $z \in [0, 32]$ .)
- **VRF.** The algorithm takes as inputs  $IP/z$  and the current slot  $slot_l$ , and outputs a value  $Y \in \mathcal{Y}$ , where  $\mathcal{Y}$  is a finite set, and a proof  $\pi$ .
- **Mix.** The algorithm take as inputs a sender set  $\mathcal{U} := (U_1, U_2, \dots, U_\beta)$ , a mix node set  $\mathcal{N} := (N_1, N_2, \dots, N_n)$  and a message vector  $\mathbf{M}$ . Then it

revokes  $(\Delta.\text{Setup}, \Delta.\text{Precom}, \Delta.\text{RealTime})$  and outputs a permuted message vector  $\Pi_n(\mathbf{M})$ .

- **Verify.** Given a message vector  $\mathbf{M} := (m_1, m_2, \dots, m_\beta)$  and a message vector  $\mathbf{M} := (m_{(1)}, m_{(2)}, \dots, m_{(\beta)})$ , this algorithm decides whether the following conditions holds: For  $\forall i \in [1, \beta]$ ,  $m_i \in \mathbf{M}$ ,  $\exists j \in [1, \beta]$ ,  $m_j \in \mathbf{M}$ ,  $m_i = m_j$ . If so, the algorithm returns 1. Otherwise the algorithm returns 0.
- **Broadcast:** The algorithm takes as inputs some  $txs$  and address  $(address, pk)$  and broadcasts them to all the nodes of the blockchain system.

**Theorem 1.** If  $\Gamma$  satisfies  $(\tau, s)$ -chain growth, then  $\Gamma^*$  satisfies  $(\tau, s)$ -chain growth.

*Proof.* We note that the side chain  $\mathcal{C}_{KE}$  and  $\mathcal{C}_{COM}$  do not contain proof of work, and the update of  $\mathcal{C}_{KE}$ ,  $\mathcal{C}_{COM}$  is independent of the update of  $\mathcal{C}_M$ . Thus we conclude that  $\mathcal{C}_M$  satisfies  $(\tau, s)$ -chain growth as the chain  $\mathcal{C}$  in  $\Gamma$ . We now prove that  $\mathcal{C}_{KE}$  and  $\mathcal{C}_{COM}$  satisfy  $(\tau, s)$ -chain growth.

Consider the chains  $\mathcal{C}_{KE}^1, \mathcal{C}_{KE}^2$  and  $\mathcal{C}_{COM}^1, \mathcal{C}_{COM}^2$  at the onset of two slots  $slot_1, slot_2$ , with  $slot_2$  at least  $s$  slots ahead of  $slot_1$ , and  $\tau_{KE}, \tau_{COM}$  are the speed coefficient of  $\mathcal{C}_{KE}$  and  $\mathcal{C}_{COM}$ . Since a key exchange block  $B_{KE}$  and a commitment block  $B_{COM}$  are jointly generated by all mix nodes at the expected speed  $\tau_{KE}$  and  $\tau_{COM}$  respectively, we derive that  $\mathcal{C}_{KE}$  and  $\mathcal{C}_{COM}$  are updated at the expected speed  $\tau_{KE}$  and  $\tau_{COM}$ . Then we hold the following two equations  $\text{len}(\mathcal{C}_{KE}^2) - \text{len}(\mathcal{C}_{KE}^1) = \tau_{KE} \cdot s$  and  $\text{len}(\mathcal{C}_{COM}^2) - \text{len}(\mathcal{C}_{COM}^1) = \tau_{COM} \cdot s$ . Thus we conclude that  $\mathcal{C}_{KE}$  and  $\mathcal{C}_{COM}$  satisfies  $(\tau, s)$ -chain growth. In summary,  $\Gamma^*$  satisfies  $(\tau, s)$ -chain growth.  $\square$

**Theorem 2.** Let  $H, G$  be two collision-resistant cryptographic hash functions. If  $\Gamma$  satisfies  $(\mu, \ell)$ -chain quality, then  $\Gamma^*$  satisfies  $(\mu, \ell)$ -chain quality.

*Proof.* We emphasize that proof of work is not contained in the key exchange chain  $\mathcal{C}_{KE}$  and the commitment chain  $\mathcal{C}_{COM}$ . And the security of  $\mathcal{C}_{KE}$ ,  $\mathcal{C}_{COM}$  is depend on the security of the main chain  $\mathcal{C}_M$  since  $\mathcal{C}_M$  takes the states of  $\mathcal{C}_{KE}$  and  $\mathcal{C}_{COM}$  as inputs of the proof of work. Suppose an adversary  $\mathcal{A}$  wants to manipulate contents of  $\mathcal{C}_{KE}$  and  $\mathcal{C}_{COM}$ . According to the formula  $H(ctr, T(s_{ke}, s_{com}, x, txs)) < D$  (Definition 7.),  $\mathcal{A}$  has to amend the corresponding main block  $B_M$ . Note that the capabilities of the adversary and external environment in  $\Gamma^*$  are exactly the same as that in  $\Gamma$ . Thus we can conclude that the main chain  $\mathcal{C}_M$  is the only factor affecting the chain quality property. We show below that  $\mathcal{A}$  has only a negligible probability of violating chain quality of  $\Gamma^*$ .

Let us denote by  $B_M^i$  the  $i$ -th block of the main chain  $\mathcal{C}_M$  at some slot intervals so that  $\mathcal{C}_M := B_M^1 \cdots B_M^{\text{len}(\mathcal{C})}$ . From Definition 4. we know that the number of main blocks generated by  $\mathcal{A}$  in chain  $\mathcal{C}_M$  are at most  $\mu \cdot \text{len}(\mathcal{C})$ . According to [31],  $\mathcal{A}$  can not generate more than  $\mu \cdot \text{len}(\mathcal{C})$  main blocks with the current computing hash power. Or the adversary  $\mathcal{A}$  could try to build an valid candidate block  $B_M^*$  to replace a validate block  $B_M^j$  generated by honest parties in  $\mathcal{C}_M$ , where  $j \in [1, \text{len}(\mathcal{C})]$ ,  $B_M^* \neq B_M^j$  and  $H(B_M^*) = H(B_M^j)$ . By the collision-resistance property of hash function we can draw a

**Algorithm 1**The Additive Homomorphism Mix-net Protocol  $\Delta$ 

- 1: **procedure SETUP PHASE**
- 2:  $(Q, K_i, k_{i,j}) \leftarrow \text{Setup}(pk_U^i, pk_M^j)$ : This algorithm is invoked by senders  $U_i$  with address  $(\text{address}_U^i, pk_U^i)$  and mix nodes  $M_j$  with address  $(\text{address}_N^j, pk_N^j)$ , where  $i \in [1, \beta]$ ,  $j \in [1, n]$ . The algorithm takes as inputs the public key of senders and mix nodes and returns a shared key  $k_{i,j}$  between a send  $U_i$  and a mix node  $M_j$ . A slot key  $K_i = \prod_{j=1}^n k_{i,j}$  and the system public key  $Q = \sum_j pk_M^j = \sum_j Q_j$ .
- 3: **procedure PRECOMPUTATION PHASE** Precom := (Preprocess, Mix, Postprocess)
- Preprocess. The mix nodes generate the fresh  $r, s, \pi$  values and computes the encryption  $\mathcal{E}(r_i^{-1})$ . At the same time the mix nodes issues the commitment values of the fresh values  $COM_r, COM_s, COM_\pi$ . Then they collectively compute the product of the received values by sending the following message to the next mix node:
- $$\mathcal{E}_Q(\mathbf{R}_i) = \begin{cases} \mathcal{E}_Q(r_1) & i = 1 \\ \mathcal{E}_Q(\mathbf{R}_{i-1}) \times \mathcal{E}_Q(r_i) & 1 < i \leq n \end{cases}$$
- Eventually, the last mix node sends the final values  $\mathcal{E}_Q(\mathbf{R}_n)$  to the first mix node as input for the next step and issues the commitment value  $COM_{\mathcal{E}_Q(\mathbf{R}_n)}$ .
- Mix. The mix node together mix the values and compute the results  $\Pi_n(\mathbf{R}_n) \times S_n$ , under encryption. The mix nodes perform this mixing by having each mix node  $i$  send the following message to the next mix node:
- $$\mathcal{E}_Q(\Pi_n(\mathbf{R}_n) \times S_i) = \begin{cases} \pi_1(\mathcal{E}_Q(\mathbf{R}_n) \times \mathcal{E}_Q(s_1)) & i = 1 \\ \pi_i(\mathcal{E}_Q(\Pi_{i-1}(\mathbf{R}_n) \times S_{i-1})) \times \mathcal{E}_Q(s_i) & 1 < i \leq n \end{cases}$$
- As with the first step, the last mix node sends the final encrypted values  $\mathcal{E}_Q(\Pi_n(\mathbf{R}_n) \times S_n)$  to the first mix node.
- Postprocess. To complete the precomputation, each mix node  $N_i$  computes its decryption shares  $D_{d_i}(\sum_i x_i \cdot G)$ , where  $(x, c) = \mathcal{E}_Q(\Pi_n(\mathbf{R}_n) \times S_n)$ , and keep its secret. Then each mix node issues the commitment values  $COM_{D_{d_i}}$  of their secret shares. The message parts  $c$  are multiplied with all the decryption shares to retrieve the plaintext values  $\Pi_n(\mathbf{R}_n) \times S_n$ . The last mix node to be used in the real time phase stores the decrypted precomputed values.
- 4: **procedure REALTIME PHASE** RealTime := (Preprocess, Mix, Postprocess)
- 5: Each user constructs its message  $MK_j^{-1}$  (for slot  $j$ ) by multiplying the message  $M_j$  and it sends it to the first mix node, which collects all messages and combines them to get a vector  $\mathbf{M} \times \mathbf{K}^{-1}$ .
- Preprocess. Each node  $N_i$  sends  $k_i \times r_i$  to the next mix node, which uses them to compute  $\mathbf{M} \times \mathbf{R}_n = \mathbf{M} \times \mathbf{K}^{-1} \times \prod_{i=1}^n k_i \times r_i$  and the last node sends the result to the first node.
  - Mix. Each node  $N_i$  computes  $\pi_i(\Pi_{i-1}(\mathbf{M} \times \mathbf{R}_n) \times S_{i-1}) \times s_i$ , where  $\Pi_0$  is the identity permutation and  $S_0 = 1$ . The last node sends a commitment to its message  $\Pi_n(\mathbf{M} \times \mathbf{R}_n) \times S_n$  to every other node.
  - Postprocess. Each node  $N_i$  opens its precomputed decryption share for  $(x, c) = ((\Pi_n(\mathbf{R}_n) \times S_n)^{-1})$ , while the last node sends its decryption share multiplied by the value in the previous step and the message component:  $\Pi_n(\mathbf{M} \times \mathbf{R}_n) \times S_n \times D_n(x) \times c$ . Finally, the permuted message can be decrypt as  $\Pi_n(\mathbf{M}) = \Pi_n(\mathbf{M} \times \mathbf{R}_n) \times S_n \times \prod_{i=1}^n D_i(x) \times c$ .
- 6: **Note:** The shared key  $k_{ij}$  in **Setup Phase** is generated through ECDH protocol. After the mix node set  $(N_1, N_2, \dots, N_n)$  is selected from miners. Every sender sends  $tx_{KES}$  to all mix nodes. Since senders and mix nodes know the public key of each other, they can execute the ECDH protocol and obtain the shared key  $k_{ij}$ .

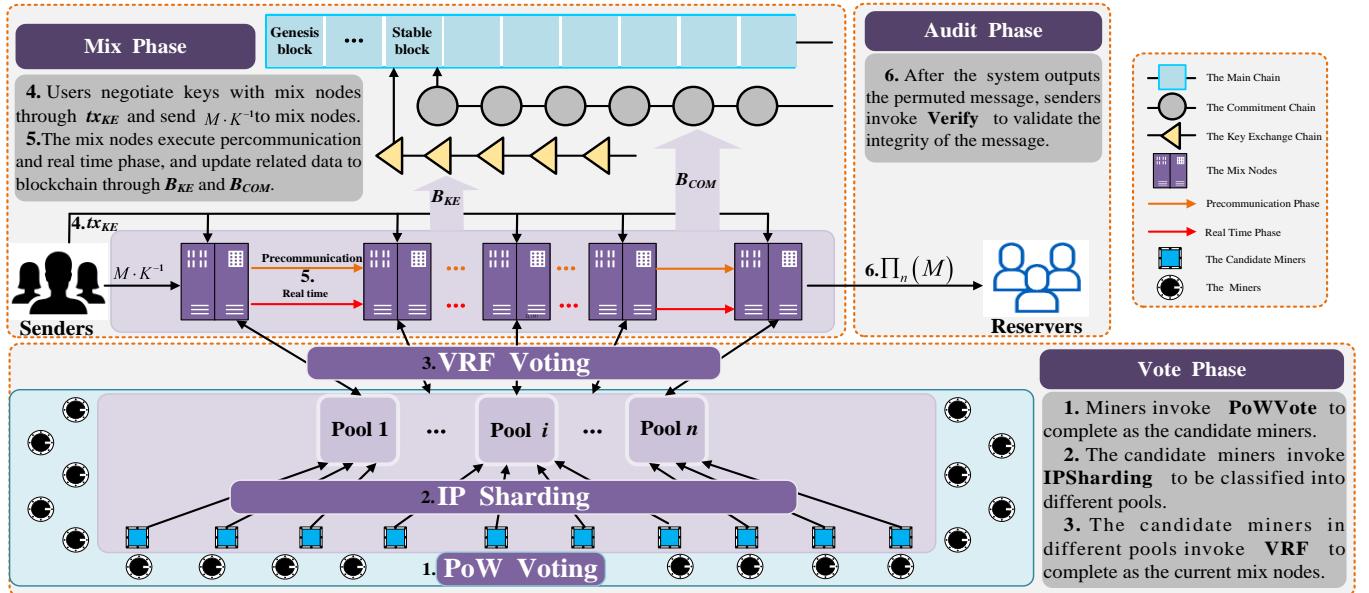


Fig. 4. The detailed orchestration of BCMIX.

conclusion that the adversary has only a negligible chance of producing such a candidate block  $B_M^*$  where  $H(B_M^*) = H(B_M^j)$ . Hence  $\Gamma^*$  satisfies  $(\mu, \ell)$ -chain quality.  $\square$

**Theorem 3.** Let  $B_M^w$  be a stable block which is the genesis of the chain  $\mathcal{C}_{KE}$  and  $\mathcal{C}_{COM}$ . If  $\Gamma$  satisfies  $k$ -common prefix, then  $\Gamma^*$  satisfies  $k$ -common prefix.

*Proof.* Note that the chain  $\mathcal{C}_{KE}$  and  $\mathcal{C}_{COM}$  will not fork since they do not contain proof-of-work and are uniquely generated by the mix nodes. Hence  $\mathcal{C}_{KE}$  and  $\mathcal{C}_{COM}$  satisfy the  $k$ -common prefix property. We recall that the behaviors of the adversary and the honest parties in  $\Gamma^*$  are exactly the same as that in  $\Gamma$ . Thus according to the literature [], the main chain satisfies  $k$ -common prefix property. This concludes the proof.  $\square$

Given the above, the tuple  $\Gamma^* := (\text{Setup}, \text{Update}, \text{Vote}, \text{Mix}, \text{Verify})$  is a secure blockchain protocol which satisfies the properties of  $(\tau, s)$ -chain growth,  $(\mu, \ell)$ -chain quality and  $k$ -common prefix.

### C. System Design

With the description of the blockchain-based mix-net protocol, we propose the dynamic self-organizing blockchain-based mix anonymous system in this part. Our BCMIX system consists of four phases, namely: **System Initialization, Vote, Mix, Audit**. The orchestration of BCMIX is detailed in Figure 4.

**System Initialization.** This phase initializes the system parameters and generate accounts for participants. The system determine the public system parameters  $\{E_p(a, b), \mathbb{G}, G, \mathcal{K}, H, T\}$  (e.g. secp256k1 of Bitcoin). After that the entities (i.e., users and miners) invoke  $\Gamma^*\text{Setup}$  to generate their key pairs (i.e., user key pair  $(pk_u^i, sk_u^i) := (q_u^i G, q_u^i)$  and miner key pair  $(pk_m^j, sk_m^j) := (q_m^j G, q_m^j)$ ) and addresses (i.e., user address  $(address_U^i, pk_U^i)$  and miner address  $(address_N^j, pk_N^j)$  respectively). In addition, miners also generate their VRF key pairs, which are used to complete as mix nodes.

**Vote.** This phase is executed by the miners to elect the candidate mix nodes for mixing messages.

- 1) Firstly, miners invokes  $\Gamma^*\text{PoWVote}$  to solve the puzzle  $D'$ . If the algorithm  $\Gamma^*\text{PoWVote}$  returns 1, then the miner  $M_i$  is added to the candidate set  $\mathcal{M}$ . Otherwise miners re-select inputs  $s, x$  to compute  $H(ctr', T(s, x))$  until  $H(ctr', T(s, x)) < D'$ .
- 2) After a period of certain time, e.g.  $T$ ,  $\mathcal{M}$  stops accepting new miners. Then the candidates miners run  $\Gamma^*\text{IPSharding}$  (Algorithm 2) and join the node pool with the same IP prefix.
- 3) After that the candidate miners  $M_i$  in each node pool take the  $IP/z$  and current slot  $slot_l$  as inputs, and invoke  $\Gamma^*\text{VRF}$  to generate a value  $Y_i \in \mathcal{Y}$  and the corresponding proof  $\pi$ . Then the candidate miner who holds the smallest value  $Y$  in each node pool is elected as the mix node in the current slot. Finally, the elected mix nodes are networked in the order of  $Y$ .

<b>Algorithm 2</b>	The IPSharding algorithm.
<b>Input:</b>	The IP address of the candidate miners, where $ip_i := A_1.A_2.*.*$ . Here $A_1, A_2$ and $*$ denote four decimal segments; the system parameter $\sigma$ ;
<b>Output:</b>	$k$ mix node pools;
1:	$coordinate_{ip_m} := (A_1^m, A_2^m);$
2:	$ Num_{2^p, 2^q} $ denotes the number of nodes distributed in $(2^p, 2^q)$ ;
3: <b>for</b> $j = 0, j <= 2$ <b>do</b>	
4: <b>for</b> $i = 0, i <= 7$ <b>do</b>	
5: <b>if</b> $A_j^m < 2^i$ or $A_j^m > 2^{i+1}$ <b>then</b>	
6: $i++$ ;	
7: <b>else</b> break;	
8: $j++$ ;	
9: Classify the coordinate into $k$ parts such that	
	$\frac{ Num_{2^p, 2^q} _j}{ Num_{2^p, 2^q} _{min}} \leq \sigma, k \in [1, k];$
10: <b>return</b> $k$ mix node pools.	

**Mix.** In this phase, the current mix nodes  $\mathcal{N} := (N_1, N_2, \dots, N_n)$  first negotiate shared keys with users  $\mathcal{U} := (U_1, U_2, \dots, U_\beta)$ . Then the current mix nodes blind and mix messages  $\mathbf{M} := (M_1, M_2, \dots, M_\beta)$  for users.

To negotiate keys with each other through ECDH protocol, the mix nodes and the users do as follows.

- 1) After the current mix nodes are selected, they run  $\Gamma^*\text{Broadcast}$  to disseminate their addresses and the VRF parameters, i.e.  $(address, pk) || (vk, Y, \pi)$ , to the participants in the blockchain network.
- 2) Upon receiving the addresses and parameters of all mix nodes, a user  $U_i$  first invoke  $\text{VRF.Ver}$  to verify the received. If the algorithm returns 1, then the user  $U_i$  sends transactions  $tx_{KE}^i$  to every mix nodes. Then the mix nodes parse  $tx_{KE}^i$  as  $tx_{KE} := (KE, pk, \overrightarrow{\text{inputs}}, \overrightarrow{\text{outputs}}, sig)$  and obtain the public key of each user.
- 3) Finally the mix nodes  $N_j$  and the user  $U_i$  multiply their secret key with the public key of each other, and obtain the shared key  $k_{ij} = q_U^i \cdot q_N^j \cdot G$ .

After negotiating keys with users, the mix nodes network blind and mix messages as follows.

- 1) The mix nodes invoke  $\Delta.\text{Precom}$  to compute the parameters  $\mathcal{E}_Q(\Pi_n(\mathbf{R}_n) \times \mathbf{S}_n)$  utilized in the real time phase and update the corresponding commitment values  $COM_r, COM_s, COM_\pi, COM_{D_{a_i}}$  to the commitment chain  $\mathcal{C}_{COM}$  through the commitment transaction  $tx_{COM}$ .
- 2) After receiving the blind messages  $\mathbf{M} \times \mathbf{K}^{-1}$  from the users, the mix nodes invoke the algorithm  $\Delta.\text{RealTime}$  to mix the received messages and obtain the permuted message  $\Pi_n(\mathbf{M})$ .

**Audit.** In this phase, the users invoke  $\Gamma^*\text{Verify}$  to check the integrity of the permuted messages. If the algorithm  $\Gamma^*\text{Verify}$  returns 1 to all the users, then the permuted messages are considered integrated. If  $\Gamma^*\text{Verify}$  returns 0 to some users, then the users ask the current mix nodes to check their calculation through the corresponding commitment block  $B_{COM}$

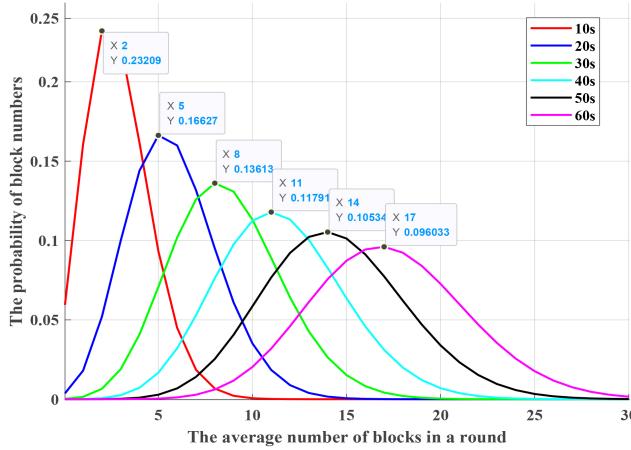


Fig. 5. The probability of the expected number of candidate miners

and identify the malicious mix nodes. The malicious mix node will be removed out of BCMIX.

## V. PERFORMANCE EVALUATION

This section presents the performance evaluation of BCMIX. We firstly theoretically analyze the number of candidate miners. Then we describe the implementation of BCMIX and evaluate its performance.

### A. The Number of candidate miners

In Bitcoin, the difficulty value  $D$ , the target  $target$  and the network hash rate satisfy the following formula:

$$D = \frac{D_{Max}}{target_{current}}, \quad Hashrate_{min} = \frac{D \cdot 2^{32}}{T},$$

where  $D_{Max}$  is a large constant,  $target_{current}$  denotes the current target and  $Hashrate_{min}$  represents the minimum hash rate required to calculate a block of difficulty  $D$  within time  $T$ . According to literature [], the process of generating  $n$  blocks within the average time  $t$  will be a Poission distribution with the expected value  $\lambda$ ,

$$P(X \geq M) = \sum_{k=M}^{\infty} \frac{\lambda^k}{k!} e^{-\lambda}.$$

Since the Bitcoin network with  $Hashrate_{min}$  mining power generates one block within average time  $T$ , we can conclude that

$$\lambda = \frac{Hashrate_{real}}{Hashrate_{min}} = \frac{Hashrate_{real} \cdot T}{D \cdot 2^{32}}.$$

Here  $Hashrate_{real}$  denotes the mining power of the Bitcoin network in the real world. To better simulate BCMIX in the real world scenario, we leverage the mining power distribution from <sup>1</sup>. In Appendix B, Table IV details the IP address and mining power of different mining pools. With  $D = 1 \times 10^{11}$ , we estimate the probability of simultaneous block generation under different  $T$  and report our results in Figure 5. We can

<sup>1</sup>[https://btc.com/stats/pool?pool\\_mode=year](https://btc.com/stats/pool?pool_mode=year)

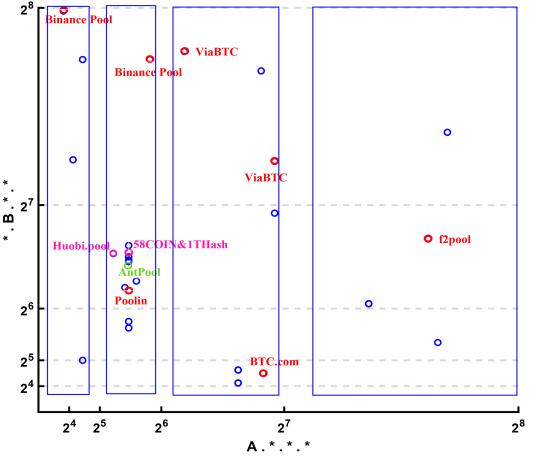


Fig. 6. The results of IPSharding algorithm when  $\sigma = 5$  (four node pools, i.e. four mix nodes). We denote an IP address as four segments, the  $x$  axis is the first segment and the  $y$  axis denotes the second segment. We can change the division strategy to control the number of mix nodes. We respectively simulate the attackers which hold 53.84% (the red circle), 63.43% (the red circles and the green circle) and 74.80% (the red circles, the green circle and the purple circles) of total computing powers.

see that when fixing the difficulty, the longer the average time to generate a block, the higher the probability of generating multiple blocks simultaneously.

### B. Implementation

In order to show the feasibility of BCMIX, we build a Bitcoin network in a desktop computer (equipped with a Ubuntu 16.04 LTS, Intel (R) Core (TM) i5-8500 CPU of 3.00GHz and 16GB RAM). We make use of two Github programs, Bitcoin-Simulator <sup>2</sup> and cMix <sup>3</sup>, to evaluate the proposed BCMIX. We set up the public parameters by utilizing the publicly available library for cryptography on the secp256k1 curve <sup>4</sup>.

We apply the Algorithm 2 to Table IV and obtain the distribution of mining pools in the real world. The results are shown in Figure 6. Combining Figure 5 and Figure 6, we can find that when fixing difficulty  $D$  to  $1.0 \times 10^{11}$ , setting the average block generation time to 20s or 30s can obtain a reasonable number of mix nodes, which is more conducive to the implementation of BCMIX. To measure the approximate time cost, we test the functionality of PoWVote, IPSharding, Key Exchange, Precomputation and RealTime under different number of mix nodes and users for 100 times. The test results are shown in Table II. We can see that PoWVote spends more time than other algorithms. In order to improve the operating efficiency of BCMIX, we elect mix nodes at regular intervals.

## VI. SECURITY ANALYSIS

According to the proposed additive homomorphism mix-net protocol and basic components of proof-of-work blockchains,

<sup>2</sup><https://github.com/arthurgervais/Bitcoin-Simulator>

<sup>3</sup><https://github.com/byronknoll/cmix>

<sup>4</sup><https://github.com/bitcoin-core/secp256k1>

TABLE II  
TIME COST (IN RM S) FOR DIFFERENT NUMBER OF MIX NODES AND USERS TO EXECUTE THE ALGORITHMS

Time Alg	Number	Mix Node	3	5	7	9								
		User	10	50	100	10	50	100	10	50				
PoWVote			29.62	28.90	30.12	29.77	31.23	29.35	29.00	30.27	32.14	31.36	30.80	29.79
IPSharding			0.65	0.65	0.65	0.93	0.93	0.93	1.46	1.46	1.46	1.88	1.88	1.88
Key Exchange			0.60	0.83	1.07	0.69	1.03	1.47	0.77	1.26	1.86	0.86	1.48	2.26
Precomputation			0.33	0.33	0.33	0.47	0.47	0.47	0.62	0.62	0.62	0.81	0.81	0.81
RealTime			0.03	0.11	0.26	0.06	0.19	0.37	0.12	0.23	0.43	0.16	0.29	0.49

We fix the difficulty  $D = 1 \times 10^{11}$  and stipulate the block generation time  $T = 30s$ . We can change the division strategy in Figure 6 to control the number of mix nodes.

TABLE III  
SUMMARY COMPARISON OF ATTACK RESILIENCE AND PERFORMANCE

Technology	Attacks						Performance	
	Traffic analysis	DDoS	Tagging	MitM	Sybil	Bandwidth	Latency	
BCMIX	Blockchain Re-encryption	✓	✓	✓	✓	✓	✓	✓
cMix	Mix Re-encryption	✓	✗	✗	✗	✓	✓	✓
Tor	Mix Multilayer Encryption	✗	✗	✗	✗	✓	✓	✓
BAR	Multicast/Broadcast	✓	✗	✗	✗	✓	✓	✗
AnonPubSub	Probabilistic Forwarding	✓	✗	✓	✗	✗	✗	✗
Tarzan	Peer-to-peer	✓	✓	✓	✗	✓	✓	✓
DiceMix	CoinJoin	✗	✗	✓	✗	✓	✓	✓

✓ - The system is secure against this attack or the system performs well.

✗ - The system is vulnerable to this attack or the system is not performing well.

\ - The system does not involve this attack of the authors did not mention the related performance.

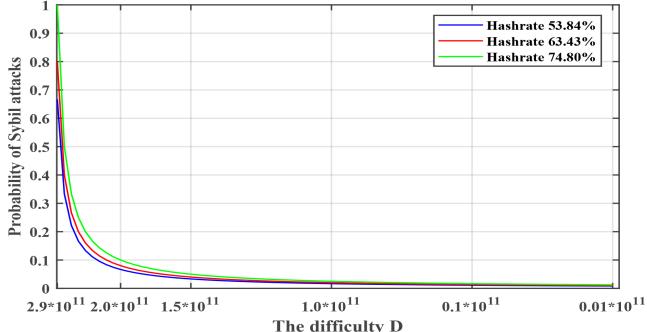


Fig. 7. The relationship between difficulty and probability to launch the Sybil attacks.

BCMIX system can satisfy all the security requirements described in Section III-C. Table III summarizes a comparison among different anonymous systems.

- **Resistance to Sybil Attacks.** We indicate that an attacker implement Sybil attacks successfully means the attacker take control of all mix nodes. To simulate a process of Sybil attacks, we assume that an IP address of mining pools in Table IV represents an identity, and all identities created by a mining pool share the mining pool's computing power equally. Figure 7 shows the requirements in terms of difficulty (computing power), in order for an attacker with different computing power to successfully launch Sybil attacks. Figure 7 indicates that, when the difficulty  $D$  is fixed, an attacker with higher computing power have a higher probability of successfully launching

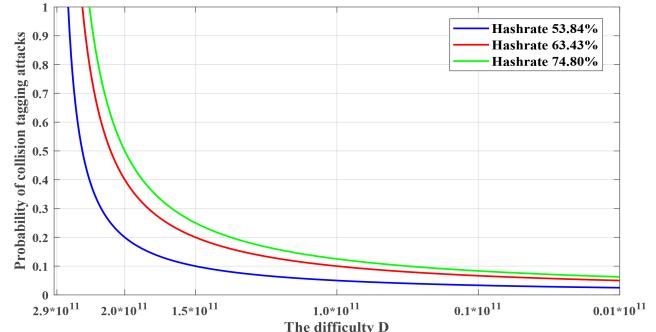


Fig. 8. The relationship between difficulty and probability to launch the collision tagging attacks.

Sybil attacks than attackers with lower computing power. And if we set the difficulty  $D$  to be very small, the probability that an attacker successfully launching Sybil attacks is almost zero.

- **Resistance to Collision Tagging Attacks.** As we mention in Section II C, an attacker compromise the last mix node  $N_l$  and any mix node  $N_i$  to launch collision tagging attacks. In this case, we can treat a collision tagging attack as a special form of a Sybil attack. We illustrates the relationship between the difficulty  $D$  and the probability of launching collision tagging attacks in Figure 8. Similar to Sybil attacks, we can set the difficulty  $D$  to be very small to resist the collision tagging attacks.

- **Sender Anonymity.** Suppose that an adversary  $\mathcal{A}$  can compromise  $\beta - 2$  users and  $n - 1$  nodes. Let  $U_x$  and

$U_y$  denote any two honest users and let  $N_i$  be the only honest mix node. The sender anonymity property guarantees that the adversary cannot distinguish the messages from the two honest users. We design the following experiments for a  $\mathcal{PPT}$  adversary  $\mathcal{A}$ .

$EXP_{an}^b(\Delta, \mathcal{A}, \lambda)$ :

$\mathcal{A}^{\text{Setup}}$ :

$$(\mathbf{Q}, K_x, K_y, K_{m,m \neq x,m \neq y}^{\mathcal{A}}) \leftarrow \text{Setup}(1^\lambda),$$

$\mathcal{A}^{\text{Precomputation}}$ :

$$\begin{aligned} (\mathcal{E}_{\mathcal{Q}}(\mathbf{R}_{\mathcal{A}}) \cdot \mathcal{E}_{\mathcal{Q}}(\mathbf{r}_{N_i})) &\leftarrow \text{Preprocess}(\mathbf{r}), \\ \mathcal{E}_{\mathcal{Q}}(\mathbf{\Pi}_n(\mathbf{R}_n) \cdot \mathbf{S}_n) &\leftarrow \text{Mix}(\mathbf{s}, \pi), \\ (\mathbf{\Pi}_n(\mathbf{R}_n) \cdot \mathbf{S}_n) &\leftarrow \text{Postprocess}(D_{d_i}), \end{aligned}$$

$\mathcal{A}^{\text{RealTime}}$ :

$$\begin{aligned} (m_b \cdot \mathbf{r}_{N_i}, m_{1-b} \cdot \mathbf{r}_{N_i}) &\leftarrow \text{Preprocess}(K_x \cdot m_b, K_y \cdot m_{1-b}), \\ \pi_{N_i}(m_b \cdot \mathbf{r}'_{N_i} \cdot s_{N_i}, m_{1-b} \cdot \mathbf{r}'_{N_i} \cdot s_{N_i}) &\leftarrow \text{Mix}, \\ (m'_b, m'_{1-b}) &\leftarrow \text{Postprocess} \end{aligned}$$

The adversary's advantage in the experiments is:

$$|\Pr[EXP_{an}^0(\Delta, \mathcal{A}, \lambda) = 1] - \Pr[EXP_{an}^1(\Delta, \mathcal{A}, \lambda) = 1]|.$$

**Definition 7. (Anonymity).** An additive homomorphism mix-net protocol  $\Delta := (\text{Setup}, \text{Precom}, \text{RealTime})$  maintains anonymity if the advantage of the adversary in the anonymity game is negligible.

**Theorem 4.** If  $\mathcal{E}$  is a ECDLP-secure additive homomorphism encryption scheme, ECDH is a ECDHP-secure key exchange protocol and a non-interactive commitment scheme COM is perfectly-hiding, then BCMIX satisfies anonymity defined in Definition 7.

*Proof.* (Sketch). We prove the security of BCMIX by reduction from the security of the encryption system  $\mathcal{E}$ . Without loss of generality, we assume that the adversary  $\mathcal{A}$  can compromise  $\beta - 2$  users and  $n - 1$  nodes. Let  $U_x$  and  $U_y$  denote any two honest users and let  $N_i$  be the only honest mix node.

In the setup phase,  $\mathcal{A}$  can control all the shared keys except  $K_x$ ,  $K_y$ . In the precomputation phase,  $\mathcal{A}$  gets command of random values  $\mathbf{R}_{\mathcal{A}}, \mathbf{S}_{\mathcal{A}}$  and random permutations  $\mathbf{\Pi}_{\mathcal{A}}$  except  $\mathbf{r}_{N_i}, \mathbf{s}_{N_i}, \pi_{N_i}$ , the corresponding ciphertext  $\mathcal{E}(\mathbf{r}_{N_i})$  and the decrypt share  $D_{N_i}$ . In the real time phase, users send blind messages to mix networks in the form of  $\mathbf{M} \times \mathbf{K}$ . The adversary  $\mathcal{A}$  can parse the blind messages as  $(K_x \cdot m_b, K_y \cdot m_{1-b})$ . During the mixing process of real time phase,  $\mathcal{A}$  decrypt the mixed messages and obtain the mixed messages  $(m_b \cdot \mathbf{r}_{N_i} \cdot s_{N_i}, m_{1-b} \cdot \mathbf{r}_{N_i} \cdot s_{N_i})$ . Finally,  $\mathcal{A}$  observe the plaintext messages of the form  $(m'_b, m'_{1-b})$ .

We assume a challenger  $\mathcal{C}$  assigns the blind messages  $(K_x \cdot m_0, K_y \cdot m_1)$ , where  $m_0 = m_b$  and  $m_1 = m_{1-b}$  as determined by a random bit  $b$ . An adversary holding  $(m_b, m_{1-b})$  predicts the association between  $(K_x \cdot m_0, K_y \cdot m_1)$  and  $(m'_b, m'_{1-b})$ . In the end, if the anonymity game adversary  $\mathcal{A}$  predicts the bit  $b$  correctly, we can infer that  $\mathcal{A}$  can calculate  $K_x$  and  $K_y$  which break the ECDHP-secure of the underlying key exchange system. Thus  $|\Pr[EXP_{an}^0(\Delta, \mathcal{A}, \lambda) = 1] - \Pr[EXP_{an}^1(\Delta, \mathcal{A}, \lambda) = 1]| \leq negl_{ECDHP}$ . Similarly, we conclude that  $|\Pr[EXP_{an}^0(\Delta, \mathcal{A}, \lambda) = 1] - \Pr[EXP_{an}^1(\Delta, \mathcal{A}, \lambda) = 1]| \leq negl_{ECDLP}$ . This concludes the proof.  $\square$

**• Resistance to MitM attacks.** In BCMIX we leverage the gossip protocol to spread messages. We assume that an attack cannot control all access networks of a blockchain node. According to [34], this is reasonable since none instances of eclipse attacks have arisen in reality up to now. Besides, many blockchain communities have fixed this vulnerability [35]. In this case, an adversary  $\tilde{\mathcal{A}}$  attempts to simulate an elected mix node to deceive a user  $\tilde{U}$ . At the same time, other nodes connected with  $\tilde{U}$  inform  $\tilde{U}$  the latest elected mix node sets, thus  $\tilde{U}$  identities  $\tilde{\mathcal{A}}$  as an adversary.

**• Single point of failure Resilience.** Nodes in BCMIX are networking dynamically. Once an elected mix node  $N_i$  loses the response for a period of time, the next mix node  $N_{i+1}$  would inform other nodes that  $i$  is crashed. Then the other nodes validate the situation of node  $i$ , and begin a new networking process if  $i$  goes down, or identify node  $N_{i+1}$  as a malicious node if  $i$  works normally.

**• Resistance to Other Attacks.** BCMIX can also resist the following attacks.

- a) Replay attacks. An attacker may retransmitting a message  $m$  from a previous session. Then the attacker compare the mixed message sets with the previous message sets which contain  $m$ , thus the attacker can associate the egress messages with the ingress messages. Since the random values and permutations are never reused, thus BCMIX resists replay attacks.
- b) Traffic analysis attacks. In connection-based systems such as Tor, attackers can distinguish between two different paths in the free mix network by counting packages and timing communication. Since BCMIX is a message-based system which batches and permutes messages during the transmitting process, attackers can not distinguish and analyze the blind messages, thus BCMIX resists traffic analysis attacks.
- c) Intersection attacks and statistical disclosure attacks. These attacks utilize information given by observing mix networks where the users can freely choose the mix node for their messages. Since BCMIX adopts a fixed cascade of mix nodes every round, thus BCMIX is not susceptible to these attacks.

## VII. CONCLUSION

In this paper, we achieved a dynamic self-organizing mix anonymous system. With blockchain technology, we elect mix nodes from public, dynamic blockchain miners. Before constructing BCMIX, we proposed BCMN protocol with the formal security models. Building on the proposed protocol, we designed a transaction-based key exchange scheme and proposed our BCMIX. Then we illustrated BCMIX can satisfy the relevant security requirements. After that we demonstrated

experimentally that BCMIX is resistant to the attacks proposed in this paper. Finally, we evaluated the performance of the prototype with the real world data and compared BCMIX with some latest anonymous systems. The results suggested that BCMIX is practical for real world deployment.

A follow-on work is to find a solution for recipient anonymity, which would improve the anonymity ability of our system. We believe that building a bidirectional anonymous system allow us to identity additional features and properties.

### VIII. APPENDIX A.

The detailed description of cMix protocol is as follows.

**Setup phase.** The mix nodes establish their decryption share  $X_i$ , and the public key  $y$  is computed. Each user  $A_j$  will individually establish a symmetric key  $k_{i,j}$  with each mix node  $N_i$  in the network. The mix nodes draw their random values  $\vec{r}_i$  and  $vect_i$  for the  $n$  slots.

**Precomputation phase.** The goal in this phase is to perform the public-key operations that is needed in the real time phase.

*Step 1-Preprocessing:* Mixnode  $N_i$  computes  $\mathcal{E}(\vec{r}_i^{-1})$ , and send their calculated vector to the network handler. The network handler then computes  $\mathcal{E}(\vec{R}_h^{-1}) = \prod_{i=1}^h \mathcal{E}(\vec{r}_i^{-1})$ .

*Step 2-Mixing:*  $N_i (i = 1, \dots, h-1)$  computes and sends the following to  $N_{i+1}$

$$\mathcal{E}(\Pi_i(\vec{R}_h^{-1}) \times \vec{T}_i^{-1}) = \begin{cases} \pi_1(\mathcal{E}(\vec{R}_h^{-1})) \times \mathcal{E}(\vec{t}_1^{-1}) & i = 1 \\ \pi_i(\mathcal{E}(\Pi_{i-1}(\vec{R}_h^{-1}) \times \vec{T}_{i-1}^{-1})) \times \mathcal{E}(\vec{t}_i^{-1}) & 1 < i \leq h \end{cases}$$

$N_h$  finally computes:  $(\vec{C}_1, \vec{C}_2) = \mathcal{E}((\vec{R}_h \times \vec{T}_h)^{-1}) = \pi_h(\mathcal{E}((\vec{R}_h^{-1}) \times \vec{T}_{h-1}^{-1})) \times \mathcal{E}(\vec{t}_h^{-1})$ .  $N_h$  sends  $\vec{C}_1$  to the other mix nodes and store  $\vec{C}_2$  locally for use in the real time phase.

*Step 3-Postprocessing:* Mixnode  $N_i$  use their decryption share  $X_i$  to decrypt the vector of random components they received in the previous step;  $\mathcal{D}_i(\vec{C}_1) = \vec{C}_1^{-X_i}$ . They publish a commitment to their calculated decryption share.

**Real time phase.** In this phase, the senders are involved.  $A_j$  constructs a blinded message  $m_j \times K_j^{-1}$ . The blinded messages are the input to the protocol, and they are combined by the network handler to yield the vector  $\vec{m} \times \vec{K}^{-1}$ .

*Step 1-Preprocessing:* Every mix node  $N_i$  calculates  $\vec{k}_i \times \vec{r}_i$ , and sends the resulting vector to the network handler. The network handler then computes  $\vec{m} \times \vec{R}_h = \vec{m} \times \vec{K}^{-1} \times \prod_{i=1}^h \vec{k}_i \times \vec{r}_i$ , hence the  $\vec{K}^{-1}$  vector is replaced with the random  $r$  values of each mix node.

*Step 2-Mixing:*  $N_i$  computes and sends the following to  $N_{i+1}$ :

$$(\Pi_i(\vec{m} \times \vec{R}_h) \times \vec{T}_i) = \begin{cases} \pi_1(\vec{m} \times \vec{R}_h) \times \vec{t}_1 & i = 1 \\ \pi_i(\Pi_{i-1}(\vec{m} \times \vec{R}_h) \times \vec{T}_{i-1}) \times \vec{t}_i & 1 < i < h \end{cases}$$

Mix node  $N_h$  computes  $\Pi_h(\vec{m} \times \vec{R}_h) \times \vec{T}_h = \pi_h(\Pi_{h-1}(\vec{m} \times \vec{R}_h) \times \vec{T}_{h-1}) \times \vec{t}_h$ .  $N_h$  commits to this vector and sends the commitment to the remaining mix nodes.

*Step 3-Postprocessing:* When mix node  $N_i (i = 1, \dots, h-1)$  receive the commitment from  $N_h$ , they send their decryption share  $\mathcal{D}_i(\vec{C}_1)$  computed in the precomputation phase to the network handler. The last mix node computes and send the following to the network handler:  $\Pi_h(\vec{m} \times \vec{R}_h) \times \vec{T}_h \times \vec{C}_2 \times \prod_{i=1}^h \mathcal{D}_i(\vec{C}_1) = \Pi_h(\vec{m} \times \vec{R}_h) \times \vec{T}_h \times (\Pi_h(\vec{R}_h) \times \vec{T}_h)^{-1} = \Pi_h(\vec{m})$

The network handler outputs  $\Pi_h(\vec{m})$ , that is a permutation of the input message.

### IX. APPENDIX B.

The distribution of mining pools in the real world. To simulate a process of Sybil attacks, we assume that an IP address of mining pools in Table IV represents an identity, and all identities created by a mining pool share the mining pool's computing power equally.

TABLE IV  
NOTATIONS OF THE REVISED MIX-NET PROTOCOL

Pool Name	IP Address	Hashrate	Proportion
		124(EH/s)	100%
f2pool	203.107.32.162	21.1048	17.02%
Poolin	47.75.234.12	19.716	15.9%
BTC.com	117.24.1.243, 117.24.1.239 117.24.1.238, 117.24.1.242	16.1076	12.99%
AntPool	47.94.135.145	13.95	11.25%
ViaBTC	116.211.155.211, 123.155.158.10	7.8988	6.37%
Huobi.pool	47.93.94.105	7.626	6.15%
58COIN&1THash	39.98.72.224	6.4728	5.22%
SlushPool	104.26.5.102, 104.26.4.102 172.67.74.105	5.6792	4.58%
OKExPool	208.43.170.231	4.9724	4.01%
unknown	47.93.94.105	4.7244	3.81%
BTC.TOP	123.56.208.222	3.9804	3.21%
BytePool	58.218.215.133	2.2816	1.84%
Binance Pool	13.248.150.68, 76.223.2.151 104.26.5.32, 104.26.4.32 172.67.70.128	2.0832	1.68%
BitFury	104.26.5.32, 104.26.4.32 172.67.70.128	2.0336	1.64%
Lubian.com	47.56.109.242	1.922	1.55%
NovaBlock	104.26.11.113, 104.26.10.113 172.67.70.128	1.488	1.20%
SpiderPool	47.52.126.9	0.62	0.5%
WAYI.CN	47.103.164.189	0.5459	0.44%
Bitcoin.com	104.18.26.217, 104.18.27.217	0.4092	0.33%
MiningCity	23.218.94.192, 23.32.241.177	0.124	0.1%
OKKONG	47.96.193.193	0.062	0.05%
TATMAS Pool	104.18.40.151, 172.67.148.229 104.1841.151	0.0496	0.04%
BitClub	213.173.105.14	0.0496	0.04%
Sigmapool.com	18.156.81.156	0.0372	0.03%
KanoPool	45.77.7.149	0.0124	0.01%
Solo CK	51.81.56.15	0.0124	0.01%

### REFERENCES

- [1] J. Sanders and D. Patterson, “Facebook data privacy scandal: A cheat sheet,” 2019.
- [2] D. R. Hayes, C. Snow, and S. Altuwajiri, “Geolocation tracking and privacy issues associated with the uber mobile application,” in *Proceedings of the Conference on Information Systems Applied Research ISSN*, vol. 2167, 2017, p. 1508.
- [3] N. Alexopoulos, A. Kiayias, R. Talviste, and T. Zacharias, “McMix: Anonymous messaging via secure multiparty computation,” in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1217–1234.
- [4] D. L. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms,” *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.
- [5] T. Lu, Z. Du, and Z. J. Wang, “A survey on measuring anonymity in anonymous communication systems,” *IEEE Access*, vol. 7, pp. 70 584–70 609, 2019.
- [6] G. Danezis and C. Diaz, “A survey of anonymous communication channels,” Technical Report MSR-TR-2008-35, Microsoft Research, Tech. Rep., 2008.
- [7] M. Edman and B. Yener, “On anonymity in an electronic society: A survey of anonymous communication systems,” *ACM Computing Surveys (CSUR)*, vol. 42, no. 1, pp. 1–35, 2009.
- [8] J. Boyan, “The anonymizer-protecting user privacy on the web,” 1997.
- [9] E. Gabber, P. B. Gibbons, D. M. Kristol, Y. Matias, and A. Mayer, “Consistent, yet anonymous, web access with lpwa,” *Communications of the ACM*, vol. 42, no. 2, pp. 42–47, 1999.

- [10] D. Chaum, D. Das, F. Javani, A. Kate, A. Krasnova, J. De Ruiter, and A. T. Sherman, “cmix: Mixing with minimal real-time asymmetric cryptographic operations,” in *International Conference on Applied Cryptography and Network Security*. Springer, 2017, pp. 557–578.
- [11] S. Prusty, B. N. Levine, and M. Liberatore, “Forensic investigation of the oneswarm anonymous filesharing system,” in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 201–214.
- [12] M. Sherr, A. Mao, W. R. Marczak, W. Zhou, B. T. Loo, and M. A. Blaze, “A<sup>3</sup>: An extensible platform for application-aware anonymity,” 2010.
- [13] J. Kong and X. Hong, “Anodr: anonymous on demand routing with untraceable routes for mobile ad-hoc networks,” in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, 2003, pp. 291–302.
- [14] P. Syverson, R. Dingledine, and N. Mathewson, “Tor: The second-generation onion router,” in *Usenix Security*, 2004, pp. 303–320.
- [15] M. J. Freedman and R. Morris, “Tarzan: A peer-to-peer anonymizing network layer,” in *Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 193–206.
- [16] M. Conti, N. Dragoni, and V. Lesyk, “A survey of man in the middle attacks,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.
- [17] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Manubot, Tech. Rep., 2019.
- [18] M. Gomulkiewicz, M. Klonowski, and M. Kutyłowski, “Onions based on universal re-encryption-anonymous communication immune against repetitive attack,” in *International Workshop on Information Security Applications*. Springer, 2004, pp. 400–410.
- [19] O. Pereira and R. L. Rivest, “Marked mix-nets,” in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 353–369.
- [20] D. Chaum, “The dining cryptographers problem: Unconditional sender and recipient untraceability,” *Journal of cryptology*, vol. 1, no. 1, pp. 65–75, 1988.
- [21] P. Kotzanikolaou, G. Chatzisofroniou, and M. Burmester, “Broadcast anonymous routing (bar): scalable real-time anonymous communication,” *International Journal of Information Security*, vol. 16, no. 3, pp. 313–326, 2017.
- [22] C. Egger, J. Schlumberger, C. Kruegel, and G. Vigna, “Practical attacks against the i2p network,” in *International workshop on recent advances in intrusion detection*. Springer, 2013, pp. 432–451.
- [23] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis, “The loopix anonymity system,” in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1199–1216.
- [24] R. Jansen, F. Tschorisch, A. Johnson, and B. Scheuermann, “The sniper attack: Anonymously deanonymizing and disabling the tor network,” Office of Naval Research Arlington VA, Tech. Rep., 2014.
- [25] T. Chothia and K. Chatzikokalakis, “A survey of anonymous peer-to-peer file-sharing,” in *International Conference on Embedded and Ubiquitous Computing*. Springer, 2005, pp. 744–755.
- [26] T. Ruffing, P. Moreno-Sánchez, and A. Kate, “P2p mixing and unlinkable bitcoin transactions.” in *NDSS*, 2017, pp. 1–15.
- [27] J. Han and Y. Liu, “Mutual anonymity for mobile p2p systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 8, pp. 1009–1019, 2008.
- [28] N. Koblitz, “Elliptic curve cryptosystems,” *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.
- [29] Y. Luo, X. Ouyang, J. Liu, and L. Cao, “An image encryption method based on elliptic curve elgamal encryption and chaotic systems,” *IEEE Access*, vol. 7, pp. 38 507–38 522, 2019.
- [30] T. Jager, “Verifiable random functions from weaker assumptions,” in *Theory of Cryptography Conference*. Springer, 2015, pp. 121–143.
- [31] J. Garay, A. Kiayias, and N. Leonardos, “The bitcoin backbone protocol: Analysis and applications,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 281–310.
- [32] J. Yu, D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo, “Repucoin: Your reputation is your power,” *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1225–1237, 2019.
- [33] S. Hohenberger, S. Myers, R. Pass *et al.*, “Anonize: A large-scale anonymous survey system,” in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 375–389.
- [34] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, “Eclipse attacks on bitcoins peer-to-peer network,” in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 129–144.
- [35] Y. Marcus, E. Heilman, and S. Goldberg, “Low-resource eclipse attacks on ethereum’s peer-to-peer network.” *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 236, 2018.

**Michael Shell** Biography text here.

PLACE  
PHOTO  
HERE

**John Doe** Biography text here.

**Jane Doe** Biography text here.