BD-SAS: Enabling Dynamic Spectrum Sharing in Low-trust Environment

Yang Xiao, *Member, IEEE,* Shanghao Shi, *Student Member, IEEE,* Wenjing Lou, *Fellow, IEEE,* Chonggang Wang, *Fellow, IEEE,* Xu Li, Ning Zhang, *Member, IEEE,* Y. Thomas Hou, *Fellow, IEEE,* Jeffrey H. Reed, *Fellow, IEEE*

***Abstract*—The spectrum access system (SAS) designated by the FCC follows a centralized server-client model where a spectrum user registers with one SAS service provider for spectrum allocation and other spectrum management functions. This model, however, is vulnerable to adversarial influence on individual SAS servers, causing concerns over system reliability and trustworthiness, especially when the ecosystem embraces a growing base of SAS service providers and heterogeneous user requirements. In this paper, we propose a blockchain-based decentralized SAS architecture called BD-SAS to provide SAS services securely and efficiently, without assuming trust in individual SAS servers. BD-SAS is backward compatible with the existing SAS infrastructure and supports the automatic execution of key spectrum management functions. A global blockchain network (G-Chain) is used for spectrum regulatory tasks while localized blockchain networks (L-Chains) are instantiated in individual spectrum zones for automating spectrum access assignment and other spectrum management activities. To further achieve security against an adaptive adversary, BD-SAS integrates a SAS server reshuffle scheme to resist adaptive corruptions on individual SAS servers. We implemented a BD-SAS prototype with practical blockchain platforms. Evaluation results demonstrate the feasibility and responsiveness of our system, wherein a spectrum access assignment can be finalized at the level of seconds.**

***Index Terms*—Radio spectrum management, dynamic spectrum access, distributed information systems, security, fault tolerance**

I. Introduction

SPECTRUM is the most important yet scarce resource for wireless communication and sensing. Spectrum regulators, such as the Federal Communications Commission (FCC) and the National Telecommunications and Information Administration (NTIA) in the United States[1], have opened up exclusively licensed bands (e.g., sub-GHz TVWS and 3.5 GHz CBRS) and unlicensed bands (e.g., 6 GHz to mmWave) for civilian use on a sharing basis. To protect the access rights of incumbent users and ensure a fair dynamic spectrum sharing (DSS) process, the FCC visioned a spectrum access system (SAS) in its rulings on the 3.5GHz Citizens Broadband Radio Service (CBRS) band [4]. Since then, the Wireless Innovation Forum (WInnForum) has been leading the standardization effort on SAS for CBRS, covering a variety of spectrum management functions from incumbents protection and spectrum access assignment to coordination between different SAS service providers. Among its core tasks, a SAS service provider generates spectrum access assignment in response to requests from its registered users [5] and different SAS service providers also need to periodically communicate with each other to synchronize service state in common service regions [6], [7].

A. Motivation for Decentralized SAS

As the NTIA and FCC plan to open up or re-purpose more spectrum for shared use, such as 3.4-3.55 GHz, 3.7-4.2 GHz (C Band), 5.9 GHz (DSRC), and lower 37 GHz bands, the regulators are soliciting comments and DSS solutions for these bands [8]. The SAS model, which has seen initial success in the CBRS band, poses a strong candidate for that purpose, as the existing SAS service providers can conveniently extend their spectrum management service to the other bands. However, the existing SAS model faces major challenges of centralized trust and limited scalability.

First, it follows the centralized server-client paradigm and assumes that spectrum users can place absolute trust in the SAS service provider they have subscribed to. This assumption, however, is questionable in the evolving DSS landscape where there can be an increasing number of SAS service providers servicing heterogeneous bands and users. There will be an escalated risk that individual SAS service providers do not operate faithfully or follow the exact spectrum allocation rules. A malfunctioning or malicious SAS service provider would bring devastating outcomes to its subscribed spectrum users. Second, according to the WInnForum specifications [7], SAS service providers periodically communicate with each other to synchronize service states based on request-response. The current inter-SAS synchronization process, called Cooperative Periodic Activities among SASs (CPAS) [6], is performed on a daily basis, contributing to a long wait (next day) for spectrum usage. From the security perspective, it is

A preliminary version of this work has appeared as Chapter 6 in Yang Xiao's Ph.D. dissertation [1].

Yang Xiao is with the Department of Computer Science, University of Kentucky, Lexington, KY, USA (email: xiaoy@uky.edu).

Shanghao Shi and Wenjing Lou are with the Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA (email: {shanghaos,wjlou}@vt.edu).

Chonggang Wang and Xu Li are with InterDigital, Princeton, NJ, USA (email: {chonggang.wang,xu.li}@interdigital.com).

Ning Zhang is with the Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO, USA (email: zhang.ning@wustl.edu).

Y. Thomas Hou and Jeffrey H. Reed are with the Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA (email: {thou,reedjh}@vt.edu).

[1] The FCC is an independent government agency that regulates the non-Federal use of spectrum and is overseen by the US Congress [2]; the NTIA manages the Federal use of spectrum and is responsible for advising the President on telecommunications and information policy issues [3].

also defenseless against a malicious SAS service provider who disseminates false or tampered records, which could bring chaos to honest SAS service providers. Third, as we anticipate more players to join the DSS ecosystem to either provide or use SAS service, the server-client paradigm would bring excessive regulatory overhead that could be undesirable to the regulators. Meanwhile, the FCC has voiced support for a market-driven approach to enabling more flexible and self-managed spectrum sharing modes, including PA license leasing, SAS-managed spectrum exchanges, secondary spectrum markets, and other SAS value-added services [4], [9]. In all cases, a new SAS paradigm that is secure by design, scalable, and self-governed is highly sought-after.

Prospect of Decentralization. From the security perspective, spectrum users must be guaranteed reliable spectrum management service even if individual SAS servers are not trustworthy. From the performance perspective, there needs an efficient inter-SAS coordination mechanism that allows different SAS service providers to quickly synchronize service states without involving a central mediator. In this regard, we consider a decentralized and collectively managed SAS paradigm desirable in that spectrum access assignments are finalized via consensus by a group of SAS servers who can synchronize their service states in a timely manner. Users and SAS servers alike are encouraged to participate in the process honestly to realize the high autonomy of spectrum management. The decentralized, self-governed paradigm provides a natural ground for innovative value-added services while minimizing regulatory overhead. For practical purposes, this decentralized SAS model should be backward compatible with the existing SAS in terms of participants and task model, and also does not degrade SAS service quality, such as generating spectrum assignment with low latency.

Blockchain as a Potential Solution. Blockchain emerged as a secure-by-design technology for enabling decentralized payment networks. Based on a cryptography-hardened transactional model and consensus-based consistency mechanisms, blockchain enables trustworthy transaction processing and ledger keeping among mutually distrustful participants, even if a certain portion of them may behave maliciously [10]. The decentralization, transparency, consensus-based security, immutable ledger keeping, and support for the self-executing smart contracts have made blockchain an ideal technology to enable decentralized, automated spectrum management [11]. The FCC has indicated its interest in employing blockchain and distributed ledger technology for future spectrum sharing systems [12], [13]. Since then multiple blockchain-based SAS solutions have been proposed [14]–[16]. Still, there lacks a concrete solution in the literature on how to decentralize the SAS amid malicious SAS servers while being backward compatible with the existing SAS.

B. Our Contribution

To solve the challenges of service centralization and fault tolerance, building upon our previous vision [17], [18], we formulate a *decentralized SAS model* encompassing a participation model and a task model. The participants, namely regulators, SAS servers, local witnesses, and spectrum users, get involved in four tasks—user management, spectrum access assignment, record keeping, and regulation enforcement. We identify three key requirements that arise with decentralization: *fault tolerance* of all tasks when malicious SAS servers exist, *responsiveness* of spectrum access assignment, and *backward compatibility* with the existing SAS infrastructure which has been implemented for CBRS.

BD-SAS. We formally introduce the **B**lockchain-based **D**ecentralized **SAS** architecture dubbed BD-SAS, extending from our previous proof-of-concept design [18]. BD-SAS consists of two layers of smart contract-enabled blockchains: the G-Chain at the global scale for regulatory purposes and inter-SAS information exchange, and L-Chains at local regions for the actual spectrum access assignment. The G-Chain is participated by regulators, SAS servers, and witnesses, who maintain a unified record on spectrum regulations and digests of local SAS service states by curating a carefully designed G-Chain *regulatory contract* C_R. An L-Chain is dedicated to spectrum access management for a specific geographical region and participants include SAS servers who serve that region and witnesses who represent stable local spectrum users in that region. To enable automated spectrum assignment, a carefully designed *spectrum access contract* denoted C_{SA} is instantiated on the L-Chain and encodes the access request function. The function is invoked by a spectrum user and outputs an access assignment (or an inquiry response) indicating the allocated channels and conditions to use. Assignment results are finalized in the L-Chain ledger and open for auditing.

G-Chain is secure under the assumption that the majority of SAS servers are honest globally. To defend against powerful adversaries who can adaptively corrupt SAS servers to compromise a target L-Chain, we design a security mechanism called SAS server reshuffle to ensure that the SAS servers serving any L-Chain at any time are also majority-honest (i.e., THE same threat threshold as with G-Chain) with overwhelming probability. To minimize spectrum access assignment latency, i.e., for the requirement on responsiveness, we design the L-Chain workflow where the execution of spectrum access requests is separated from the total ordering of them, which are undertaken respectively by the SAS servers and witnesses. This separation scheme is essential to the responsiveness of the assignment process, as a user can start using the spectrum right after the SAS servers execute its request before the execution is finalized by local witnesses. And such response time is at most linear in the number of those SAS servers.

In summary, we make the following contributions in this paper:

- Visioning on the future spectrum sharing landscape, we propose a decentralized SAS model that allows spectrum users to access reliable SAS service without having to trust an individual SAS server. This model contains backward-compatible abstractions for system participants and tasks, and key security and performance requirements.
- We introduce the BD-SAS architecture to realize the decentralized SAS model. BD-SAS is the first blockchain-

based SAS solution that achieves decentralization of SAS service while being compatible with the existing SAS infrastructure and supporting automatic execution of SAS functions.
- Within BD-SAS, we design a cross-chain security mechanism, called the SAS server reshuffle procedure, to defend BD-SAS against an adaptive adversary who can exert Byzantine influence on individual SAS servers. Moreover, we leverage the separation of consensus and execution to achieve higher efficiency in spectrum assignment.
- We implemented a BD-SAS prototype using the Ethereum Rinkeby testnet (for emulating the G-Chain) and Hyperledger Fabric [19] (for realizing an L-Chain). Evaluation results demonstrate BD-SAS's capability of providing spectrum access assignment to users within five seconds under our most constraining network delay setting, showing BD-SAS's feasibility amid the tight timing regime of existing SAS operation.

Road map §II introduces the existing SAS model and several blockchain-based SAS designs related to our work. §III provides abstractions for SAS participants and tasks and main requirements for the decentralized SAS model. §IV introduces BD-SAS, a novel decentralized SAS solution along with its setups and operations. §V analyzes the system's security and performance properties. §VI introduces our prototype implementation. §VII evaluates the prototype's performance. §VIII discusses potential extensions to BD-SAS. Finally, §IX concludes the paper.

II. Background and Related Work

A. The Existing Spectrum Access System

The SAS concept was coined by the FCC as a new DSS paradigm to open up previously under-utilized spectrum while protecting the rights of incumbent users. It was legalized in the FCC's 2015 ruling on the 3.5GHz CBRS band [4]. Based on a three-tiered access model, the SAS is designated by the FCC for managing the shared access to the CBRS band while protecting the preemptive right of Incumbent Access (IA) users and the licensed privilege of Priority Access (PA) users. The General Authorized Access (GAA) users request spectrum access from the SAS but do not receive presumed interference protection.

Since 2016, the WInnForum has been leading the CBRS standardization effort, including specifications on SAS-CBSD interface [5] (CBSD stands for citizens broadband radio service devices), inter-SAS communication [7], communication security and PKI requirements [20], and rules for protecting incumbent operations [21]. Each SAS server, which is proprietary to a commercial administrator, maintains a database on the spectrum availability and receives CBSD registrations. The spectrum assignment process follows a server-client model and consists of three main request-response procedures—"Inquiry", "Access", and "Heartbeat". When a SAS server receives an access request from a registered CBSD, it responds with a "Grant" specifying the access assignment details, including allocated channel range, expiration time, and Heartbeat interval. Each CBSD with an active grant needs to send periodic Heartbeat requests to the SAS server and receive Heartbeat responses. Each Heartbeat response authorizes the CBSD to commence RF transmission using the granted channels until a certain transmit expiration time. When the transmit expiration time is reached without a new Heartbeat response (grant will be suspended) or the grant gets terminated, the CBSD should stop transmitting within 60 seconds [5]. To maximize spectrum utilization, the Heartbeat interval can be as tight as 30 seconds in commercial SAS implementations [22]. To facilitate coordination across different SAS servers, SAS servers communicate with each other and synchronize service states on a daily basis [6], [7]. This daily synchronization procedure called the Cooperative Periodic Activities among SASs (CPAS), requires all SAS servers to exchange full-dump CBSD records so that they may generate substantially similar allocations for the same CBSDs (i.e., in new Grants). It however usually leads to a long wait for the grant approval—except a small portion of the spectrum is reserved for same-day approval with limited transmission power, most grants are fully approved only after the next day's CPAS procedure [23].

B. Blockchain for Spectrum Management

Prior wisdom has explored the prospective use of blockchain technology for spectrum management. It is identified in [11], [25] that the key features of blockchain technology, namely decentralization, transparency, ledger immutability, and consensus-based security, are appealing to both spectrum users and regulators in the DSS scenarios. In particular, the FCC has been eyeing on blockchain's potential in enabling novel spectrum sharing and trading applications with minimal regulatory touch [9], [13]. Weiss *et al.* [11] provide a qualitative analysis of the pros and cons of blockchain technologies when applied to different spectrum-sharing modes according to the authors' previous typology [26]. Ariyarathna *et al.* [14] propose a digital-token-based spectrum access platform wherein a smart contract system is used by primary users as a trusted third-party service for advertising and leasing spectral tokens to secondary users. Grissa *et al.* [15] formulate a hierarchical blockchain framework called TrustSAS to enable privacy-preserving spectrum sharing among secondary users. Local blockchain networks are established among secondary users for spectrum query aggregation and response distribution while a global blockchain is used for general records keeping. Zhang *et al.* [16] propose a blockchain-enhanced spectrum sharing system in the CBRS band where PA users establish local blockchain ledgers that record spectrum usage rules for local GAA users. Similarly, *et al.* [24] propose a blockchain-assisted system aiming to improve the service for GAA users, where the PA users run a consensus mechanism to generate spectrum allocations for GAA users and use blockchain for storage.

In the schemes mentioned above, either a third-party smart contract platform [14], individual SAS servers [15], or a PA consortium is assumed to process allocation requests from users [16], [24]. The first two paradigms are still susceptible to the security impact of single-point failures while the last paradigm is not compatible with the currently deployed SAS

TABLE I
COMPARISON OF BLOCKCHAIN-BASED DYNAMIC SPECTRUM SHARING SCHEMES

Scheme	Blockchain's role in spectrum sharing	Who curate blockchain	Fault-tolerant allocation	CBRS-SAS compatibility	SAS server coordination
Ariyarathna *et al.* [14]	Rule/Record keeping	Third party	No	No	(N/A)
Grissa *et al.* [15]	Record keeping	GAA users	No	Yes	No
Zhang *et al.* [16]	Allocation, record keeping	PA users	Yes	No	(N/A)
Li *et al.* [24]	Allocation, record keeping	PA users	Yes	No	(N/A)
BD-SAS (this work)	Allocation, record keeping	SAS servers	Yes	Yes	Yes

where PA users are customers of spectrum allocation service provided by SAS servers. As conceptualized in our previous vision [17], we believe that a decentralized and backward-compatible SAS model is highly desirable in that the security and efficiency of spectrum allocation come from the honest majority of SAS servers and the system can conveniently coexist with the current SAS in CBRS. A comparison of mentioned schemes and our scheme is shown in Table I.

III. SYSTEM MODEL

In this section, we describe the participants, main tasks, and key requirements of a decentralized SAS model. Crucially, the mode is designed to be compatible with the existing CBRS-based SAS implementation while general enough to accommodate other spectrum bands.

Geographical concepts. We first clarify the geographical concepts of our model. The *global* scale refers to the entire spectrum jurisdiction, such as the territory of the US. In contrast, the *local* scale refers to a specific *region*. All regions operate independently for local spectrum management while being subject to global regulations. The region concept is inclusive of CBRS "zone" which typically refers to a US county [7] and a region may consist of one or more contiguous CBRS zones.

A. Participants

We define four types of participants:

- **Regulator** is a government entity that publishes regulations on spectrum usage in its jurisdiction. Examples of regulators are the FCC and the NTIA in the US.
- **SAS Server** is a cloud server capable of providing regulation-compliant spectrum allocation service to spectrum users in certain regions. Each SAS server is managed by a SAS administrator, candidates of whom include the current ones in CBRS (e.g., Google, Federated Wireless, CommScope, Sony, Amdocs) and other potential cloud service providers.
- **Spectrum user** operates a certified RF device and acts as a local client to the SAS for spectrum access assignment. Typical spectrum users include private LTE base stations, 5G access points, and campus hot spots.
- **Witness** is an entity that participates in local spectrum management on behalf of its associated spectrum users. Candidates of witnesses include operators and proxies (as in CBRS) of spectrum users.

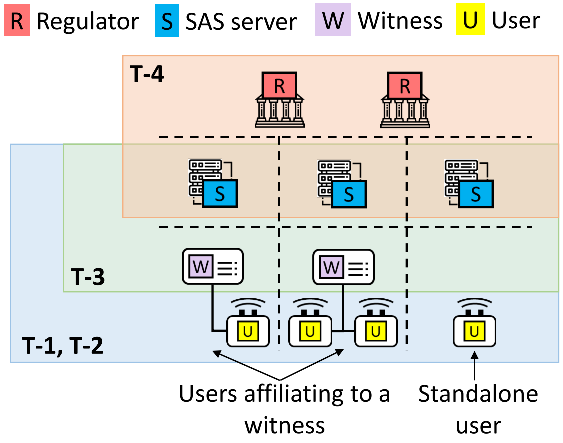


Fig. 1. Task involvement of participants in the decentralized SAS model. Dashed lines represent trust boundaries.

To ensure compatibility with both the existing and decentralized models, we make three assumptions about the participants. First, although multiple SAS servers can be managed by one administrator, every SAS server should operate with a unique identity when providing service to spectrum users. This is in line with the current SAS implementation [7]. We further require that for a specific region, at most one SAS server under each SAS administrator can provide service to that region. This is to enhance the fairness and fault tolerance of SAS service at the local level, as we will provide a realization in §IV-C. Second, a spectrum user can be either stand-alone or associated with a witness; a witness must be backed by one or more local spectrum users, as is shown in Figure 1. This arrangement gives flexibility to existing spectrum users for participating in local spectrum management, as an occasional user (e.g., a small-scale GAA user in CBRS) may not want to involve in SAS management other than being a pure client. Third, we exclude federal incumbent users such as naval radars and satellite ground stations from the spectrum user category as they have preemptive access rights and do not need the spectrum allocation service from the SAS. The signaling mechanism of incumbent appearance, as provided by the Environmental Sensing Capability (ESC) of the current SAS framework [21], is modeled as trusted broadcast messages coming from a regulator node.

B. Main Tasks

T-1. User registration. The system accepts registrations from spectrum users. The registration process validates the user-provided device information (including the device certificate, region, and RF capability) and creates a unique user ID that will be used in later communications and tasks.

T-2. Access assignment. The system generates spectrum access assignments (specifying location, channel range, expiration time, etc.) per user request. The calculation of the assignment follows a predefined interference model that consumes the current state of spectrum sharing. The system also allows a user to simply inquire about the real-time spectrum availability (e.g., on location and channel range) before it decides on operational parameters in its ensuing spectrum access request.

T-3. Record keeping. The system keeps an irreversible record of spectrum access requests and assignments. In the decentralized SAS model considered in this paper, this task dictates that all SAS servers and witnesses who have participated in the access management for a specific spectrum region should agree on a unified record for that region. This record serves three purposes—for synchronizing SAS service states (required by the existing SAS [7]), enabling regulatory audits, and underpinning any compensation scheme.

T-4. Regulation enforcement. The system allows a regulator to publish regulations on the above tasks. A regulation can be either long-term or short-term. Long-term regulations include the addition of spectrum regions, designation of a universal interference model, publication of priority access licensees (as in the CBRS case), etc. Short-term regulations include notification of incumbent user appearance, addition/removal of SAS servers, etc.

The task involvement for participants is shown in Figure 1. Importantly, regulators do not involve in the day-to-day spectrum management business (**T-1**, **T-2**, **T-3**) while spectrum users act as clients for **T-1** and **T-2**.

C. Key Requirements for Decentralization

In the current SAS model, **T-1** and **T-2** have been standardized as a server-client process [5] while **T-3** and **T-4** are trivial due to the trust on individual SAS servers. In comparison, the decentralized SAS model does not assume trust in any individual participant except the regulators. Spectrum users do not need to trust any individual SAS server, nor do SAS servers or witnesses trust each other. Moreover, the performance impact of decentralization should be kept in control. We identify the following key requirements:

R-1. Fault tolerance. All four tasks should be executed correctly given that a certain portion of SAS servers is faulty or malicious (i.e., "Byzantine"). The detailed threat model is given in §III-D. In specific, for any spectrum region, users should get correct assignments (**T-2**) and SAS servers and witnesses should keep a unified assignment record (**T-3**) as long as fewer than half of SAS servers can be Byzantine.

R-2. Responsiveness. The generation of an actionable access assignment per user request (**T-2**) should be swift, and on par with the existing SAS's server-client model. Here "actionable" means the assignment is complete and valid before being written into the record. This implies that the assignment latency should be less than linear in the number of SAS servers involved in the assignment generation. Moreover, the system should accommodate the increasing number of spectrum users without significantly compromising responsiveness.

TABLE II
LIST OF SYSTEM VARIABLES IN BD-SAS

Symbol	Definition
C_R	The Regulatory Contract on the G-Chain.
C_{SA}	Spectrum Access Contract on an L-Chain.
$\mathcal{SG}_i$	Group of SAS servers serving the i^{th} L-Chain.
N	Total number of SAS servers globally
F	Maximum number of Byzantine SAS servers globally
W_i	Size of the i^{th} L-Chain's Witness group
S_i	Size of $\mathcal{SG}_i$.
B	G-Chain block interval in seconds
T_i^E	i^{th} L-Chain's epoch length in B.
T_i^S	i^{th} L-Chain's shift length in B.
R_i	Number of epochs before a shift ends for the SAS server reshuffle procedure to start at the i^{th} L-Chain.

R-3. Backward compatibility. The system should be compatible with the existing SAS infrastructure and coexist with the server-client model when fulfilling **T-1** and **T-2**. This offers flexibility to SAS servers and spectrum users in their day-to-day operation, as they can gradually transition into the new decentralized model.

D. Threat Model

We assume individual SAS servers may suffer from Byzantine fault, i.e., arbitrarily deviating from their normal routine, due to server failure or adversarial corruption. The Byzantine fault model is also considered in the prior art of blockchain-based SAS [15] for modeling the decision-making of secondary users. Generally, the Byzantine model is inclusive of all potential server faults and is considered the most severe fault model in the literature of distributed computing [27] and blockchain systems [10]. It includes the case that a SAS server provides false or tampered information to peer SAS servers or spectrum users. We call unaffected SAS servers "honest". The adversarial corruption can be also adaptive in that a SAS server may act honestly at first and turn malicious at an arbitrary point. In all cases, we assume the honest SAS servers always take up more than 50% of SAS servers globally at any time. We assume witnesses of a region are self-organized and have the direct incentive to provide correct witness service to our system's local operation. To provide a reasonable design scope in this paper, we consider that they can run a local election process to choose the most reputable witnesses among themselves. Lastly, we assume the regulators, such as the FCC and NTIA in the US, are trusted.

IV. THE BD-SAS ARCHITECTURE

A. BD-SAS Overview

We introduce the *blockchain-based decentralized SAS* architecture, dubbed BD-SAS, to realize the aforementioned decentralized SAS model. BD-SAS consists of two layers of blockchain networks: a Global Chain (G-Chain) and region-specific Local Chains (L-Chains). We define *curator* as an entity that participates in the blockchain consensus and maintains a blockchain instance, and *client* as an entity that can create an account and issue transactions but does not participate in consensus. Following this definition, the G-Chain is a public blockchain to which anyone can be a client; but only regulators and SAS servers can be the curators, with the former in charge of curator permission control. In comparison, the L-Chain of any region j is a fully permissioned blockchain, with the local witnesses and a group of SAS servers (denoted $\mathcal{SG}_j$) as curators, while local spectrum users act as clients. $\mathcal{SG}_j$ is responsible for user management and spectrum access assignment. Specially, we call the witnesses who founded the L-Chain and assumed long-term managerial responsibility the *anchor witnesses*, such as the PA users in CBRS. Anchor witnesses also represent all other witnesses in the G-Chain. Anticipating the presence of an adaptive adversarial who can corrupt targeted SAS servers, $\mathcal{SG}_j$ needs to be periodically re-selected through a random reshuffle mechanism.

The data structures of the G-Chain and one L-Chain are shown in Figure 2. Both the G-Chain and L-Chain follow the account-balance model as in Ethereum [28] and support on-chain state machines that enable smart contract functionalities. The G-Chain's state contains the global account balances and a regulatory contract $\mathcal{C}_R$, which allows regulators to publish spectrum regulations and create new L-Chain profiles (**T-4**). $\mathcal{C}_R$ also encodes a record-keeping function that enables SAS servers to update local service states and to claim service compensations (**T-3**) and a multiparty function realizing random reassignment of SAS servers. We will later show that the latter function is essential for our system in achieving resilience against adaptive SAS server corruptions. The SAS servers update the L-Chain state for every *epoch*, which spans over T_i^E consecutive G-Chain blocks. Every epoch start of the L-Chain is marked by a beacon block, whose header includes an extra hash pointer to the most recent G-Chain block. The SAS server group $\mathcal{SG}_j$ is randomly re-selected for every *shift*, which spaces over T_i^S consecutive G-Chain block cycles. T_i^E and T_i^S are design parameters and fixed during the i^{th} L-Chain's bootstrapping, and T_i^S is a multiple of T_i^E. An L-Chain's state contains a spectrum access contract $\mathcal{C}_{SA}$ that documents local witnesses, spectrum users, and assigned SAS servers. $\mathcal{C}_{SA}$ also encodes the functions for user registration (**T-1**) and spectrum access assignment (**T-2**). For every epoch, the assigned SAS servers are responsible for conveying recent regulations to the L-Chain at the epoch start (**T-4**) and updating the local service state to the G-Chain at the epoch end (**T-3**).

Next, we focus on the practical design of G-Chain and L-Chain and elaborate on how the four tasks are fulfilled.

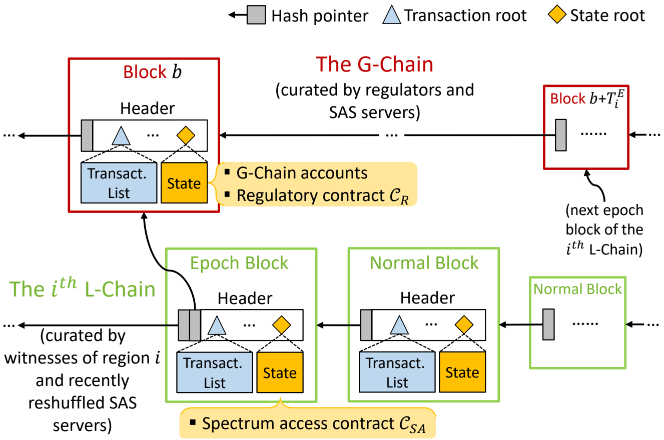

Hash pointer
Transaction root
State root
Block b
The G-Chain
(curated by regulators and SAS servers)
Block b+T_i^E
Header
…
……
Transact. List
State
▪ G-Chain accounts
▪ Regulatory contract $\mathcal{C}_R$
(next epoch block of the i^{th} L-Chain)
The i^{th} L-Chain
(curated by witnesses of region i and recently reshuffled SAS servers)
Epoch Block
Normal Block
Header
Transact. List
State
Normal Block
▪ Spectrum access contract $\mathcal{C}_{SA}$

Fig. 2. Ledger structures of the G-Chain and one L-Chain (each region has its own L-Chain) of BD-SAS.

B. G-Chain Operation

The G-Chain can build on existing blockchain implementations through two paradigms: (1) reusing an existing public blockchain system such as Ethereum or its testnets or (2) bootstrapping from a new consortium blockchain network. The former paradigm builds on the reliability of the public blockchain's smart contract functionality, while the latter gives freedom for choosing more efficient consensus protocols. We adopt the first paradigm for prototyping (see §VII). At the starting phase, regulators and an initial group of SAS servers start with a G-Chain block, where a regulator submits a G-Chain transaction to create $\mathcal{C}_R$. New SAS servers join the G-Chain network by connecting to the URL endpoints of existing regulators and SAS servers and passing TLS-based mutual authentication based on an existing public key infrastructure (PKI), which is conveniently provided in the existing SAS ecosystem [7], [20]. SAS servers can synchronize to the G-Chain ledger and interact with $\mathcal{C}_R$ after passing the mutual authentication procedure with a regulator.

Encoding $\mathcal{C}_R$. The pseudocode of $\mathcal{C}_R$ is shown in Algorithm 1. Its public variables include the profiles of regulators, SAS servers, and L-Chains, as well as interference model parameters and an incidence list for incumbent appearances. A regulator can update the above public variables by sending a transaction invoking the *Publish* function, which fulfills the regulation publication part of **T-4**. The *ResharePropose* and *ReshuffleConfirm* functions are parts of the SAS server reshuffle procedure as we elaborate in §IV-C.

Local Update Procedure. When concluding an epoch, the SAS servers in $\mathcal{SG}_i$ need to update their local service state onto the G-Chain, fulfilling the global part of **T-3**. To do so, they use a simple round-robin mechanism across epochs—the S_j SAS servers take turns to call the *LocalUpdate* function of $\mathcal{C}_R$ for every new epoch. The function updates the epoch, L-Chain state root *sroot*. Every anchor witness of L-Chain j checks the new updated *sroot* with their local version and calls *LocalUpdate* to provide its approval signature.

All G-Chain participants can send tokens to each other through transactions, akin to cryptocurrency payment. The anchor witnesses and SAS servers of an L-Chain can also use the local update procedure to claim compensation for their L-Chain spectrum access assignment service (not shown in Algorithm 1). In this paper, we consider the compensation scheme an important but standalone task to be addressed in a separate work.

C. SAS Server Reshuffle Procedure

We require the SAS server group $\mathcal{SG}_j$ for each L-Chain j to be randomly replaced for every shift (i.e., T_j^S G-chain

Algorithm 1: Regulatory Contract C_R Pseudocode (essential functions only)

```
var Regulators[], Servers[]
var LC[].{$desc$, Witnesses[], $SG$[], $shift, epoch, sroot$}
  //L-Chains profiles
var iModel.{$param1, param2, ...$} //Interference model
var Incidents.{$VacateOrders$[], ...}
var Lottery[][].{$hash, proof, shift, status$}
Function Init() //Contract creation by a regulator
    Initialize Regulators, Servers, iModel
Function Publish(regulation) //$\mathcal{S}$ represents the function
  caller
    if $\mathcal{S} \in$ Regulators then
        Update Servers[], LC[], iModel, or Incidents per
          regulation
Function ReshufflePropose($j$, $hash$, $proof$) //$j$ is the
  target L-Chain index
    if $\mathcal{S} \in$ Servers and
      $(system.block -$ LC$[j].shift) \in [0, T_j^S - T_j^E)$ and
      Lottery$[j][\mathcal{S}].shift \leq$ LC$[j].shift$ then
        Lottery$[j][\mathcal{S}].\{hash, proof, shift\} \leftarrow$
          $\{hash, proof,$ LC$[j].shift + T_j^S\}$
Function ReshuffleConfirm($j$, $selected$[]) //Invoked by
  L-Chain $j$'s witnesses
    if $\mathcal{S} \in$ LC$[j]$.Witnesses and
      $(system.block -$ LC$[j]$.shift$) \in [T_j^S - T_j^E, T_j^S)$ then
        if the majority of LC$[j]$.Witnesses confirm
          $selected$[] then
            LC$[j].SG[] \leftarrow selected[]$
            LC$[j].shift \leftarrow$ LC$[j].shift + T_j^S$
Function LocalUpdate($j$, $epoch$, $sroot$) //Invoked by
  L-Chain $j$'s SAS server group
    if $\mathcal{S} \in$ LC$[j].SG$ and
      $epoch \in [LC[j].epoch, system.block]$ then
        if the majority of LC$[j].SG$ provide the same
          LC$[j].\{epoch, sroot\}$ then
            LC$[j].\{epoch, sroot\} \leftarrow \{e, sroot\}$
```

blocks). This is crucial to our system's resilience against adaptive corruptions on SAS servers at the local level, and to avoid the case that SAS servers of one L-Chain belong to the same SAS administrator. This procedure takes advantage of the cryptographic primitive verifiable random function (VRF).

VRF was first introduced by Micali *et al.* [29] in 1999 for providing both unpredictability and verifiability of pseudorandom functions. It allows a function caller i to generate a random $hash$ and a proof π with its private key sk_i. The proof π allows others to verify the validity of the hash using the caller's public key pk_i. The randomness of $hash$ means it looks uniformly distributed to others without knowing π. Specifically, VRF generation and verification are as follows:

- $\langle hash, \pi\rangle \leftarrow \mathbf{VRF}_{sk_i}(role||seed)$
- $decision \leftarrow \mathbf{VerifyVRF}_{pk_i}(role||seed, hash, \pi)$

wherein $role$ is a descriptor, such as '*sas_server_#i*', and $seed$ is a public random seed known to the system. $decision$ is a binary value taking TRUE or FALSE.

VRF-based SAS Server Reshuffling. Back to our system, SAS servers participate in the reshuffle procedure individually. If SAS server i is interested in joining $\mathcal{SG}_j$ (i.e., serving L-Chain j) for the next shift, at the start of the R_j^{th} last epoch of the current shift, SAS server i needs to do the following:

- Generate a $hash_i$ and a proof π_i by executing
$$\langle hash_i, \pi_i\rangle \leftarrow \mathbf{VRF}_{sk_i}(\text{‘}sas\_server\_i\text{’}||sroot) \tag{1}$$
where $sroot$ is the L-Chain j's state root of the last shift;
- Submit $hash_i$ and π_i to G-Chain by calling C_R's *ServerReshuffle* function which updates the Lottery$[j][i]$ variable.

When the last epoch starts, each of L-Chain j's anchor witnesses collects the available hash-proof pairs from $Lottery[j][\cdot]$ from C_R and performs the following steps:

- Sort the hash-proof pairs by hash value in ascending order;
- Along this order, for every hash-proof pair $\langle\pi_i, hash_i\rangle$ (denote SAS server i the generator), verify it by performing
$$decision \leftarrow \mathbf{VerifyVRF}_{pk_i}(\text{‘}sas\_server'||sroot, hash_i, \pi_i) \tag{2}$$
(*Server i does not share the same administrator with anyone in $selected$[]*)
- If $decision$ = TRUE, add i to $selected$[]; then continue the last step for the next hash-proof pair in the order;
- Terminate when $selected$[] has S_j entries, and submit $selected$[] to C_R by calling *ReshuffleConfirm*.

Lastly, *ReshuffleConfirm* updates $\mathcal{SG}_j$ in C_R only when the majority witnesses of L-Chain j also sign for the same $selected$[]. A newly updated SAS server i for L-Chain j will need to connect to the anchor witnesses through TLS communication before the start of the new shift, under the PKI established by the regulator (as is in CBRS [20]).

D. L-Chain Operation

L-Chain is a special-purpose permissioned blockchain where the SAS servers and witnesses have different responsibilities in the ledger curating process, with the former providing transaction execution service (thus enforcing state changes from the last block) while the latter providing transaction ordering service (thus finalizing new blocks). Both SAS servers and witnesses maintain the L-Chain ledger. At the bootstrapping phase, the genesis block is established among the initial anchor witnesses and SAS servers, which are regulator-certified entities and published in C_R. The genesis block serves the L-Chain's first beacon block with a hash pointer to the most recent G-Chain block. The genesis block also comes with an initial C_{SA} setup, specifying the region information and initial anchor witnesses. For the CBRS band and others that have priority licensees, the initial anchor witnesses can be conveniently undertaken by the priority licensees. Since L-Chain is permission-controlled, a new witness must first operate a valid spectrum user (or the proxy for a group of users) and pass the user registration process.

Separation of Execution and Consensus. We adopt a separation paradigm for L-Chain transaction execution and

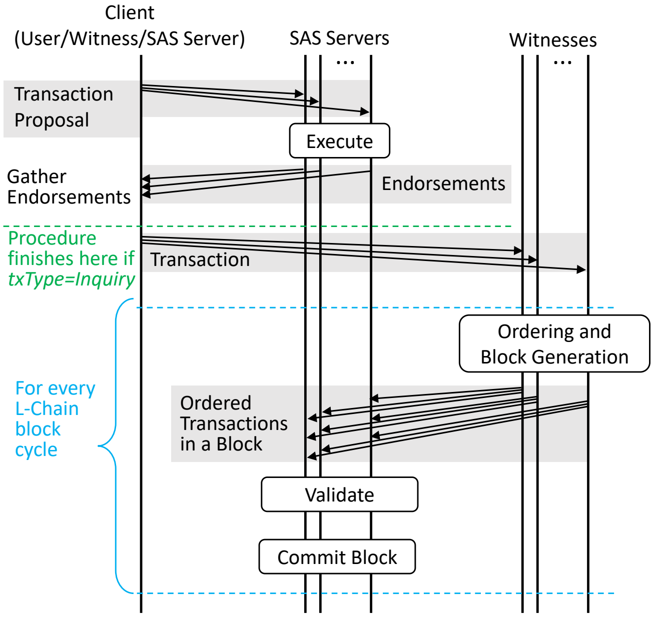

Client
(User/Witness/SAS Server)
SAS Servers
Witnesses
Transaction Proposal
Execute
Gather Endorsements
Endorsements
Procedure finishes here if txType=Inquiry
Transaction
Ordering and Block Generation
For every L-Chain block cycle
Ordered Transactions in a Block
Validate
Commit Block

Fig. 3. L-Chain workflow built on Hyperledger Fabric's execute-order-validate model [19].

consensus making, where SAS servers and witnesses undertake transaction execution and transaction serialization respectively. The separation concept was originally proposed in [30] to reduce server redundancy and increase modularity in state machine replication systems and later adapted to permissioned blockchain frameworks such as Tendermint and Hyperledger Fabric [19] for efficiency and modularity purposes. It is in contrast to traditional blockchain consensus that relies on an atomic consensus module, such as Ethereum, where transaction ordering and execution are performed in one operation. While both paradigms can realize a blockchain state machine and smart contracts and yield similar throughput performance, the separation paradigm supports the client-in-the-loop endorsement mechanism which provides better architectural modularity, flexibility, and low latency in obtaining service. To realize this design, we use Hyperledger Fabric [19] to implement L-Chain and one L-Chain transaction workflow is shown in Fig. 3. A client, e.g., a spectrum user, sends out a transaction proposal containing input to an L-Chain contract. The SAS servers execute the contracts and return the outputs and state changes as *endorsements*. The client needs to collect more than half of the endorsements before sending out the real transaction for block generation. After the witnesses generate a block containing serialized transactions, they broadcast the block to SAS servers who will validate the block proof and all other endorsements before committing the block. The witnesses are only responsible for the block generation (i.e., transaction ordering) and keep records of the blocks sent to SAS servers.

From the deployment perspective, this separation paradigm is key to the system's responsiveness in processing spectrum access requests (**R-2**) as it capitalizes on the physical differences between SAS servers and local witnesses. First, SAS servers have presumably sufficient computing capability for executing transactions. Meanwhile, local witnesses are not necessarily powerful computers, as they represent diverse types of spectrum users. Second, local witnesses tend to physically reside in the region where their associated users access the spectrum. Their close proximity yields low communication delays, which enables the usage of deterministic consensus schemes such as Raft [31] or efficient BFT protocols [32], [33] for the ordering task. In comparison, we do not assume any locality of SAS servers and they may reside in a cloud cluster across the country.

Encoding C_{SA}. The spectrum access contract C_{SA} profiles local spectrum participants (i.e., witnesses and users) and enables SAS servers to provide consistent spectrum management service. C_{SA} pseudocode is shown in Algorithm 2. At the start of a shift, everyone in the newly selected SAS server group SG calls the *ShiftUpdate* function indicating its incoming service, which needs approvals from the majority of witnesses. At each epoch onset, SG fetches the latest regulatory information from C_R, including updates of interference model parameters and anchor change confirmation in case an anchor witness was changed by the regulator. Adding new or removing existing witnesses requires the majority of existing witnesses to invoke the *WitnessUpdate* function. Next, we provide details C_{SA} to fulfill two essential spectrum management tasks.

User Management (T-1). For a new spectrum user to register with the L-Chain, it needs to pass the off-line device registration procedure (e.g., the CBSD registration protocol [5]) with each connected SAS server. After which the servers add this user by calling *UserUpdate* with a majority vote. User removal also is accomplished through *UserUpdate*, via either the user's own action or the majority of SG.

Access Assignment (T-2). A spectrum user submits its spectrum access request by invoking the *Request* function along with operational parameters, including the requested zone, channel range, and effective isotropic radiated power (EIRP). This function encodes a channel assignment algorithm that calculates a decision based on the existing user's device parameter, operational parameters, as well as current channel availability, through a predefined interference model (i.e., iModel in Algorithm 2). For this iModel and later implementation, we adopt a simple first-come-first-serve allocation criterion, where the assignment is approved if the requested channel range at the request zone is available and the user's EIRP would not cause harmful interference to users of neighboring zones. More complex channel allocation schemes exist in the literature, as we discuss in §VIII. Besides the *Access* request type, there also exist *Inquiry* and *Heartbeat* request types, which enable spectrum users to perform simple inquiries on readily available channels and provide periodic proofs of liveness (per the existing SAS in CBRS) respectively. Our implementation in §VII also included all three types of requests to demonstrate this backward compatibility.

Performance Features. In terms of L-Chain performance, as long as the user's requests are less frequent than L-Chain's block frequency, the response can be finalized by $\Omega(1)$ scale of the L-Chain block interval. In the implementation, we set L-Chain's block interval to one second, while in the current SAS implementation, the tightest response timing requirement

Algorithm 2: Spectrum Access Contract C_{SA} Pseudocode (essential functions only)

```
var Zones[].{$id, desc, licensees$} //Zone coverage of this local region
var ZCH[][].{$zone, status$} //Readily available zone-channel pairs
var iModel.{$schema, para1, para2, ...$} //Interference model
var Witnesses[].{$type$} //type is ANCHOR or NORMAL
var SG[] //SAS server group for this L-Chain
var Users[].{$tier, opParam$} //Users profile
var A[].{$desc, zone, chs, ERP, status$} //Access assignment, indexed by user id
Function IntfModel(z, c, EIRP) //Internal function
    Checks if EIRP is high enough to cause harmful interference to neighboring zones of $z$ on channel $c$, using parameters specified in iModel. If yes, return FAIL; otherwise textbfreturn SUCCESS;
Function Init() //Contract creation (omitted for brevity)
Function EpochUpdate() //Allows servers to update variables by voting (omitted for brevity)
Function ShiftUpdate() //Witnesses accept new SAS servers by voting (omitted for brevity)
Function WitnessUpdate(tp, users, action, p)//Update by voting (omitted for brevity)
Function UserUpdate(u, desc, opPara, action) //$\mathcal{S}$: function caller
    if $\mathcal{S} \in$ SG and $action$ = ADD then
        Add User[$\mathcal{S}$].{$desc, par$} if the function is called by the majority SG
    if $action$ = REMOVE then
        Remove Users[$u$] if $u = \mathcal{S}$ or the function is called by the majority SG
Function Request(desc, type, zone, chs, EIRP) //type: Inquiry/Access/Heartbeat
    if $\mathcal{S} \in$ Users and $type = Inquiry$ then
        return ZCH[$z$][$c$].$status$ for all $c \in chs$
    if $\mathcal{S} \in$ Users and $type = Access$ then
        if $\exists c \in chs$ that ZCH[$zone$][$c$] = OCCUPIED or IntfModel(zone,c,EIRP)=FAIL then
            return FAIL
        else
            A[$\mathcal{S}$].{$desc, zone, chs, EIRP, status$} ← {$desc, zone, chs, EIRP$, ASSIGNED}
            ZCH[$zone$][$chs$] ← OCCUPIED
    if $\mathcal{S} \in$ Users and $type = Heartbeat$ then
        A[$\mathcal{S}$].{$desc, status$} ← {$desc$, AUTHORIZED}
```

(a.k.a. response to Heatbeat) should be done within 60 seconds [5], [22]. we require all transactions be finalized in 5 blocks or otherwise fail. On the other hand, if a spectrum user has a high level of trust in the availability of its local witnesses, they can start using the allocating channels right after receiving the majority of SAS server endorsements, just like receiving an *Inquiry* response.

V. Analyses

Next, we analyze the security and performance properties of BD-SAS under the aforementioned threat model (§III-D). We assume there are N SAS servers globally, F of which are potentially Byzantine. Every L-Chain's SAS server group size is fixed to S and witness population W.

Proposition 1 (Fault tolerance)**.** *As long as $N > 2F$ (i.e., the honest majority), G-Chain operations are safe against Byzantine SAS servers. The same fault tolerance also applies to L-Chain except for exponentially diminishing probability as S increases.*

Proof: The G-Chain's tolerance against Byzantine SAS servers follows from the underlying blockchain consensus which is secure under the honest majority assumption. L-Chain's tolerance derives from the verifiable randomness of the SAS server reshuffle procedure (§IV-C) executed for every shift. The chance that the majority of the S SAS servers are Byzantine is $(\frac{F}{N})^{\lceil S/2 \rceil}$, which diminishes exponentially as S increases since $N > 2F$. □

Remark 1. *The liveness aspect of L-Chain security, i.e., all valid L-Chain transactions should be eventually finalized in the ledger, depends on the availability of the local witnesses and the consensus protocol they run for the ordering task. We consider it a standalone challenge as the witnesses can internally manage their own trust levels and choice of consensus protocol as long as there is an honest majority. In the implementation, we stick to Fabric's Raft consensus for prototyping.*

Proposition 2 (Adaptive corruption resilience)**.** *Both G-Chain and L-Chain operations are resilient to an adaptive adversary who can corrupt any SAS server in the time scale of shifts.*

Proof: The G-Chain's resilience against adaptive corruptions on SAS servers follows as long as the honest majority assumption holds at all times. Meanwhile, L-Chain's adaptive corruption resilience is ensured by the timing regime of the SAS server reshuffle procedure (§IV-C). That is, SAS servers call the *ReshufflePropose* function only at the start of the R-th last G-Chain blocks of before the current shift ends. The adversary has no additional information (i.e., no different than random guessing) on whether a SAS server will be chosen by an L-Chain. Similarly, the adversary's successful corruption on certain SAS servers for the current shift does not last into the next shift. □

Proposition 3 (Responsiveness)**.** *Given that SAS servers have stable connections to spectrum users and each spectrum user submits spectrum requests (i.e., inquiry, access, or heartbeat) less frequently than the L-Chain block's maximum transaction throughput, the spectrum user can get an access assignment (or an inquiry/heartbeat response) within the time scale of $\Omega(1)$ seconds (i.e., irrespective of SAS server group size S).*

Proof: This follows from L-Chain's separation paradigm as is shown in Fig. 3. A user is ready to propose a formal transaction when its transaction proposal collected endorsements from the majority of the S SAS servers, within $O(S)$ TLS communication sessions. In our Hyperledger Fabric-based L-Chain, each of such communication sessions follows the server-client model and they can be performed in parallel by a user. Assuming a stable connection between SAS servers and the user, the total delay for a user to gather the majority of endorsements is thus capped by the maximum endorsement delay from SAS servers, not the sum of all delays. This means that the time needed for the user to generate a formal L-Chain transaction is irrespective of S. Meanwhile, an assignment is not finalized until passing the ordering and block generation phase which is handled by the witnesses. The finalization can be done by the next L-Chain block cycle as long as users' requests do not overwhelm the L-Chain block's maximum transaction throughput, which is determined by the witness group's internal communication and processing power. □

VI. Implementation

We used the Ethereum Rinkeby testnet [34] to emulate the G-Chain (i.e., a public blockchain) and the Hyperledger Fabric platform to implement the L-Chain prototypes.[2] The reason for choosing Rinkeby to emulate G-Chain is two-fold. First, it offers a complete smart contract tool set (identically to that in the Ethereum main blockchain). Second, it has a static $B = 15s$ block interval which can act as a stable timing source for BD-SAS's epoch and shift changes. We implemented the G-Chain regulatory contract C_R in Solidity with 202 lines of code.[3]

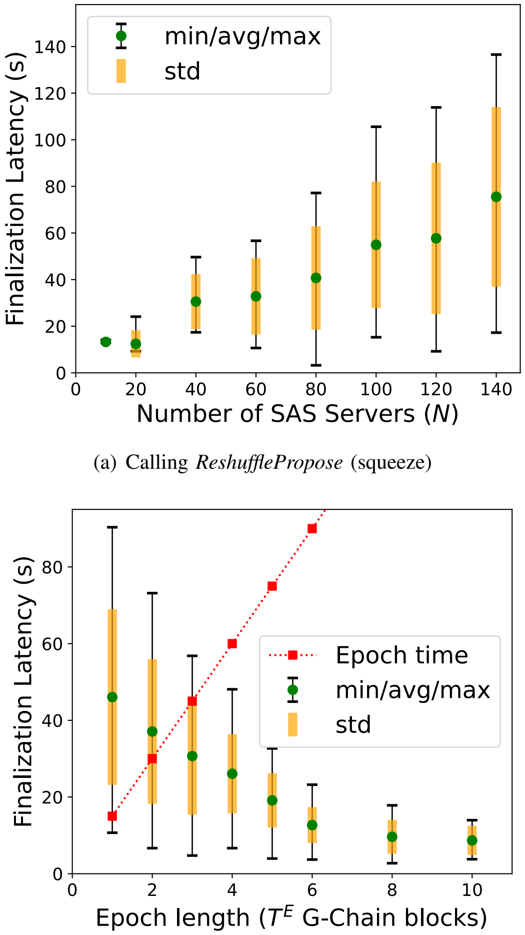
Our L-Chain implementations consisted of $\{3, 5, 7, 9\}$ SAS servers, each with fixed 5 witnesses, all of which were instantiated in docker containers in a Linux testbed on top of an AWS T2.xlarge machine. The L-Chain block interval was fixed to 1 second and every block could enclose up 1MB of transactions. Fabric's native Raft consensus (a crash-fault-tolerant protocol) was used for the witnesses' ordering task. The epoch time T^E took value from $\{1, 2, ..., 10\}$ (in B), and the shift time T^S was fixed to 1000 (in B), where the L-Chain index is left out for convenience. We implemented the spectrum access contract C_{SA} in Golang with 234 lines of code. C_{SA} is designed to enable three types of spectrum access operations: *Access*—which returns a channel assignment, *Inquiry*—which returns the information of a readily available channel-location, and *Heartbeat*—which is called periodically by a spectrum user to demonstrate liveness and returns an authorization for continuing channel usage. It is noted that our C_{SA} implementation adopts a straightforward first-come-first-serve (FCFS) channel allocation model and does not consider multi-user interference coordination when calculating an assignment, which would need an optimization-based allocation technique. We will explore this option in future work.

Lastly, for the SAS server reshuffle procedure, we used the VRF implementation of [35] which is based on the elliptic curve signature system ed25519 with 32-byte private/public keys, 64-byte hashes, and 80-byte proofs.

VII. Evaluation

A. G-Chain Performance and Feasibility

We evaluated the feasibility of Rinkeby as G-Chain by testing its response latency for fulfilling two key G-Chain operations: reshuffle and local update. The first operation involves N SAS servers calling the *ReshufflePropose* function of C_R simultaneously, simulating a squeeze situation that stress-tests the G-Chain, which in practice will also provide the highest security against adaptive attacks since the VFR results are exposed with minimal time. For each experiment (corresponding to one bar in Figure 4(a)), we measured the minimum, average, and maximum transaction finalization latency, as well as the standard deviation for all the N simultaneous *ReshufflePropose* calls. It is observed from Figure 4(a) that the average finalization latency of *ReshufflePropose* calls generally increases linearly with N. This indicates that we would need to increase the epoch length given the larger N, for having the reshuffle procedure finished in one epoch. For example, if $N = 140$, we would need to set a minimum epoch time of 10 G-Chain block cycles (i.e., 150s). The second operation involves $N = 100$ SAS servers, representing 20 L-Chains, calling the *LocalUpdate* function C_R periodically but at different instants, simulating an uncongested and orderly situation. For each experiment (corresponding to one bar in Figure 4(b)), we



(a) Calling *ReshufflePropose* (squeeze)

(b) Calling *LocalUpdate* (regular)

Fig. 4. Stress test transaction finalization latency of Ethereum Rinkeby testnet (as G-Chain). (a) A once-a-shift squeeze scenario where all N SAS servers call C_R's *ReshufflePropose* simultaneously. (b) The regular operation scenario where N (=100) SAS servers (representing 5 L-Chains) call C_R's *LocalUpdate* intermittently across an epoch.

[2] The source code is available at https://github.com/yang-sec/bdsas

[3] The deployed contract is viewable at https://rinkeby.etherscan.io/address/0xDBd97d9d6e61dB19e3Dd0eAfcaF132507BEC1098.

TABLE III
ROUND TRIP TIMES (IN MILLISECONDS) AMONG REAL CLOUD NODES AND THE NODE RESEMBLANCE TO L-CHAIN ENTITIES.

Node	Location	L-Chain Resemblance	1	2	3	4	5	6	7
1	AWS (N.Virginia)	Local Witnesses	-	-	-	-	-	-	-
2	Lab (N.Virginia)	Local Users	2.4	-	-	-	-	-	-
3	AWS (Ohio)	SAS Server 1	11.8	12.0	-	-	-	-	-
4	AWS (California)	SAS Server 2	62.0	82.3	50.1	-	-	-	-
5	AWS (Oregon)	SAS Server 3	66.9	71.3	49.5	21.8	-	-	-
6	Google (Nevada)	SAS Server 4	62.2	73.5	62.2	18.6	40.3	-	-
7	Google (Utah)	SAS Server 5	52.8	66.6	52.9	20.8	41.1	20.0	-

measured the minimum, average, and maximum transaction finalization latency as well as the standard deviation for all 100 *LocalUpdate* calls. It is observed from Figure 4(b) that longer epochs generally lead to finalization latency (due to less congestion) and only when $T^E \geq 4$ (i.e., 60 seconds) can the G-Chain ensure every update is confirmed within one epoch.

For the SAS server reshuffle procedure, we benchmarked the VRF operations on a Linux machine with 16GB memory and a 3.0GHz CPU. It took 537ms to generate the VRF hash and proof, and 441ms to verify them. We can see that these off-chain delays added to the reshuffle procedure are negligible compared to the time cost of their on-chain part.

We remark that in practical implementation, we can opt for establishing a brand new G-Chain for BD-SAS for exclusive use (i.e., instead of using a public blockchain platform). The permissioned G-Chain would provide more stable latency performance and low congestion since it does not need to accommodate non-BD-SAS traffic. We will consider such implementation in the future instantiation of BD-SAS.

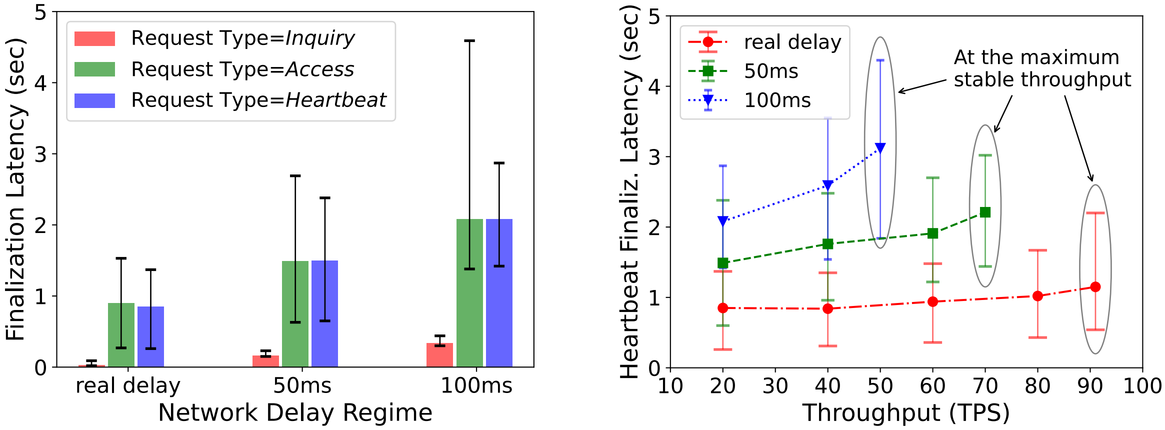
B. L-Chain Performance

Next, we evaluate the performance of our L-Chain prototypes with respect to two metrics: the finalization latency of a spectrum request and the throughput capacity for the L-Chain to handle a certain volume of such requests. In particular, we evaluated all three types of requests: *Inquiry*, *Access*, and *Heartbeat*. The tests on processing *Inquiry* and *Access* requests were configured with a fixed invocation frequency of 20 transactions per second (TPS). The tests on *Heartbeat* requests were configured with varying invocation frequencies from 20 TPS to the maximum stable throughput (i.e., all transactions result in success). This emulates the practical case that the *Heartbeat* requests would be much more frequent, potentially than the other two. We used Hyperledger Caliper [36], a blockchain benchmarking tool, to simulate the abovementioned transaction traffic which invokes the *Request* function with the corresponding request type. Moreover, to simulate network delays in the Internet, we used three delay regimes for L-Chain participants:

- *real*: the end-to-end packet delay statistics measured among commercial cloud servers across the US. This delay regime was only used on the L-Chain with 5 SAS servers. The measurement and L-Chain node resemblance are shown in Table III.
- $50ms$ or $100ms$: the synthetic end-to-end packet delay added uniformly to every pair of L-Chain participants. These two delay regimes were used on the L-Chain with $S \in \{3, 5, 7, 9\}$ SAS servers.

We first evaluated the performance of one L-Chain with 5 SAS servers under all three delay regimes, as is shown in Figure 5. Figure 5(a) shows the finalization latency of three types of requests with the request frequency fixed at 20 TPS. For each experiment (one bar in Figure 5(a)), we measured the average, minimum, and maximum finalization latency for all transactions processed by the L-Chain during a 100-second run (i.e., 2000 transactions in total). We observe that even under the harshest $100ms$ delay regime, the system is able to finalize an *Inquiry* request, an *Access* request, or a *Heartbeat* request within 0.5s, 5s, or 3s respectively. The higher latency variation in finalizing the *Access* request was likely caused by the varying size of channel-location availability slots in our C_{SA} implementation, while an *Inquiry* or *Access* request invokes constant look-ups. Figure 5(b) shows the finalization latency of *Heartbeat* requests only but with varying throughput pressure. For each experiment (one bar in Figure 5(b)), we measured the same latency statistics during a 100-second run for all successful transactions processed by the L-Chain (i.e., 100×throughput transactions in total). The throughput values tested were capped by a "maximum stable" point, indicating that further increase would lead to permanently unfinalized transactions. We observe that higher delays reduce the maximum state throughput, which is intuitive in that higher delays decrease the system's overall service efficiency.

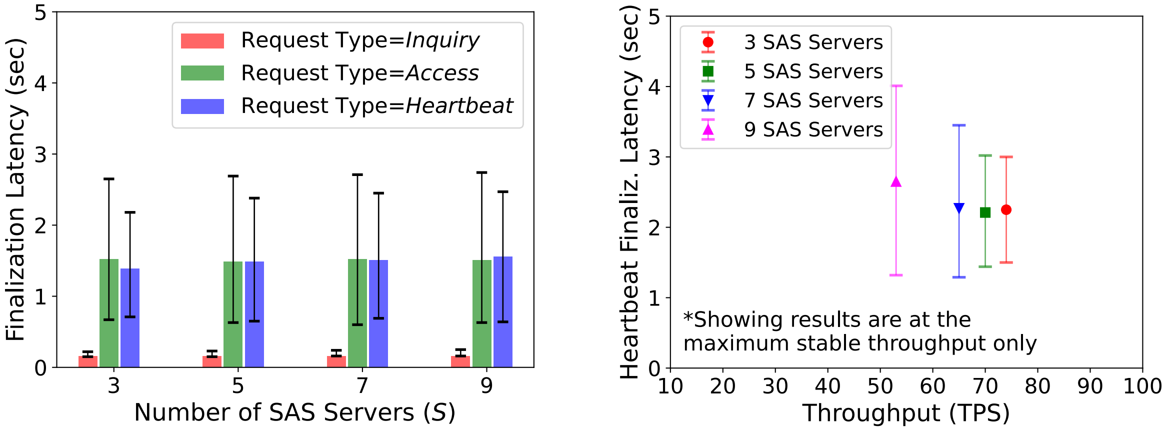
In both Figure 5(a) and Figure 5(b), it is observed that under the real delay scenario and within the stable throughput region, the three types of requests can be finalized within 2 seconds. This latency performance demonstrates the L-Chain's potential of improving the existing CBRS-SAS's grant approval process (i.e., responding to *Access* requests) from a next-day wait to several seconds wait. The core reason is that the L-Chain allows the SAS servers to synchronize states for every L-Chain block cycle and thus update the common L-Chain ledger in a deterministic and incremental manner, enabling the L-Chain contract to generate online allocations. In comparison, the current SAS is not designed to provide online allocations nor tolerate Byzantine participants. The CPAS process dictates that SAS servers should faithfully exchange full-dump records [6] before finalizing allocations the next morning (US Eastern Time). Moreover, Figure 5(a) also demonstrates the L-Chain's capability of fulfilling the CBSD liveness requirement (i.e., sending routine responses to *Heartbeat* requests) at a significant performance margin, for which the Heartbeat interval is typically set to 30 or 60



(a) At constant 20 TPS throughput.

(b) *Heartbeat* stress test.

Fig. 5. Average request finalization latency (error bar represents the minimum and maximum value) of the L-Chain with 5 SAS servers under three different delay regimes.



(a) At constant 20 TPS throughput.

(b) *Heartbeat* stress test.

Fig. 6. Average request finalization latency (error bar represents the minimum and maximum value) of L-Chains with $S \in \{3, 5, 7, 9\}$ SAS servers under the $50ms$ delay regime.

seconds [22].

We further investigated the performance implication of SAS population S. The results for L-Chains with four different S under the same $50ms$ delay regime are shown in Figure 6. Figure 6(a) shows that under the stable throughput scenario, S has a negligible impact on the finalization latency. Meanwhile, Figure 6(b) shows that larger S does become a constraining factor on the maximum stable throughput and also on the finalization latency with the request invocation rate approaches the maximum stable throughput. Considering that a larger S provides a higher level of adaptive attack resilience (see Proposition 2), we will need to choose a practical S in the design phase to strike a balance between security and throughput capacity in practical deployment.

We caution that our experiment used a simple first-come-first-server allocation scheme; an optimal allocation algorithm would most likely incur heavy and non-linear computation at SAS servers. Readers are deferred to §VIII for discussions on accommodating complex allocation algorithms. However, the general observation is that using an active and ledger-based inter-SAS coordination mechanism (such as a blockchain) represents an effective way to automate spectrum management tasks and shorten the spectrum user's waiting time, compared to relying on the CPAS procedure for daily allocation assignment.

VIII. Discussion

Optimality of Spectrum Allocation. The current spectrum assignment function, as is encoded in our spectrum access contract (Algorithm 2) and implemented in our prototype, is a straightforward first-come-first-serve allocation scheme. In practice, spectrum allocation can deploy a much more complex, hopefully, an optimization-based algorithm to guarantee succinct spectrum assignment with specific service requirements such as fairness, bounded response time, or channel utilization rate, as was indicated in prior arts [37]–[40]. However, instantiating such an optimization-based allocation algorithm on blockchain smart contracts would incur a very high on-chain cost, as every invocation of the algorithm would be executed and stored by every L-Chain curator. To cope with this scalability constraint, we are eyeing two potential solutions: (1) customizing optimization-based allocation algorithms into minimal yet effective versions so they are suited for blockchain deployment; (2) using dedicated secure environments, such as a hardware-based trusted execution environment (TEE) [41], [42], to execute the algorithms off-chain and return the result on-chain using a secure commit protocol [43], [44].

Privacy Challenge. The current BD-SAS's spectrum access assignment function is executed in a transparent, on-chain manner in that every L-Chain curator knows each user's assignment record. This can be a disincentive for privacy-aware spectrum users, who may not wish to share their access or device information with other witnesses, whose associated users also reside in the same local region. To provide privacy protection to spectrum users, aside from the off-chain TEE solutions mentioned above, we could employ obfuscation mechanisms, such as those with differential privacy [45], [46], to allow users to control their privacy leakage. Moreover, using efficient on-chain multiparty computation is also a potential direction to explore.

IX. Conclusion

We formulated a decentralized SAS model for fault-tolerant and automated dynamic spectrum sharing and introduced the BD-SAS architecture as a concrete solution. BD-SAS consists of two layers of blockchains: a G-Chain for global regulation and SAS server management and L-Chains for providing spectrum access assignment to spectrum users in the local regions, without having the users place trust in individual SAS servers. We implemented a BD-SAS prototype using Ethereum Rinkey testnet to emulate the G-Chain and Hyperledger Fabric platform to implement the L-Chains. The result demonstrated the feasibility of G-Chain in performing the critical security mechanisms including SAS server reshuffling as well as the responsiveness of L-Chain in generating spectrum access assignments amid stringent timing requirements. Lastly, we identified challenges for BD-SAS and future directions.

Acknowledgment

This work was supported in part by the U.S. National Science Foundation under grants 1916902, 1916926, 2154929, and 2154930, the Army Research Office under grant W911NF-20-1-0141, and a gift from InterDigital.

References

[1] Y. Xiao, *Blockchain and Distributed Consensus: From Security Analysis to Novel Applications*. PhD thesis, Virginia Tech, 2022.

[2] Federal Communications Commission (FCC), "What we do." https://www.fcc.gov/about-fcc/what-we-do, accessed December 14, 2022.

[3] National Telecommunications and Information Administration (NTIA), "About NTIA." https://ntia.gov/page/about-ntia, accessed December 13, 2022.

[4] The Office of the Federal Register (OFR) and the Government Publishing Office, "OFR: Electronic Code of Federal Regulations, Title 47: Telecommunication, Part 96 - Citizens Broadband Radio Service." https://www.ecfr.gov/cgi-bin/text-idx?node=pt47.5.96, 2015.

[5] Wireless Innovation Forum, *Signaling Protocols and Procedures for Citizens Broadband Radio Service (CBRS): Spectrum Access System (SAS) - Citizens Broadband Radio Service Device (CBSD) Interface Technical Specification*, 12 2016. Version V1.0.1.

[6] Wireless Innovation Forum, *Spectrum Sharing Committee Policy and Procedure Coordinated Periodic Activities Policy*, 6 2018. Version V1.3.0.

[7] Wireless Innovation Forum, *Signaling Protocols and Procedures for Citizens Broadband Radio Service (CBRS): Spectrum Access System (SAS) - SAS Interface Technical Specification*, 3 2020. Version V1.3.2.

[8] M. Kratsios, "Emerging technologies and their expected impact on non-federal spectrum demand," *Executive Office of the President of the United States*, 2019.

[9] S. Yrjölä, "Analysis of blockchain use cases in the citizens broadband radio service spectrum sharing concept," in *International Conference on Cognitive Radio Oriented Wireless Networks*, pp. 128–139, Springer, 2017.

[10] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[11] M. B. Weiss, K. Werbach, D. C. Sicker, and C. E. C. Bastidas, "On the application of blockchains to spectrum management," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 2, pp. 193–205, 2019.

[12] Federal Communications Commission (FCC), "Remarks of commissioner jessica rosenworcel [at] mobile world congress americas, los angeles, california, september 13, 2018." https://docs.fcc.gov/public/attachments/DOC-354091A1.pdf, 2018.

[13] L. Mearian, "FCC eyes blockchain to track, monitor growing wireless spectrums." https://www.computerworld.com/article/3393179/fcc-eyes-blockchain-to-track-monitor-growing-wireless-spectrums.html, accessed April 20, 2023.

[14] T. Ariyarathna, P. Harankahadeniya, S. Isthikar, N. Pathirana, H. D. Bandara, and A. Madanayake, "Dynamic spectrum access via smart contracts on blockchain," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2019.

[15] M. Grissa, A. A. Yavuz, and B. Hamdaoui, "Trustsas: a trustworthy spectrum access system for the 3.5 ghz cbrs band," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1495–1503, IEEE, 2019.

[16] H. Zhang, S. Leng, and H. Chai, "A blockchain enhanced dynamic spectrum sharing model based on proof-of-strategy," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2020.

[17] S. Shi, Y. Xiao, W. Lou, C. Wang, X. Li, Y. T. Hou, and J. H. Reed, "Challenges and new directions in securing spectrum access systems," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6498–6518, 2021.

[18] Y. Xiao, S. Shi, W. Lou, C. Wang, X. Li, N. Zhang, Y. T. Hou, and J. H. Reed, "Decentralized spectrum access system: Vision, challenges, and a blockchain solution," *IEEE Wireless Communications*, pp. 1–9, 2022.

[19] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, p. 30, ACM, 2018.

[20] Wireless Innovation Forum, *CBRS Communications Security Technical Specification*, 6 2020. Version V1.2.0.

[21] Wireless Innovation Forum, *CBRS Operational Security*, 7 2017. Version V1.0.0.

[22] Google, "Changes in Google SAS response." https://support.google.com/sas/answer/9981557, accessed April 20, 2023.

[23] Google, "SAS sync (or IAP) & what it means for you." https://support.google.com/sas/answer/9554929, accessed December 21, 2022.

[24] Z. Li, W. Wang, J. Guo, Y. Zhu, L. Han, and Q. Wu, "Blockchain-assisted dynamic spectrum sharing in the CBRS band," in *2021 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 864–869, IEEE, 2021.

[25] Z. Zhang, Y. Xiao, Z. Ma, M. Xiao, Z. Ding, X. Lei, G. K. Karagiannidis, and P. Fan, "6g wireless networks: Vision, requirements, architecture, and key technologies," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 28–41, 2019.

[26] M. B. Weiss and W. Lehr, "Market based approaches for dynamic spectrum assignment," *Available at SSRN 2027059*, 2009.

[27] H. Attiya and J. Welch, *Distributed computing: fundamentals, simulations, and advanced topics*, vol. 19. John Wiley & Sons, 2004.

[28] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[29] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*, pp. 120–130, IEEE, 1999.

[30] J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin, "Separating agreement from execution for byzantine fault tolerant services," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pp. 253–267, 2003.

[31] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX Annual Technical Conference (Usenix ATC 14)*, pp. 305–319, 2014.

[32] J. Sousa, A. Bessani, and M. Vukolic, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pp. 51–58, IEEE, 2018.

[33] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pp. 347–356, 2019.

[34] Etherscan.io, "Rinkeby testnet documentation." https://www.rinkeby.io/, accessed April 20, 2023.

[35] NCC Group, "Reference implementation of a verifiable random function (vrf) from IETF draft-irtf-cfrg-vrf-06 specification." https://github.com/nccgroup/draft-irtf-cfrg-vrf-06, accessed April 20, 2023.

[36] Hyperledger Project, "Hyperledger caliper." https://hyperledger.github.io/caliper/, accessed April 20, 2023.

[37] K. S. Manosha, S. Joshi, T. Hänninen, M. Jokinen, P. Pirinen, H. Posti, K. Horneman, S. Yrjölä, and M. Latva-aho, "A channel allocation algorithm for citizens broadband radio service/spectrum access system," in *2017 European Conference on Networks and Communications (EuCNC)*, pp. 1–6, IEEE, 2017.

[38] X. Ying, M. M. Buddhikot, and S. Roy, "Sas-assisted coexistence-aware dynamic channel assignment in cbrs band," *IEEE Transactions on Wireless Communications*, vol. 17, no. 9, pp. 6307–6320, 2018.

[39] S. Basnet, Y. He, E. Dutkiewicz, and B. A. Jayawickrama, "Resource allocation in moving and fixed general authorized access users in spectrum access system," *IEEE Access*, vol. 7, pp. 107863–107873, 2019.

[40] N. Jai, S. Li, C. Li, Y. T. Hou, W. Lou, J. H. Reed, and S. Kompella, "Optimal channel allocation in the cbrs band with shipborne radar incumbents," in *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pp. 80–88, IEEE, 2021.

[41] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: what it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, pp. 57–64, IEEE, 2015.

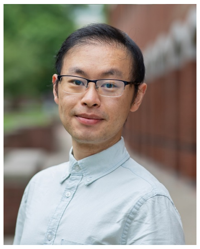
[42] V. Costan and S. Devadas, "Intel sgx explained." Cryptology ePrint Archive, Paper 2016/086, 2016. https://eprint.iacr.org/2016/086.

[43] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song, "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 185–200, IEEE, 2019.

[44] Y. Xiao, N. Zhang, J. Li, W. Lou, and Y. T. Hou, "Privacyguard: Enforcing private data usage control with blockchain and attested off-chain contract execution," in *European Symposium on Research in Computer Security*, pp. 610–629, Springer, 2020.

[45] C. Dwork, "Differential privacy: A survey of results," in *International conference on theory and applications of models of computation*, pp. 1–19, Springer, 2008.

[46] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 1054–1067, 2014.



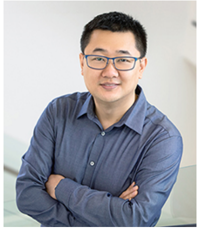
Yang Xiao [M] received the Ph.D. degree in Computer Engineering from Virginia Tech in 2022. He is currently an Assistant Professor with the Department of Computer Science at the University of Kentucky, Lexington, KY, USA. His research interests lie in network security, distributed system security, blockchain and decentralized systems, and mobile network security.

Shanghao Shi [S] is currently pursuing the Ph.D. degree with the Department of Computer Science at Virginia Tech, supervised by Prof. Wenjing Lou. He received his B.S. degree from the School of Information and Communication Engineering at Beijing University of Posts and Telecommunications. His research interests lie in wireless security and IoT security.

Wenjing Lou [F'15] is the W. C. English Endowed Professor of Computer Science at Virginia Tech and a Fellow of the IEEE. Her research interests cover many topics in the cybersecurity field, with her current research interest focusing on wireless network security, trustworthy AI, blockchain, and security and privacy problems in the Internet of Things (IoT) systems. Prof. Lou is a highly cited researcher by the Web of Science Group. She received the Virginia Tech Alumni Award for Research Excellence in 2018. She received the INFOCOM Test-of-Time paper award in 2020. She was the TPC chair for IEEE INFOCOM 2019 and ACM WiSec 2020. She was the Steering Committee Chair for IEEE CNS conference from 2013 to 2020. She is currently a steering committee member of IEEE INFOCOM and IEEE Transactions on Mobile Computing. She served as a program director at the US National Science Foundation (NSF) from 2014 to 2017.

Chonggang Wang [F'17] is a Principal Engineer at InterDigital Communications Inc. He has 20+ years of experience in the field of communications, networking, and computing including research, development and standardization of wireless systems, Internet of Things (IoT), quantum internet, internet protocols. He currently leads a technical team in the Customer Project & Partners department of InterDigital's Research and Innovation Wireless Lab. In this role he and his team focus on research, innovation, and standardization of blockchain technology and its applications for future communications and computing systems (e.g., 5G/6G, decentralized machine learning, federated learning). He and his team actively engage in collaborations with leading universities/institutions and industry to explore future networking and networked systems. He participates industry standardization activities with IETF, ETSI, 3GPP, oneM2M, and IEEE. His research interests also include blockchain technologies and applications, 5G/6G systems, quantum internet, edge computing, and intelligent IoT. He is a Fellow of the IEEE for his contributions to IoT enabling technologies. He's the founding Editor-in-Chief of IEEE IoT Journal and is currently the Editor-in-Chief of IEEE Network. He holds more than 100 US granted patents.

Xu Li currently is a senior staff engineer at InterDigital Communications Inc. His research interests include 3GPP wireless systems, Internet-of-Things (IoT), blockchain technology, and data semantics. He has published technical papers on mainstream international journals and conferences, such as IEEE INFOCOM, IEEE TPDS, IEEE JSAC, etc. and has been on the technical program committee of major technical conferences such as IEEE Globecom, IEEE ICC, IEEE WCNC, etc. His current major activities include wireless system standardization (such as 3GPP, oneM2M, IETF, W3C, etc.) and he has more than 70 US approved/pending patent applications.



Ning Zhang [M] received the Ph.D. degree from Virginia Tech in 2016. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Washington University in St. Louis. Before that, he was with an industry as a cyber engineer and a technical lead for over ten years. His research focus is system security, which lies at the intersection of security, embedded systems, computer architecture, and software.

Y. Thomas Hou [F'14] received his Ph.D. from NYU Tandon School of Engineering in 1998. He is currently Bradley Distinguished Professor of Electrical and Computer Engineering at Virginia Tech, Blacksburg, VA, USA, which he joined in 2002. He was a Member of Research Staff at Fujitsu Laboratories of America in Sunnyvale, CA from 1997 to 2002. His current research focuses on developing innovative real-time solutions to complex science and engineering problems arising from wireless and mobile networks. He is also interested in wireless security. He has published over 350 papers in IEEE/ACM journals and conferences. His papers were recognized by 10 best paper awards from IEEE and ACM, including an IEEE INFOCOM Test of Time Paper Award in 2023. He holds six U.S. patents. He authored/co-authored two graduate textbooks: *Applied Optimization Methods for Wireless Networks* (Cambridge University Press, 2014) and *Cognitive Radio Communications and Networks: Principles and Practices* (Academic Press/Elsevier, 2009). Prof. Hou was named an IEEE Fellow for contributions to modeling and optimization of wireless networks. He was/is on the editorial boards of a number of IEEE and ACM transactions and journals. He was Steering Committee Chair of IEEE INFOCOM conference and was a member of the IEEE Communications Society Board of Governors. He was also a Distinguished Lecturer of the IEEE Communications Society.

Jeffrey H. Reed [F'04] is the Willis G. Worcester Professor of Electrical and Computer Engineering (ECE) in the Bradley Department of Electrical and Computer Engineering at Virginia Tech. He is the Founding Director of Wireless @ Virginia Tech, one of the largest wireless research groups in the United States, and the previous Interim Director and now CTO for the Commonwealth Cyber Initiative for the State of Virginia. In 2010, He founded the Ted and Karyn Hume Center for National Security and Technology and served as its interim director. His current areas of expertise are in software-defined radios (SDRs), AI-enabled 5G wireless, wireless security/information assurance, interference analysis, and vehicular communications.