AsyncSC: An Asynchronous Sidechain for Multi-Domain Data Exchange in Internet of Things

Lingxiao Yang, Xuewen Dong✉, Zhiguo Wan, Sheng Gao, Wei Tong, Di Lu, Yulong Shen, Xiaojiang Du

IEEE INFOCOM 2025 - IEEE Conference on Computer Communications | 979-8-3315-4305-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/INFOCOM55648.2025.11044780

***Abstract*—Sidechain techniques improve blockchain scalability and interoperability, providing decentralized exchange and cross-chain collaboration solutions for Internet of Things (IoT) data across various domains. However, current state-of-the-art (SOTA) schemes for IoT multi-domain data exchange are constrained by the need for synchronous networks, hindering efficient cross-chain interactions in discontinuous networks and leading to suboptimal data exchange. In this paper, we propose AsyncSC, a novel asynchronous sidechain construction. It employs a committee to provide Cross-Blockchain as a Service (C-BaaS) for data exchange in multi-domain IoT. To fulfill the need for asynchronous and efficient data exchange, we combine the ideas of aggregate signatures and verifiable delay functions to devise a novel cryptographic primitive called delayed aggregate signature (DAS), which constructs asynchronous cross-chain proofs (ACPs) that ensure the security of cross-chain interactions. To ensure the consistency of asynchronous transactions, we propose a multilevel buffered transaction pool that guarantees the transaction sequencing. We analyze and prove the security of AsyncSC, simulate an asynchronous communication environment, and conduct a comprehensive evaluation. The results show that AsyncSC outperforms SOTA schemes, improving throughput by an average of 1.21 to 3.96 times, reducing transaction latency by 59.76% to 83.61%, and maintaining comparable resource overhead.**

I. Introduction

The proliferation of the Internet of Things (IoT) has led to ubiquitous data interactions between smart terminals and sensors [1]. For instance, in mobile crowd-sensing, individuals with smartphones and wearables can share data to achieve comprehensive environmental perception [2]. According to Cisco, the number of IoT devices worldwide is expected to reach 500 billion by 2030 [3]. Thus, equipping IoT with a secure, efficient, and scalable multi-domain data exchange infrastructure is crucial. However, most current IoT data storage and sharing infrastructures are centralized, requiring IoT organizations across domains to trust a third-party center for data management, resulting in scalability issues and single point of failure risks [4].

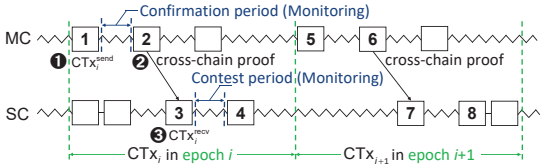


Fig. 1. Illustration for current sidechain-based cross-chain interactions. Wavy lines indicate omitted blocks. Blocks: **1.** $\mathsf{CTx}_i^{\mathsf{send}}$ is initiated; **2.** The cross-chain proof of $\mathsf{CTx}_i^{\mathsf{send}}$ is sent to SC; **3.** A $\mathsf{CTx}_i^{\mathsf{recv}}$ is created; **4.** $\mathsf{CTx}_i^{\mathsf{recv}}$ is stabilized on SC, and then CTx_i is completed; **5 to 8.** The CTx_{i+1} is executed at the next epoch.

The increasing interest in blockchain technology has driven researchers to explore decentralized solutions to enhance IoT capabilities [5]–[7]. However, due to resource limitations of IoT devices and potential privacy concerns, it is difficult to directly apply public blockchains to IoT scenarios. Permissioned blockchains offer data immutability and multi-party maintenance benefits of public blockchains while providing higher performance, flexibility, and privacy protection [8]. Consequently, some solutions explore using permissioned blockchains to meet the dynamic high-performance requirements of IoT scenarios [9], but still require cross-chain interaction capabilities between permissioned blockchains [10].

Sidechain provides a secure and efficient cross-chain solution with two modes: (i) *Parent-child mainchain-sidechain*. The traditional sidechains are pegged to the mainchain and achieve scaling by offloading traffic from the mainchain. Sidechain nodes monitor the mainchain states, and the mainchain verifies the cross-chain proofs from the sidechain maintainers for cross-chain interactions [11]. (ii) *Parallel sidechain*. Sidechain and mainchain are independent parallel blockchains, which can act as each other's sidechain and verify mutual cross-chain proofs for two-way cross-chain interactions [12], [13]. Considering context, the latter is more suitable for data exchange among organizations in different IoT domains. Each organization runs an independent permissioned blockchain to manage IoT data and cross-chain communication without third-party dependency through cross-chain proofs.

Motivation. (i) Currently, sidechain applications focus on cryptocurrency exchange on public blockchains, prioritizing security over performance in synchronous network settings [11]–[14], as shown in Fig. 1. During cross-chain processes, the mainchain and sidechain should continuously monitor

✉ Corresponding author: *Xuewen Dong* (xwdong@xidian.edu.cn).

Lingxiao Yang and *Xuewen Dong* are with the School of Computer Science and Technology, Xidian University, the Engineering Research Center of Blockchain Technology Application and Evaluation, Ministry of Education, and also with the Shaanxi Key Laboratory of Blockchain and Secure Computing, Xi'an 710071, China. *Zhiguo Wan* is with the Zhejiang Lab, Hangzhou 311121, China. *Sheng Gao* is with the School of Information, Central University of Finance and Economics, Beijing 100081, China. *Wei Tong* is with the School of Information Science and Engineering, Zhejiang Sci-Tech University, Hangzhou 310018, China. *Di Lu* and *Yulong Shen* are with the School of Computer Science and Technology, Xidian University, and also with the Shaanxi Key Laboratory of Network and System Security, Xi'an 710071, China. *Xiaojiang Du* is with the School of Engineering and Science, Stevens Institute of Technology, Hoboken 07030, USA.

This work is supported in part by the National Key R&D Program of China (2023YFB3107500), the National Natural Science Foundation of China (62220106004, 62232013, 62272425, 62072487), the Technology Innovation Leading Program of Shaanxi (2022KXJ-093, 2023KXJ-033), the Innovation Capability Support Program of Shaanxi (2023-CX-TD-02), and the Beijing Natural Science Foundation (L222050).

the block status, which increases the communication burden (see Section II-A for details). This does not meet the need for efficient cross-domain data exchange in discontinuous networks for resource-constrained devices. (ii) In permissioned blockchains, most schemes use trusted relays for cross-chain interactions [15]–[17], but the introduction of third-party relays adds complexity and privacy risk to the system. (iii) Recent advances utilize threshold signatures and zero-knowledge proof techniques to generate succinct cross-chain proofs [18]–[20]. However, to ensure the consistency of cross-chain interactions, strict synchronization between blockchains is still required, and the continuous monitoring of blocks results in significant on-chain overhead.

In addition, asynchronous cross-chain interaction differs from the synchronous method in that asynchronous cross-chain protocols typically achieve higher performance. This is because it departs from the strict sequential nature of transaction execution in synchronous mode and provides asynchronous concurrency. However, protocol design under asynchronous conditions poses greater challenges and must address how to ensure data consistency and security across different blockchains in the absence of a global clock.

Challenge. Ensuring the persistence and liveness of blockchains for cross-chain interactions without synchronous network settings is a significant challenge for asynchronous cross-chain communication. In other words, it is necessary to ensure that valid asynchronous cross-chain transactions can be smoothly executed, eventually recorded in blocks, and confirmed as stable by a sufficient number of subsequent blocks. This poses the following technical challenges: (i) How to construct a cross-chain interaction mode in asynchronous environments to enable efficient cross-domain data exchange for nodes with resource constraints and unstable network connections. (ii) How to design lightweight and secure asynchronous cross-chain proofs to reduce the computational burden. (iii) How to ensure the sequential consistency of cross-chain transactions in asynchronous mode.

This paper proposes an **async**hronous **s**ide**c**hain alternative construction called AsyncSC, which focuses on a parallel sidechain model without the need for synchronous network settings between blockchains. It utilizes an internal trusted committee formed by blockchain maintainers instead of relying on external relays. The main **contributions** are as follows:

• **Asynchronous cross-chain mode.** We present the first asynchronous cross-chain mode suitable for proof-of-stake (PoS) public blockchains and permissioned blockchains, which can construct an honest majority committee. The security of cross-chain interactions is ensured by lightweight asynchronous cryptographic proofs committed by the committee. The committee cyclically provides Cross-Blockchain as a Service (C-BaaS) to enable efficient data interactions across multiple IoT domains in network-unstable environments.

• **Innovative cryptographic proofs.** We introduce a novel cryptographic primitive called delayed aggregate signature (DAS) to generate asynchronous cross-chain proofs (ACPs) underpinning IoT multi-domain data exchanges. DAS combines the ideas of aggregate signatures and verifiable delay functions (VDFs) to generate an aggregate signature for multiple transactions after a controlled delay and supports public fast verification. As the foundation of C-BaaS, DAS is crucial for secure asynchronous cross-chain interactions, enabling the continuous generation of ACPs in asynchronous mode.

• **Restricted-readable consistency mechanism.** We design a transaction sequence guarantee mechanism to ensure the consistency of asynchronous cross-chain transactions. It maintains a multilevel buffer pool to coordinate the main-chain and sidechain for transaction ordering and confirmation, thus avoiding conflicts. The mechanism ensures the internal confidentiality of different IoT data domains, making the transaction sequence readable only to the authenticated leader.

• **Comprehensive evaluation.** We implement a prototype based on Hyperledger Fabric [21], and ChainMaker [22] and the evaluation results demonstrate that AsyncSC outperforms SOTA solutions. It improves transaction throughput by an average of $1.21 \sim 3.96\times$, reduces latency by 59.76% to 83.61%, enhances success ratio by a relative 46.03% to 135.9%, and maintains comparable overhead.

II. Background and Preliminaries

A. Current Sidechain-Based Cross-Chain Interactions

To achieve better security, existing sidechain-based cross-chain interaction schemes [11]–[14], [18] require the main-chain (MC) and sidechain (SC) to perform cross-chain transactions (CTxs) in (semi-)synchronous network settings [23]. The term "synchronous setting" assumes that the maintainers of MC and SC have roughly synchronized clocks to denote the current time slot. Message exchange among transaction participants incurs a maximum delay of Δ time slots. A scenario with $\Delta = 0$ is referred to as a synchronous model, while a scenario with $\Delta > 0$ is termed semi-synchronous.

We now proceed to illustrate the example in Fig. 1 under synchronous network mode. Assuming a CTx_i for cryptocurrency exchange from MC to SC, it typically involves three steps: ❶ A $\mathsf{CTx}_i^{\mathsf{send}}$ locks coins on MC. ❷ MC generates a cross-chain proof committing to the execution of $\mathsf{CTx}_i^{\mathsf{send}}$ after monitoring the stabilization of the block recording $\mathsf{CTx}_i^{\mathsf{send}}$ on the ledger L_{MC} through a sufficient number of subsequent block confirmations[1] (also called the confirmation period [11]). ❸ SC receives the cross-chain proof under the synchronous network, verifies it and executes the $\mathsf{CTx}_i^{\mathsf{recv}}$ to create the corresponding coins. Similarly, CTx_i is not successfully executed until the block containing $\mathsf{CTx}_i^{\mathsf{recv}}$ is stabilized on the L_{SC} (*i.e.*, after the contest period). The subsequent CTx_{i+1} is not executed until the above steps of CTx_i are completed.

The primary performance bottleneck in the existing cross-chain interaction process arises from the continuous monitoring required by MC and SC to determine the stability of $\mathsf{CTx}_i^{\mathsf{send}}$ and $\mathsf{CTx}_i^{\mathsf{recv}}$, respectively, before initiating a new CTx_{i+1}. The CTxs are sequentially executed under the

[1]In the Bitcoin blockchain, it typically takes six subsequent blocks to confirm the stability of a transaction within a block.

TABLE I: FEATURE COMPARISON WITH EXISTING SIDECHAINS.

Schemes \ Features	Sidechain mode	Cross-chain proof[1]	Asynchronous network/mode	Batch commitment[2]	IoT data exchange[3]
Pegged sidechains [11]	Parent-child	SPV proof	✗	✗	✗
PoW sidechains [12]	Parallel	NIPoPoWs	✗	✗	✗
PoS sidechains [13]	Parallel	ATMS	✗	✗	✗
Ref. [14], Ge-Co [18]	Parent-child	TSS	✗	✗	✗
AsyncSC (This work)	Parallel	ACP	✓	✓	✓

[1] SPV stands for simplified payment verification; NIPoPoWs stands for non-interactive proofs of proof-of-work; ATMS stands for ad-hoc threshold multi-signatures; TSS stands for threshold signature schemes.
[2] Each cross-chain proof in existing sidechains supports committing only a single transaction at a time.
[3] Existing sidechains only support cross-chain exchanges involving cryptocurrencies.

synchronous network. Furthermore, the monitoring processes of MC and SC necessitate constant state synchronization, thus increasing the communication load. In addition, we compare the features of AsyncSC with existing sidechains in Table I.

B. Cross-Chain Interactions in Asynchronous Networks

To meet the needs of resource-constrained multi-domain IoT devices for cross-chain data exchange in discontinuous networks, IoT devices must be able to store data locally while offline and upload it to the blockchain once online. This aspect is not the primary focus of this paper, as existing implementations are already mature [24]–[26]. More importantly, cross-chain interactions between blockchains need to support asynchronous transmissions.

To distinguish our proposal from existing work, we elucidate cross-chain interactions in asynchronous networks. As shown in Fig. 2, under synchronous networks, cross-chain proofs are assumed to be transmitted from MC to SC within a finite time, and the timing required for each CTx does not overlap. However, in this paper's context, message transmission between MC and SC may be interrupted, causing CTxs in synchronous mode to be blocked. To ensure proper business operations, we propose allowing the timings of CTxs to partially overlap in asynchronous mode.

Specifically, after $\mathsf{CTx}_i^{\mathsf{send}}$ has been executed (*i.e.*, the transaction is signed, consensused, and recorded on-chain) on MC, the generation of cross-chain proof can begin. The cross-chain proof of MC is delayed to be output after $\mathsf{CTx}_i^{\mathsf{send}}$ is stabilized, and this delayed output can be verified by SC. In other words, this delayed cross-chain proof commits the stability of $\mathsf{CTx}_i^{\mathsf{send}}$ in MC, so MC does not need to continuously monitor the state of $\mathsf{CTx}_i^{\mathsf{send}}$. This reduces communication overhead and allows MC to concurrently execute the next CTx_{i+1}, thereby improving transaction efficiency.

In an asynchronous network, MC's cross-chain proofs may be transmitted to SC after unknown delays due to unstable communications. Thus, SC needs to ensure that the order of asynchronous transactions is consistent with MC. We will explain this issue further in Section III-B.

C. Cryptographic Primitives

Aggregate Signature. The aggregate signature is a cryptographic digital signature technique that supports aggregation [27]. Given n signatures from n different users on n different messages, it can aggregate these signatures into a single, short signature. This aggregate signature convinces the verifier that n users indeed signed the n original messages (*i.e.*, user i signed message M_i for $i = 1, ..., n$). This reduces storage, transmission, and verification costs. In blockchain, aggregate signatures enable batch verification of transactions [28].

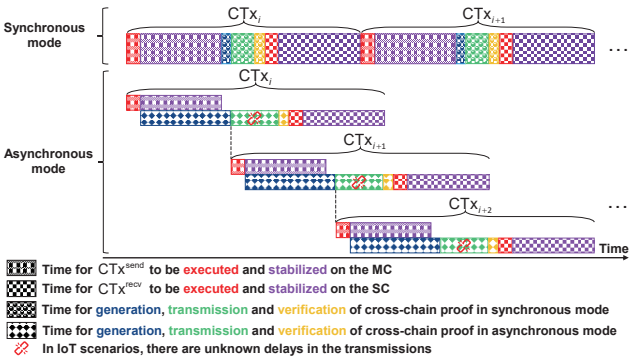


Fig. 2. Illustration for asynchronous cross-chain interactions.

Verifiable Delay Function. The verifiable delay function (VDF) is a cryptographic function [29]. It allows a prover to use the public parameters pp to generate a deterministic, unique value y corresponding to the input x and a proof π by running an evaluation algorithm $(y, \pi) \leftarrow \mathtt{Eval}(pp, x)$ with input arbitrary data x. The execution of $\mathtt{Eval}$ requires a specified number of consecutive steps, causing a delay predetermined by a delay parameter t. Once $\mathtt{Eval}$ is computed, anyone can use the public parameters pp to quickly and publicly verify the output of $\mathtt{Eval}$ via $\{accept, reject\} \leftarrow \mathtt{Verify}(pp, x, y, \pi)$.

III. ASYNCSC: SYSTEM OVERVIEW

This section introduces the AsyncSC system model, discusses the transaction consistency problem in asynchronous cross-chain mode, and presents the assumptions.

A. System Model

This paper focuses on multi-domain data exchange scenarios in IoT, as illustrated in Fig. 3. IoT sensor data is uploaded to permissioned blockchains managed by different organizations via transport nodes like IoT gateways and base stations for decentralized management. Independent blockchains in a parallel chain relationship require cross-chain interaction for data exchange among different organizations. Due to resource constraints of IoT devices and unstable network connections, cross-chain communication is achieved by generating and verifying asynchronous cross-chain proofs (ACPs).

Next, we describe the entities involved in cross-chain interactions and the workflow depicted in Fig. 4.

Entities. AsyncSC includes the following three entities.

Organization. It consists of permissioned blockchain nodes S^{Org} that maintain data from different IoT domains. These nodes may be elected to a committee to provide Cross-Blockchain as a Service (C-BaaS), which records data from IoT devices on-chain and facilitates cross-chain interactions.

Committee. It is a set of nodes $S^C = \{C_i\}|_{i=1}^m \in S^{Org}$ that are fairly elected from the organization by periodically running an election protocol[2] via smart contracts. These nodes provide C-BaaS, verify and sign transactions $\{\mathsf{CTx}_i\}|_{i=1}^n$ packed per

[2]Existing committee election schemes are well established, as noted in the Ref. [18], [30]–[32], and their optimization is part of our future work.

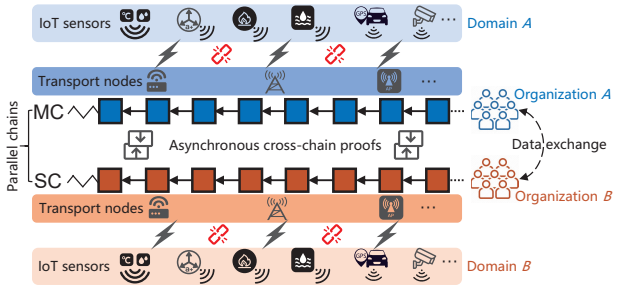


Fig. 3. System model of AsyncSC.

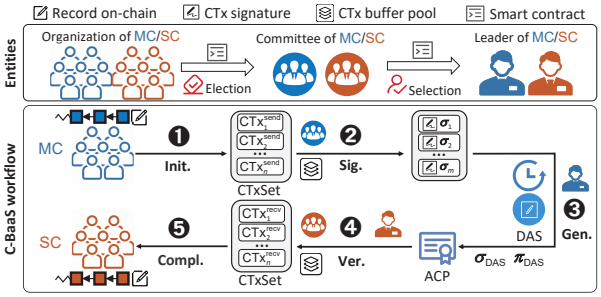


Fig. 4. Cross-chain interaction entities and workflow in AsyncSC.

epoch, and generate ACPs by the leader. Additionally, the committee maintains a buffer pool to ensure the consistency of asynchronous CTxs.

Leader. It is a node C^L with the highest election value in the committee, *i.e.*, $C^L \in S^C$ and $C^L \leftarrow \texttt{max.ElecVal}(\{C_i\}|_{i=1}^m)$. It is responsible for performing delayed signature aggregation of the CTx set $\mathsf{CTxSet} := \{\mathsf{CTx}_i\}|_{i=1}^n$ for each epoch of the committee and generating ACPs. Additionally, it verifies the received ACPs and finalizes the corresponding CTxs.

Workflow. The C-BaaS consists of the following steps:

❶ ***CTx initiation.*** Suppose in an epoch, the organization to which MC belongs initiates a batch of CTxs, *i.e.*, $\{\mathsf{CTx}_i\}|_{i=1}^n$. The nodes of MC start to record $\{\mathsf{CTx}_i^{\mathsf{send}}\}|_{i=1}^n$ on-chain. At the same time, the committee of MC receives a CTxSet message formed by packing these transactions.

❷ ***Committee signing.*** The committee of MC starts to provide C-BaaS. First, the committee puts the transactions of CTxSet into the buffer pool. Then, each committee member $C_i \in S^C$ signs the CTxSet using his/her private key via any EUF-CMA-secure digital signature scheme, *i.e.*, $\sigma_i \leftarrow \texttt{Sig}(\mathsf{CTxSet}, sk_i)$ for $i = 1, ..., m$. Finally, the committee sends a sequence of $\{(pk_i, \sigma_i)\}|_{i=1}^m$ to the leader of MC.

❸ ***ACP generation.*** The leader of MC executes a delayed signature aggregation on the committee's signatures, *i.e.*, $(\sigma_{\mathsf{DAS}}, \pi_{\mathsf{DAS}}, \cdot) \leftarrow \texttt{DASig}(\cdot)$. Here, the $\cdot$ notation indicates that some inputs or outputs of the algorithm are omitted (details in Section IV-A). The outputs of DASig along with the public parameter pp form an ACP representing the validity commitment of CTxSet on MC.

❹ ***ACP verification.*** Upon receiving the ACP from MC, the leader of SC verifies the validity of the ACP, *i.e.*, $1/0 \leftarrow \texttt{DASVer}(\cdot)$. Upon successful verification, the committee of SC queues the transactions within CTxSet into the buffer pool awaiting on-chain recording. To ensure the consistency of asynchronous CTxs, the SC committee conducts a transaction sequentiality check utilizing the buffer pool maintained by MC to prevent conflicts.

❺ ***CTx completion.*** The organization to which SC belongs records the $\{\mathsf{CTx}_i^{\mathsf{recv}}\}|_{i=1}^n$ in CTxSet on-chain. Once these transactions are stabilized within the ledger L_{SC}, the corresponding CTxs for this epoch are finalized.

B. Problem Statement

Due to unstable network connections across various IoT domains, asynchronous communication occurs between blockchains. For instance, during epoch i, the SC may not promptly receive the ACP_i generated by MC for that epoch, leading to potential delays in the workflows ❹ and ❺ as depicted in Fig. 4. Subsequent transactions on SC may conflict if they depend on transactions committed by ACP_i.

Specifically, suppose at epoch i, MC records a temperature sensor data as 10 ℃. At epoch $i+1$, the ambient temperature rises by 5 ℃, updating the data to 15 ℃. If SC does not receive the temperature data from epoch i in time due to network delay or other reasons, it may record the data at epoch $i+1$ directly as 15 ℃ and ignore the 10 ℃ that should have been recorded. This results in the data being recorded out of order, thereby affecting the consistency and accuracy of the system.

To avoid this issue, we address the consistency problem of asynchronous CTxs by maintaining transaction buffer pools via committees and devising an asynchronous transaction sequence guarantee mechanism (see Section V-A).

C. Assumptions

Assumptions. The upper bound Δ on message transmission delay due to asynchronous communication may be unknown. In practice, we assume a negligible probability of $\Delta = \infty$ between MC and SC. To ensure safety and liveness, MC retransmits the CTxs to SC if they exceed an asynchronous response time threshold Δ_{async}. The CTxs of each epoch eventually complete their execution through the C-BaaS provided by the committee. If the verification of MC's cross-chain proof passes, then SC will accurately complete the corresponding cross-domain data on-chain and stabilize it. To minimize the overhead caused by the committee elections, we assume a long-term election period, with each period containing multiple epochs for packing transactions.

Threat model. We consider that an adversary may attempt to violate the protocol by forging signatures or delaying/denying message transmission. However, the adversary's behavior is rational, *i.e.*, it weighs the costs and benefits of the attack. As in existing blockchain systems, we assume that the nodes maintaining the permissioned blockchains are majority honest [14], [33]. The adversary does not possess enough computational power to break the participants' signatures or compromise blockchains and smart contracts.

IV. Delayed Aggregate Signature

In this section, we introduce a novel cryptographic primitive known as the delayed aggregate signature (DAS), which serves as the fundamental building block of ACP and the core of C-BaaS. DAS integrates properties from aggregate signatures and

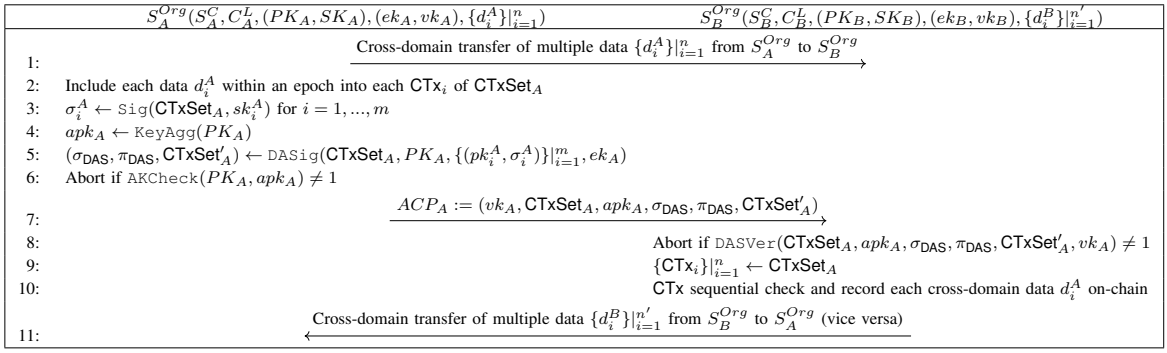


Fig. 5. Procedure of applying DAS to AsyncSC. In the figure, organization *A* cross-domain transfers data $\{d_i\}|_{i=1}^n$ within an epoch to organization *B* and vice versa. (PK_A, SK_A) represents the public-private key pairs of the committee nodes $S_A^C \in S_A^{Org}$ in organization *A*. (ek_A, vk_A) are the publicly accessible evaluation and verification keys used in DAS by organization *A*.

VDFs. It provides a way to: (i) Aggregate multiple individual signatures $\{\sigma_i\}|_{i=1}^m$ into a single aggregate signature σ_{DAS} after a controllable delay δ and produce a proof of delay π_{DAS}. (ii) Enable the verifier to quickly and publicly verify the σ_{DAS} and π_{DAS}, thereby effectively verifying the authenticity of $\{\sigma_i\}|_{i=1}^m$ and the delay δ.

A. Definition of Delayed Aggregate Signature

Definition 1. (Delayed Aggregate Signature). *A delayed aggregate signature is a tuple of algorithms* $\Pi = (\mathtt{ParGen}, \mathtt{KeyGen}, \mathtt{Sig}, \mathtt{Ver}, \mathtt{KeyAgg}, \mathtt{AKCheck}, \mathtt{DASig}, \mathtt{DASVer})$*, whose syntax is described below:*

- $(sp, pp) \leftarrow \mathtt{ParGen}(1^\lambda, t)$: *A parameter generation algorithm that takes as input the security parameter* λ *and the delay parameter* t*, and outputs the global system parameters* sp *and the public parameters* $pp = (ek, vk)$ *consisting of an evaluation key* ek *and a verification key* vk*.*

- $(pk_i, sk_i) \leftarrow \mathtt{KeyGen}(sp)$: *A key generation algorithm that inputs* sp *and generates a public-private key pair* (pk_i, sk_i) *for the invoker.*

- $\sigma_i \leftarrow \mathtt{Sig}(sk_i, M)$: *A common signature algorithm that inputs the private key* sk_i *and a message* $M \in \{0,1\}^*$*, and outputs a signature* σ_i*.*

- $1/0 \leftarrow \mathtt{Ver}(M, pk_i, \sigma_i)$: *A signature verification algorithm that inputs the message* M*, the public key* pk_i *and the signature* σ_i*, and outputs 1 (accept) if the verification is successful and 0 (reject) otherwise.*

- $apk \leftarrow \mathtt{KeyAgg}(PK)$: *A key aggregation deterministic algorithm that inputs a set of public keys* $PK = \{pk_i\}|_{i=1}^m$ *and outputs an aggregate public key* apk*.*

- $1/0 \leftarrow \mathtt{AKCheck}(PK, apk)$: *An aggregate key checking algorithm that inputs a public key set* PK *and an aggregate public key* apk*, and outputs 1 if the recomputation check is correct and 0 otherwise.*

- $(\sigma_{\mathsf{DAS}}, \pi_{\mathsf{DAS}}, M') \leftarrow \mathtt{DASig}(M, PK, \{(pk_i, \sigma_i)\}|_{i=1}^m, ek)$: *A delayed aggregation signature algorithm that inputs the message* M*, the set of public keys* PK*, a sequence of public key-signature pairs* $\{(pk_i, \sigma_i)\}|_{i=1}^m$ *and the evaluation key* ek*, and outputs a delayed aggregate signature* σ_{DAS}*, a proof of delay* π_{DAS} *and a unique output* M' *corresponding to message* M*.*

- $1/0 \leftarrow \mathtt{DASVer}(M, apk, \sigma_{\mathsf{DAS}}, \pi_{\mathsf{DAS}}, M', vk)$: *A delayed aggregate signature verification algorithm that inputs the message* M*, the aggregate public key* apk*, the delayed aggregate signature* σ_{DAS}*, the delayed proof* π_{DAS}*, the unique output* M' *of the message* M *and the verification key* vk*, and outputs 1 if the verification is successful and 0 otherwise.*

B. Realization of Delayed Aggregate Signature

The specific realization of DAS is partially inspired by two primitives: aggregate signatures and VDFs, both initially proposed by Boneh *et al.* [27], [29]. Aggregate signatures or multi-signatures have been improved by Ref. [28], [34], [35], while VDFs have been enhanced by Ref. [36], [37]. It is noteworthy that the improvements on these primitives are independent of this paper. We extract the signature compression property of aggregate signatures and the delay verifiability property of VDFs to implement compact ACPs.

We refer to constructions in the multi-signature scheme [38] and the VDF scheme for incremental verifiable computation (IVC) [29] to realize the DAS scheme. It relies on Gap Diffie-Hellman (GDH) groups and succinct non-interactive arguments of knowledge (SNARK) for its efficient construction. Due to the page limit of this submission, the detailed construction of the DAS, along with its security definition and proof, can be found in the full version published online.

C. Applying DAS to AsyncSC

To provide the C-BaaS to the organizations managing different IoT domains, the committee S^C of each organization S^{Org} runs $\mathtt{ParGen}$ and $\mathtt{KeyGen}$ to obtain the parameters (sp, pp) and assign the (pk_i, sk_i) pairs.

Fig. 5 illustrates the procedure for the cross-domain transfer of packaged data set $\{d_i^A\}|_{i=1}^n$ within an epoch from organization *A* to organization *B*. First, the nodes S_A^{Org} of organization *A* initiate CTxs. Each data d_i^A within the epoch is included in its corresponding CTx_i, and these CTxs form a CTxSet_A (Line 2). The committee nodes S_A^C then sign CTxSet_A (Line 3). Subsequently, the leader C_A^L aggregates the public keys PK_A of S_A^C with $\mathtt{KeyAgg}$ (Line 4).

The main procedures then include ACP generation and verification. The fundamental concept underlying asynchronous C-BaaS is that the delay δ introduced by executing $\mathtt{DASig}$ ensures the stabilization of transactions within CTxSet in the ledger of the source chain. Simultaneously, when the committee aggregates the signatures of CTxSet, it confirms

the transactions as anticipated. Thus, ACP_A commits to the validity of CTxSet_A on organization A's blockchain. Consequently, organization B can process corresponding cross-domain transactions on its blockchain whenever ACP_A is received, provided the verification is successful. Thus, DAS enables the continuous generation of ACPs in a discontinuous network and supports rapid public verification of these ACPs.

As described earlier, the leader of organization A obtains a validity commitment $(\sigma_{\mathsf{DAS}}, \pi_{\mathsf{DAS}}, \mathsf{CTxSet}'_A)$ by running `DASig` (Line 5). If the committee's aggregate public key apk_A check passes (Line 6), confirming that ACP_A is indeed produced by the honest committee, the leader forwards the DAS commitment and the pre-existing information in an ACP message to organization B (Line 7). After receiving ACP_A, any node of organization B, typically the leader, can verify the validity of the DAS commitment by running `DASVer` (Line 8). Essentially, `DASVer` involves checking both σ_{DAS} and π_{DAS}. If both verifications are successful, the authenticity of the cross-domain data $\{d_i^A\}|_{i=1}^n$ within CTxSet_A and the associated delay δ are confirmed. Consequently, this data set is certified as stable and tamper-proof on organization A's blockchain.

Finally, organization B performs a sequential check of the transactions in CTxSet_A (Lines 9-10). If the check does not conflict with the order maintained by the source chain, each corresponding cross-domain data d_i^A is recorded onto the blockchain. Once the transactions stabilize on the target blockchain, the cross-domain data for this epoch is transferred. Similarly, the process of transferring data from organization B to organization A follows the same procedure (Line 11).

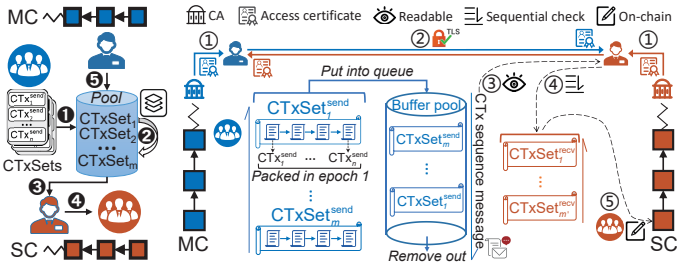
V. System Details of AsyncSC

In this section, we further describe the asynchronous transaction sequence guarantee mechanism and discuss the details of the delay settings of the DAS.

A. Transaction Sequence Guarantee Mechanism

To maintain the consistency of asynchronous CTxs, as shown in Fig. 6, we propose a restricted-readable transaction sequence guarantee mechanism. This mechanism can be implemented in various ways; here, we provide a buffer pool-based implementation idea. The analysis and optimization of its utility are left for future work.

The sequential check consists of five main steps, as shown in Fig. 6(a). ❶ *Receiving*. When the MC's committee receives the CTxSet for each epoch, its leader maintains these CTxs in the local multilevel buffer pool. To improve efficiency and minimize conflicts, we draw inspiration from Ref. [39]. The multilevel buffer pool uses an adaptive heap structure to categorize and store transactions according to different priorities (*e.g.*, timestamp, importance, dependency). The adaptive heap dynamically adjusts the order based on the current load situation and the dependencies of the CTxs, ensuring that CTxs with high priority and early timestamps are located at the top of the heap. ❷ *Ordering*. The MC leader dynamically adjusts the order of the pooled CTxs through a sliding window. It also dynamically adjusts the window size and movement



(a) Checking steps. (b) CTx sequence restricted readable.

Fig. 6. Asynchronous transaction sequence guarantee mechanism.

speed based on real-time transaction traffic. ❸ *Processing*. The leader of SC performs conflict checking on the CTxs in each CTxSet received asynchronously based on the buffer pool maintained by MC, and then sends the processed set of CTxs to the committee for confirmation. ❹ *Confirming*. The committee of SC uses smart contracts to automate the confirmation of the transaction order and ensure consistency through BFT consensus algorithms such as Tendermint [40]. After confirming that there are no conflicts, the relevant CTxs are executed in an orderly manner. ❺ *Removing*. Finally, after confirming that the CTxs have stabilized through a persistent query on the ledger, the leader of MC removes the executed CTxs from the buffer pool.

In the context of this paper, to adhere to the internal confidentiality of permissioned blockchains, the CTx sequence maintained by the buffer pool is restricted to be readable only by authenticated leaders, as shown in Fig. 6(b). ① Upon leader nodes' access to the permissioned blockchains, certificate authorities (CAs) issue access certificates to them. ② When cross-chain interactions are required, leaders on both sides exchange certificates over a secure communication channel such as TLS to authorize each other. ③ Then, the leader of SC can receive a signed message from MC containing only the transaction sequence in the buffer pool. ④ The leader of SC then performs a sequential check on the received CTxs based on this message. ⑤ The committee of SC completes the on-chain recording of CTxs after checking.

B. Delay Settings for DAS

The core idea of implementing DAS lies in the fact that the `DASig` algorithm produces a controlled delay by performing a series of sequential computations on the input for a given number of steps, such as the iterative computation of the SNARK utilized in the IVC adopted in this paper. At the same time, it ensures that the results can be verified quickly. Under given hardware and algorithmic conditions, the time required to compute `DASig` for a given length of input is fixed. This predictability of computation time allows the system designers to set specific delays as needed. We ensure that each computation goes through the same required steps and time by choosing the appropriate delay parameter t in the parameter generation phase of the DAS.

With the workflow of Fig. 4 in mind, we now detail the settings for the size of the delay generated by the DAS. Our goal is to ensure that when SC receives the ACP, the

delay δ generated by running DASig guarantees that the CTxSet, *i.e.*, all of the CTxs in an epoch, have stabilized on MC. Thus, the key is to make the last transaction of this epoch on-chain stable. We now discuss the size of the ideal delay δ. These CTxs are recorded on MC in Fig. 4-❶. Assume that the last CTx on-chain in this epoch is of length $T_{LastCTx}$ from the last time slot of the epoch. Thus, $T_{LastCTx}$ is included in the duration required for the stabilization of CTxSet. Additionally, the DASig algorithm is started in Fig. 4-❸, *i.e.*, the time for the committee to carry out the signatures in Fig. 4-❷, T_{Sig}, is already accounted for in the length of time required for the stabilization of CTxSet. Let the time required for a transaction to stabilize on the permissioned blockchain be T_{Stab}, then the ideal delay $\delta = T_{Stab} - T_{LastCTx} - T_{Sig}$. Since T_{Stab} and T_{Sig} are essentially constant in a specific permissioned blockchain and $T_{LastCTx}$ is easily obtained, it is feasible to set a suitable delay parameter t to control δ.

In practice, the delay δ of DAS is a tradeoff between security and efficiency, and we usually set δ slightly larger than its ideal duration to ensure stronger security. The optimization of this tradeoff is left for future work. Additionally, if the transaction volume on MC surges or the intra-chain communication deteriorates, the block may take longer to stabilize, thus requiring a longer δ. For different blockchains, δ is adjusted according to the duration of their ledger state evolution [41].

VI. Security Analysis

We define and prove the security of AsyncSC in this section.

Persistence and liveness are key properties that ensure blockchain security [42]. In short, persistence ensures all honest participants agree on stable transactions and their order in the ledger. Liveness ensures transactions from any honest participant are eventually recorded in the ledger of all honest participants, guaranteeing continued ledger progress. Combined with the definition of correct cross-chain communication by Zamyatin *et al.* [41], we define the security of AsyncSC as follows.

Definition 2. (Security). *For MC and SC with persistence and liveness, AsyncSC must have the following properties:*

- **Correctness.** *If both CTx^{send} and CTx^{recv} are as expected, they will be recorded in the ledgers L_{MC} and L_{SC}, respectively. If either does not match, both parties will abort. Additionally, there is no case where CTx^{send} or CTx^{recv} is recorded in the ledger by only one party.*

- **Stability.** *If CTx^{send} is as expected, then eventually CTx^{send} will be recorded in L_{MC} and CTx^{recv} will be recorded in L_{SC}. Given a common prefix parameter $k \in \mathbb{N}$, the blocks containing CTx^{send} and CTx^{recv} are located more than k blocks away from the ends of L_{MC} and L_{SC}, respectively.*

The correctness property combines effectiveness and atomicity as described in Ref. [41]. It is a safety property that ensures CTxs match the parties' expectations and maintains the consistency of cross-chain interactions. The stability property guarantees the eventual termination of the cross-chain protocol, making it a liveness property.

Theorem 1. *Provided that MC and SC satisfy persistence and liveness properties, AsyncSC is considered secure.*

Proof. By Definition 2, an arbitrary probabilistic polynomial time (PPT) adversary $\mathcal{A}$ exists in two cases if it violates the correctness property. The notation can be expressed as:

- **C1:** $\mathsf{CTx}^{send} \in \mathsf{L}_{MC} \wedge \mathsf{CTx}^{recv} \notin \mathsf{L}_{SC}$.
- **C2:** $\mathsf{CTx}^{send} \notin \mathsf{L}_{MC} \wedge \mathsf{CTx}^{recv} \in \mathsf{L}_{SC}$.

For **C1**, the possible reason is that $\mathcal{A}$ delays/denies sending the ACP to SC. As we mentioned, MC will retransmit CTxs that exceed the threshold Δ_{async}, so the final state will be $\mathsf{CTx}^{send} \notin \mathsf{L}_{MC} \wedge \mathsf{CTx}^{recv} \notin \mathsf{L}_{SC}$. If the committee of MC forges signatures, then it needs to corrupt at least $\frac{|S^C|}{2} + 1$ members. This has a negligible probability of happening in an honest-majority permissioned blockchain, and violates the security of committee elections [30]–[32].

For **C2**, the possible reason is that the organization of SC forges signatures. Similarly, $\mathcal{A}$ needs to corrupt at least $\frac{|S^{Org}|}{2} + 1$ nodes, which has negligible probability. If $\mathcal{A}$ corrupts the leader of MC to forge ACP and make SC include CTx^{recv} in L_{MC}, then it needs to forge the MC committee's aggregate public key and the DAS output to pass DASVer. This obviously cannot hold, since a fake apk cannot pass AKCheck. Furthermore, the security of DAS relies on the collision-resistant hash function [38], and the soundness and correctness of the IVC scheme [29]. The probability that a PPT adversary $\mathcal{A}$ breaks through a publicly setup DAS is negligible.

Thus, the probability that $\mathcal{A}$ successfully violates the correctness property is negligible.

We further analyze whether $\mathcal{A}$ can violate the stability property. Suppose there is a valid CTx^{send}. Since MC satisfies the liveness property, it will eventually stabilize in L_{MC}, *i.e.*, $\mathsf{CTx}^{send} \in \mathsf{L}_{MC}$. During the protocol run, CTx^{send} will first be packed into the CTxSet. Then, the committee of MC signs the CTxSet. The leader of MC generates the ACP and sends it to SC. After the leader of SC verifies successfully, a sequential check of the CTxs in CTxSet will be performed. Since MC satisfies persistence, CTx^{recv} can maintain the same order as CTx^{send}. Thus, valid CTx^{recv} will be recorded in L_{SC}. Since SC satisfies liveness, the final state is $\mathsf{CTx}^{recv} \in \mathsf{L}_{SC}$. Thus, the stability of AsyncSC follows from this.

Therefore, AsyncSC satisfies correctness and stability, proving its security. □

VII. Performance Evaluation

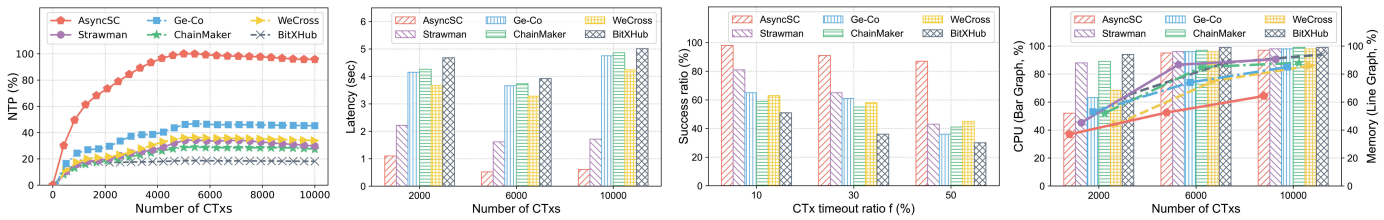
A. Implementation

To evaluate AsyncSC, we implement a prototype based on two well-known permissioned blockchains: Hyperledger Fabric [21] and ChainMaker [22], using the Go. We realize DAS by modifying the multi-signature MuSig2[3] and Harmony's VDF[4]. The prototype code will be made public soon.

Baselines: For a fair comparison, the baselines for evaluating AsyncSC include two cross-chain prototypes we developed

[3]https://github.com/aureleoules/musig2-coordinator

[4]https://github.com/harmony-one/vdf



(a) Throughput vs. the # of CTxs. (b) Latency vs. the # of CTxs. (c) Success ratio vs. the timeout ratio. (d) Overhead vs. the # of CTxs.

Fig. 7. Performance comparison of different baselines across various metrics.

and three enterprise-level projects supporting cross-chain interactions on permissioned blockchains. Their names and core ideas are as follows: (i) *Strawman*. An incomplete AsyncSC prototype using SPV proof for each CTx as a commitment. (ii) *Ge-Co* [18]. A sidechain scheme that uses committee voting for threshold signatures. (iii) *ChainMaker* [22]. Uses cross-chain proxy, SPV, and transaction contracts, supporting cross-chain operations with Fabric. (iv) *WeCross* [43]. Relay-based cross-chain routing that enables data interoperability between Fabric and FISCO BCOS [16]. (v) *BitXHub* [17]. Provides an InterBlockchain Transfer Protocol (IBTP) based on relay chains for reliable routing between permissioned blockchains.

B. Settings

Testbed: It consists of 16 virtual machines, each equipped with an Intel i7-13700 CPU, 8 GB of RAM, and a 5 Mbps network link. To simulate semi-synchronous/asynchronous communication, we use the `tc`[5] command on each machine to control transmission delay Δ ($100 \sim 500$ ms) and network jitter ($\pm$ 25 ms), and to generate CTx timeouts (*i.e.*, $\Delta > \Delta_{async}$) with a ratio of f. We launch 20 nodes on each machine using Docker containers. We run Hyperledger Fabric v2.4 on half of the machines and ChainMaker v2.1 on the other half to simulate cross-domain data exchanges using heterogeneous permissioned blockchains by different domains. Additionally, the versions of the compared projects are WeCross v1.2 and BitXHub v1.11.

Dataset: We use the real-world IoT dataset `TON_IoT`[6], which contains over two million pieces of raw telemetry data collected by weather and Modbus sensors [44]. Blockchains from different organizations first store different data and then exchange a number of data per epoch.

Metrics: The following metrics are used to evaluate the performance of cross-chain interactions. (i) *Throughput*. The number of successful executions of CTx per second by the system (TPS). (ii) *Latency*. The duration for a CTx to stabilize on SC from the time it is initiated by MC. (iii) *Success ratio*. The percentage of successful executions in initiated CTxs (TSR). (iv) *CPU and memory utilization*. The resource consumption of the system during execution.

C. Performance Comparison

We perform comprehensive CTx-driven simulations by adjusting the test parameters. The number of CTx per test ranges from 1,000 to 10,000. We control Δ to follow a normal distribution between 100 and 500 ms to simulate the typical semi-synchronous communication between IoT devices. At the same time, we control some of the CTxs to timeout (we set $\Delta_{async} = 10$ s) proportionally from 10% to 50% to simulate asynchronous communication. We set the epoch duration to 600 ms. The block generation time for each blockchain is 100 ms. A block is considered stabilized if it is confirmed by the next five blocks (*i.e.*, $T_{Stab} = 500$ ms). The detailed experimental results are shown in Fig. 7.

1) Throughput: We first normalize the throughput results (NTP) to eliminate environmental differences between the different schemes and facilitate comparisons. The value of NTP reflects the efficiency of the system in highly concurrent data transfer. Fig. 7(a) shows the curves of throughput with an increasing number of CTx. Initially, the NTP of all schemes rises with the number of CTxs. The performance bottleneck is approached when the number of CTxs reaches 4,000 to 6,000. However, AsyncSC shows an average increase in throughput of $1.21 \sim 3.96 \times$ compared to other schemes. The reason for AsyncSC's advantage in throughput is twofold. First, AsyncSC improves proof efficiency by batch committing packaged CTxs through ACP. Second, the test simulates asynchronous communication with timed-out CTxs, where traditional synchronous schemes experience more transaction aborts. In contrast, AsyncSC demonstrates the benefits of asynchronous concurrency. Additionally, the actual throughput and our improvements for all schemes are shown in Table II.

TABLE II: THROUGHPUT OF ASYNCSC AND BASELINES.

Schemes	Strawman	Ge-Co	ChainMaker	WeCross	BitXHub	**AsyncSC**
Avg. TPS	457.93	652.70	402.92	497.40	290.82	**1442.47**
Avg. increase	2.15 ×	1.21 ×	2.58 ×	1.90 ×	3.96 ×	-

2) Latency: Fig. 7(b) shows the average latency from initiation to stabilization for each CTx at transaction numbers of 2,000, 6,000, and 10,000. The results show that AsyncSC reduces the latency by 59.76% to 83.61% compared to the other schemes. Notably, the average latency of AsyncSC is 0.74 s. Although there exists a message delay, AsyncSC uses ACP to commit an epoch of packed CTxs. Even if part of the transaction times out and retransmits, the asynchronous concurrency greatly reduces the overall transaction latency. AsyncSC's advantages in asynchronous large-scale data exchange scenarios will be more pronounced.

3) Success ratio: Fig. 7(c) shows the average success ratio of 10,000 CTxs tested with timeout ratio f set to 10%, 30%, and 50%. The results show that AsyncSC increases the TSR by 46.03% to 135.9% compared to the other schemes. Furthermore, the relative improvement of AsyncSC is more

[5]https://linux.die.net/man/8/tc

[6]https://research.unsw.edu.au/projects/toniot-datasets

TABLE III: COMPARISON OF CROSS-CHAIN PROOFS.

Schemes	PoW sidechains [12]	PoS sidechains [13]	Ge-Co [18]	**AsyncSC**
Proof size[1]	$\mathcal{O}(\log cl)$	$\mathcal{O}(k)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Comp. cost[2]	$\mathcal{O}(\log cl)$	$\mathcal{O}(k)$	$\mathcal{O}(\lvert S^C\rvert)$	$\mathcal{O}(\frac{\lvert S^C\rvert}{n})$
Gen. time[3]	1205.12 s	54.54 s	3.91 s	**512.3 ms**
Ver. time	30.97 s	24.72 s	618.37 ms	**83.6 ms**

[1] The symbol cl denotes the chain length, and k is the common prefix parameter.
[2] The $|S^C|$ denotes committee size, and n denotes the number of CTxs in an epoch.
[3] Measure the time to generate cross-chain proofs for an epoch containing 1,200 CTxs. We set $cl \approx 2.03 \times 10^7$ (as of July 2024 on ETH); $k = 2160$; and a committee size of 10.

pronounced as f increases. We find that schemes under traditional synchronous settings struggle to handle CTx delays or timeouts properly. Even if they can, transactions are usually aborted due to conflicts. However, AsyncSC does not require MC to continuously monitor the state of CTxs, and its asynchronous transaction sequence guarantee mechanism can effectively reduce the occurrence of transaction conflicts.

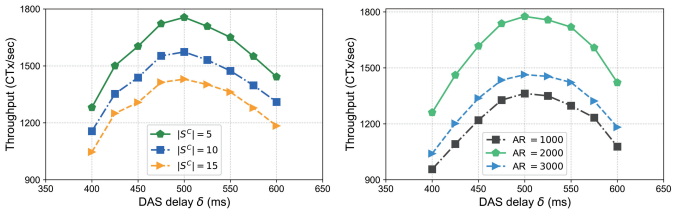
4) CPU and memory utilization: Fig. 7(d) shows the main results of the different schemes in terms of resource overhead during data exchange. It can be seen that the systems are almost fully loaded when the number of CTxs reaches 6,000, and thus the performance reaches a bottleneck. However, AsyncSC reduces the overall CPU and memory utilization by an average of 5.1% to 16.44% and 25.02% to 37.77%, respectively. This is due to the fact that AsyncSC uses ACP for batch CTx commitment and reduces the overhead of monitoring and synchronizing states between blockchains.

D. Evaluation of ACP

Table III provides a theoretical comparison between AsyncSC and some SOTA sidechain constructions. AsyncSC is based on the DAS realization of generating ACPs for n CTxs within an epoch. The cross-chain proofs are aggregated from the committee's signatures. In addition, delayed proofs are implemented in DAS using IVC iterative computations. For any sub-exponential length computation, the complexity in terms of proof size and verification cost is limited by $\mathsf{poly}(\lambda)$. This part of the proof size and computational cost is negligible [29]. Its size is not affected by the chain length or the common prefix. Thus, the proof size complexity of AsyncSC is $\mathcal{O}(1)$. The computational cost shared by a single CTx is $\mathcal{O}(\frac{|S^C|}{n})$. AsyncSC is more lightweight than the SOTA constructions.

We measure the generation and verification times for cross-chain proofs. The results in Table III show that the average generation and verification times of AsyncSC are 512.3 ms and 83.6 ms, respectively. Since AsyncSC needs to generate only one cross-chain proof for the CTxs of an epoch, which can be verified quickly, it reduces the proof generation and verification times by 1 to 4 and 1 to 3 orders of magnitude, respectively, compared to SOTA schemes.

Next, we vary the size of $|S^C|$ and n to evaluate the system throughput versus DAS delay of ACP, as shown in Fig. 8. We vary the DAS delay within the range $\delta = 400 \sim 600$ ms. Since Δ is a minimum of 100 ms and $T_{Stab} = 500$ ms, δ ensures CTx stability. The results show that throughput increases and then decreases with the increase of δ. This is because the size of δ affects the number of CTx arriving at an epoch, which in turn affects the adequacy of ACP aggregation. Fig. 8(a) shows the results of varying $|S^C|$ for a fixed CTx arrival rate (AR) of 2,000 per second. It can be seen that the larger $|S^C|$ is, the lower the throughput, but its effect is not significant. This is because the larger $|S^C|$, the more committee signatures need to be aggregated, and the aggregation process is rapid. In addition, we control the number of CTx per epoch (*i.e.*, n) by varying the AR. Fig. 8(b) shows the effect of different ARs on throughput. An AR that is too small or too large results in too few or too many CTxs being processed at each epoch, potentially causing the system to be underloaded or overloaded, thereby affecting system throughput.



(a) CTx arrival rate = 2000 CTx/sec. (b) Committee size $|S^C| = 5$.

Fig. 8. Evaluating system throughput vs. ACP's DAS delay.

TABLE IV: EVALUATING OF CTX BUFFER POOL.

Metrics	Queue size (# of CTx)			Window size (# of CTx)			Heap priority		
Options	1000	**2000**	3000	200	**400**	600	Time.	Imp.	**Dep.**
TSR (%)	90.45	**92.31**	91.17	91.36	**92.52**	91.02	92.12	90.87	**94.45**

E. Evaluation of CTx buffer pool

To verify the effectiveness of the transaction sequence guarantee mechanism in reducing conflicts, we evaluate the impact of different metric options for the buffer pool on the transaction success ratio. We attach random weights and 20% dependencies to the transactions. The timeout rate $f = 30\%$. Table IV presents the average results of our multiple control variable tests. Each metric displays three key options. The results indicate that moderately increasing the queue size can improve TSR, but an excessively large queue may reduce processing efficiency. A larger window helps improve TSR, but an excessively large window may increase the probability of conflicts. The dependency prioritization strategy improves TSR more than timestamp and importance prioritization, indicating that considering inter-transaction dependencies in multilevel buffer pools is effective.

VIII. CONCLUSION

To facilitate the trusted exchange of IoT multi-domain data, we propose AsyncSC, an efficient asynchronous sidechain construction. We present Cross-Blockchain as a Service (C-BaaS) based on a committee. To provide asynchronous cross-chain proofs (ACPs), we introduce a novel cryptographic primitive called the delayed aggregate signature (DAS). To ensure the consistency of asynchronous cross-chain transactions, we design a transaction sequence guarantee mechanism. Our analysis demonstrates the security of AsyncSC. A prototype evaluation of AsyncSC shows that it outperforms SOTA schemes. Our future work will focus on optimizing the guarantee of asynchronous transaction consistency and DAS delay strategies.

REFERENCES

[1] S. He, K. Shi, C. Liu, B. Guo, J. Chen, and Z. Shi, "Collaborative sensing in internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 3, pp. 1435–1474, 2022.

[2] J. Liu, H. Shen, H. S. Narman, W. Chung, and Z. Lin, "A survey of mobile crowdsensing techniques: A critical component for the internet of things," *ACM Transactions on Cyber-Physical Systems*, vol. 2, no. 3, pp. 1–26, 2018.

[3] Cisco. Powering an inclusive, digital future for all. [Online]. Available: https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2023/m01/powering-an-inclusive-digital-future-for-all.html

[4] Y. Liu, J. Wang, Z. Yan, Z. Wan, and R. Jäntti, "A survey on blockchain-based trust management for Internet of Things," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 5898–5922, 2023.

[5] W. Tong, X. Dong, Y. Shen, X. Jiang *et al.*, "A blockchain-driven data exchange model in multi-domain iot with controllability and parallelity," *Future Generation Computer Systems*, vol. 135, pp. 85–94, 2022.

[6] W. Tong, X. Dong, Y. Zhang, Z. Zhang, L. Yang, W. Yang, and Y. Shen, "TI-BIoV: Traffic information interaction for blockchain-based IoV with trust and incentive," *IEEE Internet of Things Journal*, 2023.

[7] X. Zhu, J. Zheng, B. Ren, X. Dong, and Y. Shen, "MicrothingsChain: Blockchain-based Controlled Data Sharing Platform in Multi-domain IoT," *Journal of Networking and Network Applications*, 2021.

[8] O. Dib, K.-L. Brousmiche, A. Durand, E. Thea, and E. B. Hamida, "Consortium blockchains: Overview, applications and challenges," *Int. J. Adv. Telecommun*, vol. 11, no. 1, pp. 51–64, 2018.

[9] Y. Feng, W. Zhang, X. Luo, and B. Zhang, "A consortium blockchain-based access control framework with dynamic orderer node selection for 5G-enabled industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2840–2848, 2021.

[10] M. Zhang, Q. Qu, L. Ning, and J. Fan, "On time-aware cross-blockchain data migration," *Tsinghua Science and Technology*, vol. 29, no. 6, pp. 1810–1820, 2024.

[11] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," vol. 72, 2014, pp. 201–224.

[12] A. Kiayias and D. Zindros, "Proof-of-work sidechains," in *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2019, pp. 21–34.

[13] P. Gaži, A. Kiayias, and D. Zindros, "Proof-of-stake sidechains," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 139–156.

[14] L. Yin, J. Xu, and Q. Tang, "Sidechains with fast cross-chain transfers," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 3925–3940, 2021.

[15] J. Kwon and E. Buchman, "Cosmos whitepaper," *A Netw. Distrib. Ledgers*, 2019.

[16] FISCO BCOS. [Online]. Available: http://www.fisco-bcos.org/

[17] S. Ye, X. Wang, C. Xu *et al.*, "Bitxhub: side-relay chain based heterogeneous blockchain interoperable platform," *Comput Sci*, vol. 47, no. 06, pp. 300–308, 2020.

[18] L. Yin, J. Xu, K. Liang, and Z. Zhang, "Sidechains with optimally succinct proof," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 4, pp. 3375–3389, 2024.

[19] A. Garoffolo, D. Kaidalov, and R. Oliynykov, "Zendoo: A zk-snark verifiable cross-chain transfer protocol enabling decoupled and decentralized sidechains," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 1257–1262.

[20] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song, "zkbridge: Trustless cross-chain bridges made practical," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3003–3017.

[21] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[22] ChainMaker. [Online]. Available: https://chainmaker.org.cn/

[23] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2018, pp. 66–98.

[24] M. Muzammal, Q. Qu, and B. Nasrulin, "Renovating blockchain with distributed databases: An open source system," *Future generation computer systems*, vol. 90, pp. 105–117, 2019.

[25] J. Lu, J. Shen, P. Vijayakumar, and B. B. Gupta, "Blockchain-based secure data storage protocol for sensors in the industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5422–5431, 2021.

[26] L. Zhou, A. Fu, G. Yang, Y. Gao, S. Yu, and R. H. Deng, "Fair Cloud Auditing Based on Blockchain for Resource-Constrained IoT Devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 5, pp. 4325–4342, 2023.

[27] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *International conference on the theory and application of cryptology and information security*. Springer, 2001, pp. 514–532.

[28] Y. Zhao, "Practical aggregate signature from general elliptic curves, and applications to blockchain," in *ACM asia conference on computer and communications security*, 2019, pp. 529–538.

[29] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Annual international cryptology conference*. Springer, 2018, pp. 757–788.

[30] H. Huang, X. Peng, Y. Lin, M. Xu, G. Ye, Z. Zheng, and S. Guo, "Scheduling most valuable committees for the sharded blockchain," *IEEE/ACM Transactions on Networking*, vol. 31, pp. 3284–3299, 2023.

[31] M. Zhai, Q. Wu, Y. Liu, B. Qin, X. Dai, Q. Gao, and W. Susilo, "Secret multiple leaders & committee election with application to sharding blockchain," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 5060–5074, 2024.

[32] N. Boehmer, M. Brill, A. Cevallos, J. Gehrlein, L. Sánchez-Fernández, and U. Schmidt-Kraepelin, "Approval-based committee voting in practice: a case study of (over-) representation in the polkadot blockchain," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 9, 2024, pp. 9519–9527.

[33] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186.

[34] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques*, 2003, pp. 416–432.

[35] J. Nick, T. Ruffing, and Y. Seurin, "Musig2: Simple two-round schnorr multi-signatures," in *Annual International Cryptology Conference*. Springer, 2021, pp. 189–221.

[36] B. Wesolowski, "Efficient verifiable delay functions," *Journal of Cryptology*, vol. 33, pp. 2113–2147, 2020.

[37] N. Ephraim, C. Freitag, I. Komargodski, and R. Pass, "Continuous verifiable delay functions," in *EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2020, pp. 125–154.

[38] A. Boldyreva, "Efficient threshold signatures, multisignature and blind signature schemes based on the gap-diffie-hellman-group signature scheme," in *PKC 2003*, 2002, pp. 31–46.

[39] A. Warner and T. F. Keefe, "Version pool management in a multilevel secure multiversion transaction manager," in *Proceedings 1995 IEEE Symposium on Security and Privacy*, 1995, pp. 169–182.

[40] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," Ph.D. dissertation, University of Guelph, 2016.

[41] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt, "Sok: Communication across distributed ledgers," in *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2021, pp. 3–36.

[42] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," *Journal of the ACM*, 2015.

[43] WeCross. [Online]. Available: https://github.com/WeBankBlockchain/WeCross

[44] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa *et al.*, "ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 485–496, 2021.