InterTrust: Towards an Efficient Blockchain Interoperability Architecture with Trusted Services

†‡Gang Wang, †Mark Nixon
†Emerson Automation Solutions, Round Rock, USA
‡University of Connecticut, Storrs, USA
Email: email.gang.wang@gmail.com

***Abstract*—Blockchain as a potentially disruptive technology can advance many different fields, e.g., cryptocurrencies, supply chains, and the industrial Internet of Things. The next-generation blockchain ecosystem is expected to consist of various homogeneous and heterogeneous distributed ledgers. These ledger systems will inevitably require a certain level of proper cooperation of multiple blockchains to enrich advanced functionalities and enhance interoperable capabilities for future applications. The interoperability among blockchains will revolutionize current blockchain design principles, like the emergence of the Internet. However, the development of cross-blockchain applications involves much complexity regarding the variety of underlying cross-blockchain communication. With that regard, we propose an efficient, interoperable blockchain architecture, *InterTrust*, to support interoperability and trustworthiness among arbitrary blockchain systems (including homogeneous and heterogeneous blockchains). It consists of an atomic cross-chain communication protocol, which can be considered an agnostic protocol to integrate existing blockchain systems smoothly. *InterTrust* is powered by two innovative techniques: threshold signature scheme and trusted hardware. The threshold signature scheme guarantees consistency and verifiability in the target blockchain systems, and the trusted hardware guarantees trusted services among distinct blockchain systems. Combining these two techniques provides an efficient cross-chain communication protocol to facilitate atomic swaps and interoperable operations between different blockchain systems. Our interoperable architecture is robust to support arbitrary blockchain systems. We also present the security analysis on the scenarios of integrating our protocol into Byzantine fault tolerance based blockchain systems.**

Keywords*—*Blockchain Interoperability, Trusted Services, Cross-chain Communication, Atomic Swaps.

I. Introduction

Blockchain has become a key enabler for implementing and advancing distributed ledgers. It allows a group of participating nodes (or parties) that do not trust each other to provide trustworthy and immutable services. Distributed ledgers were initially used as tamper-evident logs to record data. However, independent parties typically maintain them without a central authority. The blockchain became popular because of its success in cryptocurrencies, e.g., Bitcoin [1]. Emerging blockchain technological advances and applications have earned tremendous attention from both industrial and academic domains, promising to change all aspects of digital businesses of the industry [2]. Essentially, blockchain is a kind of Decentralized Ledger Technology (DLT) that heavily utilizes cryptographic primitives to secure host applications, store data, and exchange information [3]. It will profoundly impact and influence existing Internet infrastructures and promote the development of the decentralized Internet.

Naturally, different use cases have different requirements and thus demand different capabilities of blockchains. Some information cannot be exchanged freely and directly between two distinct blockchains due to various protocols and technologies. The development of independent and incompatible blockchain technologies causes significant fragmentation of blockchain research since users and developers have to choose from a set of blockchains for their use cases. As a result, this leads to a certain level of incompatibility and isolation in today's blockchain ecosystem, and we see many distinct blockchains. It is desirable to achieve some interoperable blockchain schemes to freely exchange information in future infrastructures, e.g., as a global value-exchange network [4], without involving significant changes on their underlying blockchain infrastructures. Originally, interoperability is the ability of two or more components/systems to cooperate despite differences in language, interface, and execution platform [5]. While in the context of blockchain, interoperability means connecting multiple blockchains to access information and act on that information by changing their own state or the state of another blockchain. Optimally, this would be achieved without compromising the blockchain's premise in decentralization and trustworthiness [6].

Blockchain interoperability requires that assets be moved from one blockchain to another, or users have the ability to access information from one blockchain inside another without any additional efforts from a third party. Currently, the notion of blockchain interoperability is still in a conceptual stage. It has limited practice since successful blockchain interoperability requires at least two blockchains to freely exchange information in a way that the information exchanges via the Internet. It not only needs to consider public blockchains but also needs to cooperate with private and consortium blockchains. However, due to the security and privacy issues, private and consortium blockchains may not be willing to share their information with the public [7]. Therefore, it is highly desirable to provide a generic framework to cover most existing blockchain systems without compromising the original properties of a blockchain system.

To achieve blockchain interoperability, we require the development of cross-blockchain protocols to accomplish communication between distinct blockchains. According to how information is exchanged, there exist different kinds of classifications on cross-blockchain protocols, e.g., the classifications in works [8] [9] [10]. We can consider a cross-blockchain communication protocol as an exchange protocol with some specific features, e.g., atomicity and consistency. Typically, an exchange protocol requires an atomic swap of two (or more) digital assets, e.g., x on chain X and y on chain Y. This kind of protocol, in practice, consists of a two-phase commit (2PC) mechanism, where the involved parties can explicitly abort the exchange in case of failure to reach an agreement. For example, hashed time-locks contracts (HTLCs) belong to this category. However, the work [11] shows the impossibility of a cross-blockchain communication protocol without a trusted third party for the fair exchange problem [12]. Thus, we need to establish some trust models between involved blockchains. These trust models should have the ability to provide trustworthiness to a blockchain (e.g., virtually as a component in the decentralized Internet).

Preprint online version of paper published in the proceeding of 2021 IEEE International Conference on Blockchain (Blockchain'21).

Among many trust models, trusted hardware is a good candidate to provide trustworthiness between blockchains, even in a decentralized network. Trusted hardware, e.g., Intel SGX [13]) and ARM TrustZone [14], can create a secure (tamper-proof and confidential) execution environment (aka trusted execution environment (TEE)), which further enables secure and trust communication among the involved parties. For example, in Intel SGX, the TEE technology allows applications to be executed within a protected environment called an enclave, which ensures confidentiality and software integrity. An enclave is essentially a CPU-protected address space that is accessible by no other entities but the enclave owner. In addition, it provides a level of attestation. For example, a user can verify if a specific TEE is correctly instantiated and running at a remote host via a remote attestation protocol. Our atomic swap protocol utilizes trusted hardware to facilitate trusted communication and services across distinct blockchains, such that transactions that exchange assets are in an *all-or-nothing* manner.

The exchanged information or assets also require an atomic swapping process, which can guarantee integrity and consistency among different blockchain networks. Roughly speaking, in an atomic swap, two parties trade their assets from different blockchains with each other. Both parties are required to have an account or an address on the other blockchain, and the trades must happen simultaneously on both blockchains. It needs to guarantee that either both transfers happen or neither of them happens. This property is called "atomicity", whose swapping process is indivisible [15] [16]. Each blockchain system can be considered as an Autonomous System (AS), which has its consensus protocol to guarantee consistency among all honest AS members. The communication between ASs (aka. cross-AS communication) must also be atomic to guarantee the overall consistency among all ASs. In this paper, we propose an efficient atomic cross-AS protocol, *InterTrust*, with the help of a threshold signature scheme and trusted hardware, to achieve interoperability and trusted services among blockchain systems. Our atomic cross-AS protocol assembles the two-phase commit (2PC) mechanism and the hash-locking scheme to achieve the atomicity of exchanged information. We leverage trusted hardware as interfaces between distinct ASs to provide trustworthiness. Instead of using the hash as the lock-in assets in the hash-locking scheme, we utilize a threshold signature scheme to prove the achieved agreement on the transaction between distinct ASs. And the threshold signature provided by one AS can be considered as an attestation on the state update of the other AS. By integrating both trusted hardware and threshold signature to our atomic swapping process, our scheme can achieve consistency and trustworthiness for cross-AS communication. We should mention that, in InterTrust, the trusted hardware is not a kind of trusted third party, and there is no centralized authority or entity to control this trusted hardware. Each trusted hardware still works in a decentralized manner. In the following description, we interchangeably use the terms "cross-AS" and "cross-chain", as both represent the same scenario on cross-blockchain communication.

The rest of the paper is organized as follows. Section II introduces some preliminary information on blockchain interoperability, threshold signature scheme, and trusted hardware. Section III discusses the system and threat models that our protocol is based on. Section IV gives the detailed cross-AS communication protocol. Section V provides some security analysis of the protocol when combined with BFT consensus protocols. Section VI provides an evaluation of the proposed scheme. Section VII summarizes some related works on blockchain interoperability, and section VIII concludes this paper.

II. Preliminaries

This section introduces some basic information on blockchain interoperability, followed by the concept of threshold signature and trusted hardware, both of which are key primitives of our InterTrust scheme to guarantee both interoperability and trustworthiness of a cross-AS communication.

A. Blockchain Interoperability

Cross-chain communication is one of the primary design considerations when designing an interoperable blockchain system. Currently, each blockchain system operates as an *isolated* information island, where it is difficult to obtain external data, and each blockchain executes transactions on its own ledger [17]. Blockchain interoperability requires that disparate blockchain systems can communicate with each other, with the ability to share, access, and exchange information across blockchain networks without the help of some intermediaries (e.g., a centralized authority). Thus, blockchain interoperability requires that assets can be moved from one blockchain platform to another or the users have the ability to access information from one blockchain inside another without resorting to any extra efforts from some third parties. The information exchanged also requires some atomic swapping process that is used to guarantee integrity and consistency among different blockchain networks. The term "atomic" is borrowed from database systems, where atomicity or an atomic transaction is limited to a set of binary outputs (e.g., either 0 or 1) [18]. We can adapt the concept of atomic swap to a multi-blockchain scenario, which is referred to as an atomic cross-chain swapping process.

In general, atomic swap is considered a basic operation to achieve interoperability among multiple blockchain systems. And, there exist different schemes to achieve blockchain interoperability, e.g., sidechain, notary scheme, and hash-locking [8] [9] [10]. The initial goal of a sidechain scheme is to extend functionalities of interoperable blockchain networks, where data can be sent and received between interconnected blockchain networks. Furthermore, this kind of design philosophy helps the security of the whole system. For instance, by isolating sidechains from the main chain, in case of cryptographic breaks (or maliciously designed sidechains), the damage is entirely confined to the sidechain itself. It will not affect the records on the mainchain. Notary schemes utilize a third trusted entity as an intermediary between blockchains. Thus, the notary's role is to verify the correctness and integrity of exchanged information to guarantee consistency among blockchains. One significant advantage of notary schemes is that it is simple, as no additional change is required to the underlying blockchain systems. Hash-locking is another technique to exchange assets without involving a trusted third party [19]. Roughly speaking, the hash-locking technique utilizes a hash time-locked system which puts a time lock on transactions so that both obligations are fully met. Otherwise, no transaction will occur in the involved blockchain systems, which is similar to the concept of the atomic transaction [20] [9]. Thus, Hash-locking achieves the atomic assets exchange through a time difference and the hidden hash value. Our atomic swap follows the conceptual procedures of a hash-locking scheme (e.g., with *lock* and *unlock* operations) without requiring a trusted third party.

B. Threshold Signature

The transactions exchanged between blockchains require authenticity, non-repudiation, and integrity, and in general, a digital signature

scheme can offer these features. To guarantee consistency within a blockchain system (e.g., consensus on a block), a group of participants are required to sign the transactions, and a threshold signature scheme is an ideal option. A threshold signature scheme can guarantee that for a given threshold parameter k ($1 \leq k \leq n$), any k signers from the total of n signers can be collaboratively used to generate (or *recover* during decryption) a valid signature for any given message. However, there is no way to do so when the number of signers is less than the threshold k. Thus, in a threshold signature, each signer is required to hold its distinct private signing key, and this signing key is used to produce a valid signature share [21]. A (k, n) threshold digital signature scheme allows a set of signers to generate a digital signature as a single logical entity despite $(k-1)$ faulty results (e.g., provided by malicious signers). By dividing a private key into n shares, each one is owned by a signer. Each signer uses its key share to generate a *partial* signature on a message m and sends its partial signature to a combiner signer, which combines the partial signatures into a threshold signature on m [22].

Depending on the underlying crypto-primitives, various threshold signature schemes exist, e.g., threshold RAS signature and threshold BLS signature. Following the basic digital signature scheme, all these schemes contain at least three *functional* algorithms: *KeyGen* (a key generation algorithm), *Sign* (a signing algorithm), and *Ver* (a verification algorithm). According to the underlying crypto-primitives, these functional algorithms may be different. By leveraging these three algorithms, a threshold signature scheme can guarantee the integrity of messages. Due to the lack of trust between blockchains' participants, in our scheme, we utilize a threshold signature scheme working with the underlying consensus protocol to guarantee the properties of authenticity, integrity, and consistency on exchanged messages. Besides, a valid threshold signature can be considered as a commitment scheme that allows one or more participants to commit to a chosen value while keeping it hidden from others [23]. This commitment scheme means we can use a threshold signature between blockchains to attest to the integrity and consistency of one blockchain. Different from the usages of threshold signatures in consensus protocols (e.g., BFT-based schemes on SMChain [24], HotStuff [25], and SBFT [26]), the threshold signature scheme in our InterTrust is a protocol-agnostic scheme, which provides a verifiable service as a commitment.

C. Trusted Hardware

Trusted computing has been widely recognized as a useful and essential extension in traditional security mechanisms [27]. A third trusted party is vulnerable to many weak points [28], and trust is one of the major concerns in large-scale open distributed and decentralized systems. Trusted Platform Module (TPM) [29], such as Intel Software Guard Extension (*Intel SGX*) [13]), is one of the most popular types of trusted hardware. Intel SGX is an example, and this technology allows applications to be executed within a protected environment called an enclave, ensuring confidentiality and software integrity. The enclave enables an isolated, tamper-free environment that can attest that an output represents the result of such execution and allows remote users to ensure that the attestation is correct. The wide adoption of TPM in the industry offers many security features, e.g., prevention of rollback, protection of data at rest, and early launch of anti-malware [30]. Equipped with encryption keys whose private parts never leave a TPM hardware chip, this reduces the possibility that those keys may be compromised. TPM provides a small set of primitives that can offer a high level of security assurance in many ways, e.g., by offering machine-based solid identities and preventing unauthorized attacks [30]. Both the industry and academia communities have adopted these primitives as building blocks in a variety of security systems. TPM hardware, together with its supporting software and firmware, offers a root-of-trust platform. This trust can be extended to other platform components by building a chain of trust, where each component extends its trust to the next one. And the trusted hardware can also be directly used in blockchain to provide some trusted services.

We may argue that trusted hardware can be considered as a trusted third party. However, this kind of trusted third party is not controlled by any entity. This makes the trusted hardware schemes more secure than platforms that rely on centralized servers. We should note that the trusted hardware in our scheme is only used to provide trusted services among ASs, and the trustworthiness within one AS is still provided by its underlying consensus protocol.

In general, cross-blockchain atomicity is critical to the correctness and robustness of multiple interoperable blockchain systems. In InterTrust, we combine the threshold signature scheme and trusted hardware to provide a *trusted relay* between different blockchains. By deploying a threshold signature scheme on trusted hardware, the messages exchanged between blockchain interfaces are trusted. Furthermore, the recipient can use the threshold signature scheme to verify the validity of received messages.

III. Models

This section presents the system and threat models that our interoperable blockchain architecture is based on.

A. System Model

Our system model considers the concept of an Autonomous System (AS), and each AS is responsible for one blockchain system. For example, each AS has its own network topology, consensus mechanism, rewarding mechanism, and the activities performed on each AS can be considered as internal operations. We assume that each participant in an AS has a public/private key pair (pk, sk), and the public key pk is used to define its identity. The senders' private key authenticates all messages sent over the network. Although we do not have many restrictions on operations within each AS, we put prerequisites on the communication between ASs to achieve required features, e.g., atomicity, integrity, and trustworthiness. The participants within each AS may have different capabilities, e.g., computational capability and hardware equipment. We assume each AS must equip with *at least* one participating node/gateway with trusted hardware (aka. trusted gateway or trusted relay), and the cross-AS communication must happen via these trusted gateways. We can roughly consider this trusted gateway as an edge gateway to facilitate the trusted communication and trusted services between autonomous systems. We also consider that the communication model between ASs is based on some asynchronous network model where messages are eventually delivered.

Our interoperable blockchain architecture assumes that each participating AS is either a private blockchain or a consortium blockchain, both of which require some permissions to participate in the construction of a blockchain. Also, we assume that each AS must have an eventual consensus protocol to get a block finalized in its blockchain. However, the eventual consensus protocols may be different for different ASs, and the usage and discussion on these consensus protocols are out of the scope of this paper.

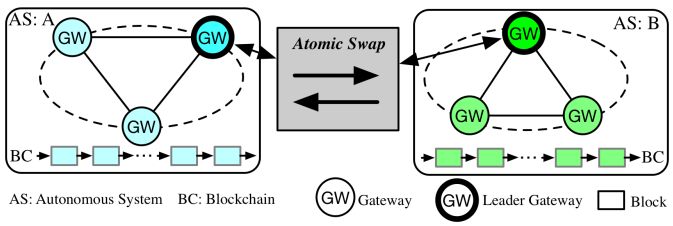

Fig. 1. Overall architecture of interoperable blockchains. The bold circled gateways equipped with trusted hardware.

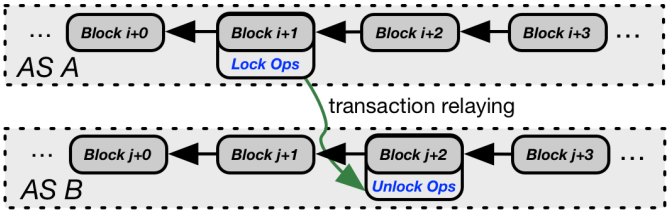

Fig. 2. A visual representation of message passing across different blockchains.

B. Threat Model

In general, our interoperable architecture achieves its trustworthiness and interoperability by relying on a trusted execution environment (TEE) and atomic swapping technology with the help of threshold signature. We assume two types of nodes, clients and participants. The participants actively participate in the construction of blockchain. The clients do not actively engage in the construction of blockchain (e.g., via consensus protocol) and only send messages to the connected participants. We assume that the clients are honest all the time (e.g., whose messages are trustable or whose messages can be verified by participants), but a participant can be either honest or malicious (e.g., Byzantine). An honest participant faithfully follows its consensus protocol. In contrast, a malicious participant may behave arbitrarily, e.g., it may refuse to participate in a consensus protocol, or it may collude with others to carry out some attacks. To ease the understating, we assume the malicious participants behave like a Byzantine. Even under Byzantine, however, we do assume that the codes that run inside the TEE enclave can neither be observed nor tampered with.

We assume trusted gateways in an AS know the basic information of their AS, e.g., the number of participants in AS, the public keys of AS members, and the adopted consensus protocol. And, these pieces of information are critical to the adoption of the threshold signature scheme. For example, if an AS utilizes Byzantine Fault-tolerant (BFT) protocol with an assumption of $n \geq 3f+1$ (where f is the maximal number of Byzantine participants and n is the number of total participants in its AS), then the threshold k in a (k, n) threshold scheme must be at least $f+1$ to get a correct result. We assume that the trusted gateway in an AS has the ability to dynamically adjust this threshold parameter to fulfill the requirement of the underlying consensus protocol. Also, the communication between ASs is via trusted gateways, and we assume the communication between trusted gateways is reliable. However, we do not fix a trusted gateway pair between any ASs. There exist multiple choices on the connections between ASs, and these connections can be dynamically adjusted.

Fig. 1 shows an overall architecture of interoperable blockchains, which consists of two ASs. Each AS is an independent blockchain system, and all cross-AS communication is through the trusted gateways (the bold ones) atomically via our atomic cross-AS swap protocol. Also, the trusted gateways function as interfaces to facilitate trusted communication and trusted services across distinct ASs. We will discuss the proposed atomic cross-AS swap protocol in Section IV.

IV. Interoperable Architecture

This section presents the detailed architecture design of InterTrust and its cross-AS communication protocol by leveraging the primitives of the threshold signature scheme and trusted hardware. From a high-level perspective, the threshold signature is mainly used to guarantee the integrity and atomicity of information within an AS and for the cross-AS communication protocol. In contrast, the trusted hardware is used primarily to ensure trusted services among ASs.

In one AS, a transaction may involve other participants in a different AS. Typically, it is a challenging task to achieve consistency since state-updating of the involved participants occurs independently in different ASs. Thus, cross-AS atomicity is one of the key features to achieve an interoperable blockchain architecture. We propose an efficient atomic cross-AS swap protocol with the help of a threshold signature to ensure the atomicity of transactions across blockchains. Our atomicity protocol allows the interleaving of transactions in an asynchronous and lock-free manner to keep ASs concurrent and fully utilized. The proposed atomicity protocol is with the help of trusted gateways (e.g., equipping with trusted hardware) and the threshold signature scheme to decouple a cross-AS transaction into a multi-round communication, and each round communication is trustable. This section first describes the overall of our interoperable architecture and then presents the details of the atomic cross-AS swap protocol.

A. System Design

Before discussing the detailed interoperable communication protocol, we first exemplify a high-level scenario on how a cross-AS protocol processes a transaction, which can be easily extended to other scenarios. Without loss of generality, we consider a cross-AS transaction. The transaction instance involves two participating nodes from different ASs, e.g., *AS A* and *AS B*, as shown in Fig. 1. A visual representation of message passing is provided in Fig. 2. Typically, the message passing of a cross-AS transaction involves two *opposite* operations, e.g., *lock* and *unlock* operations in Fig. 2. In this case, the lock operation, which only involves the state change in AS A, is handled by the participating nodes of AS A. If the specified conditions of the cross-AS transaction are met, this transaction will only be locked in a block (e.g., block $i+1$ in Fig. 2) of AS A. Once the lock operation is done, AS A will compose a *relay* transaction carrying proof of the locking operation, and this relay transaction will be forwarded to AS B for further processing to complete this cross-AS transaction. After verifying the validity of the relay transaction (with the attached proof), the participants in AS B will execute the unlock operation to claim the cross-AS transaction. Once the unlock transaction is executed and appended to the blockchain (e.g., block $j+1$ in Fig. 2) of AS B, this cross-AS transaction completes. Section IV-B will detail the operational process and the handling of unsuccessful execution. We note that the consensus protocol of each AS will guarantee consistency within its blockchain.

Atomicity across ASs requires to capture the *all-or-nothing* settlement [31]. Given a transactions tx_1 of AS_A (short for *AS A*) and a

transaction tx_2 for AS_B (short for *AS B*), an *all-or-nothing* settlement is protocol that guarantees that: 1) Both transactions get confirmed at their representative ASs; or 2) Neither transaction will get confirmed at its AS. A generic cross-AS communication protocol, which is used to achieve atomicity, typically consists of several sequential steps [11]. We briefly discuss a generic four-step protocol to help understand our atomic cross-AS protocol.

1) Setup Phase: Before two blockchains communicate, each blockchain needs to know some parameters of the other blockchain, which is similar to a negotiation process in an Internet transmission protocol. This step is called a *setup*. Typically, the setup process occurs out-of-band between the involved blockchains (or between the agents of blockchains), and thus we can omit this step.

2) Commit Phase: Upon successful setup, the cross-AS protocol goes to a *commit* phase, alternatively called *(pre-)commit* phase, in which the transaction is not finalized. The commit phase typically involves the locking and unlocking of assets on blockchains, determined by the request of the consensus protocol. This phase requires some techniques, as coordinators, to perform the locking and unlocking operations. For example, we can use some hash locking schemes or threshold signature schemes to facilitate these operations.

3) Verification Phase: The verification phase is used to verify the commitment of the execution. For example, if we consider two blockchains, source chain A and target chain B, the correctness of the commitment on A should be verified by the participants of B. Depending on the content to be verified, the actual verification process may be different, e.g., the verification on the consensus agreement on a stage or a specific state transition of a transaction. Upon successful verification, a publicly verifiable commitment is published and executed on-chain B.

4) Abort Phase: The abort phase is optional and is encountered typically in exchange protocols. If the verification fails or the execution on chain B fails, an abort phase will be performed. Based on the adopted commit protocol in *commit* phase, it may have different operations. For instance, if the commit phase adopts a locking scheme, then the locked assets will be released.

In general, our cross-AS protocol follows the generic communication patterns mentioned above, with our customized atomic commitment scheme with the help of the threshold signature scheme and trusted hardware.

One key feature provided by our InterTrust is the trusted services between ASs. Each AS is an autonomous eco-system, which has provided some interfaces to external ASs (e.g., via gateways or edge gateways as the agents or representatives of its AS). We require that such interfaces must be robust enough to provide trusted services to the outside ASs. This requirement is much realistic and practical for almost all application scenarios. For example, a corporation has its own private blockchain and provides outsourcing services to its customers. The interface for outsourcing services must be robust enough to the outside customers. We can consider such a kind of interface as a representative of its blockchain system to interoperate with outside blockchains. In our InterTrust, we require that such interface must equip with some trusted hardware to provide trusted service to other ASs, and the other recipient ASs also are required to equip with trusted hardware. In this way, they can establish a trusted interface between ASs, and the corresponding services between them are trustworthy. As shown in Fig. 1, the atomic swap happens between trusted gateways, and their communication and services are trusted. We need to mention that, for any gateway, as long as it equips with trusted hardware, it can serve as this kind of interface to communicate with other ASs, and we do not specify a designated gateway as an interface to all other ASs. For example, we can utilize some randomized algorithms to assign a trusted gateway for one round of communication (i.e., one round of atomic swap), and for the next round, it may assign a different trusted gateway to provide trusted services. By randomizing communication connections, it can decrease the chance of some potential attacks on some target gateways, and the communication between them is decentralized. Or, we can parallel the trusted communications between trusted gateways. But, the cross-AS communication must be performed between trusted gateways so that it can guarantee trusted services. The algorithm or scheme to assign these trusted gateways (as the interfaces of AS) is out of the scope of this paper.

B. Atomic Cross-AS Protocol

Our atomic cross-AS protocol is with the help of a threshold signature scheme and trusted hardware to achieve trustable atomicity between ASs. Considering heterogeneity among the participants of an AS, we require each honest participant *at least* has the ability to perform the basic verification and signing operations of a threshold signature scheme, and the trusted gateways must have the ability to perform the full functionalities of a threshold signature scheme (e.g., generating threshold share secrets, verifying signed shares, and combining operations on signed shares). This would help the verification of the messages from other ASs. For example, if an incoming message does not pass the verification of trusted gateways, this message will not be broadcast to its AS members. This will potentially reduce the number of broadcast messages. There are typically two kinds of transactions: intra-AS transactions and inter-AS transactions (aka, cross-AS transactions). The communication of intra-AS transactions is restricted within one AS, and for blockchain interoperability, we do not care about this kind of transaction. In the following description, we only discuss the cross-AS transactions and ignore the cases on intra-AS transactions. A simple Distributed Hash Table (DHT) can be employed to distinguish if a transaction is a cross-AS transaction or an intra-AS transaction. We consider a cross-AS transaction tx involves two independent ASs, a source AS X running a process ρ on ledger L_X (e.g., a *withdraw* operation), a destination AS Y running a process θ on ledger L_Y (e.g., a *deposit* operation). Processes ρ and θ work on these own eventual consensus protocols, respectively.

When a client submits a cross-AS transaction tx to the process ρ of AS X, this transaction tx will be broadcast to all participants of AS X. This transaction also is relayed to trusted gateways of AS X. There may exist different numbers of trusted gateways. Furthermore, the topologies on these trusted gateways may be different, which means there may exist many connections between two AS via these trusted gateways. Here we assume only one pair of selected trusted gateways (e.g., gateways G_X and G_Y) are interfaces to perform the atomic cross-AS communication between AS X and AS Y. The trusted gateways of the involved ASs will start a negotiation process to exchange some parameters of their representative ASs, which is a setup phase with an intention to start a cross-AS communication. Considering the privacy-preserving on the blockchain of each AS, we do not require them to exchange too much information, and we only require that each AS provides its publicly available information, e.g., the number of participants in its AS and the corresponding public keys of each participant. And that public information is enough to perform

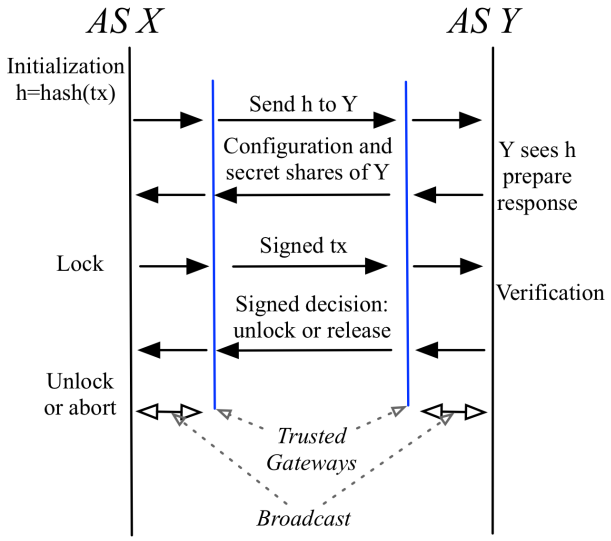

Fig. 3. Abstract of atomic cross-AS protocol. The blue boundaries represent trusted gateways to provide trusted service between ASs, and the communication between trusted gateways and its AS members are broadcast.

a threshold signature scheme. Practically, our efficient atomic cross-AS communication protocol consists of four phases as follows. And Fig. 3 shows an abstract atomic cross-AS communication protocol, in which the blue boundaries represent trusted gateways to provide trusted services between ASs.

Setup: The trusted gateway G_X of AS X initializes the communication by sending an initialization message to the target gateway G_Y of AS Y to request the configuration information of AS Y. The initialization information contains the hash of tx and the configuration information of AS X (e.g., the number of participants n in X and their corresponding public keys). Upon receiving this initialization message from X and being willing to access this initialization message, the trusted gateway of AS Y will run a $(k, n)-$threshold signature scheme to generate n shares of secrets and each participant in X will get one secret share. Note that the generation algorithm in the threshold signature scheme requires a random seed. We can use the hash of tx as this seed. After the secret shares are generated, the trusted gateway G_Y will compose a response message for this cross-AS request of G_X, whose response message includes the basic configuration of AS Y, the hash of tx, and the generated secret shares. Then, G_Y will send this response message back to the trusted gateway G_X.

Lock: Upon receiving the response message from G_Y, G_X will distribute this response message to each replica, and each replica will get a secret share. According to the hash of tx (a kind of token for the processed message for this round) in the response message, a replica will perform a *lock* operation on this transaction tx if this transaction tx is valid. Then, the replica first signs the locked transaction using its secret share from trusted gateway G_Y, and then signs the signed transaction using its own private key, and finally, sends its signed transaction (including two signatures) to the trusted gateway G_X. Meanwhile (during replicas sign the transaction), the trusted gateway G_X also performs the key generation algorithm of $(k', m)-$threshold signature scheme to generate m shares of secrets, where m is the number of participants in AS Y (getting this parameter from the response message of G_Y during setup phase). If G_X receives enough validly signed transactions from AS X, it will first perform a verification to verify if he can recover the signed tx to the original transaction tx. If so, the collected signatures together with the generated m secret shares for AS Y will be forwarded to G_Y. Otherwise, it will withhold until receiving enough valid signatures.

Verification: Upon receiving these signatures and secret shares from G_X, G_Y will distribute these signatures to each replica, and each replica will perform verification on these signatures and recover the transaction tx. After the transaction tx is obtained, the replica will perform verification to see if the transaction is valid in AS Y. If the execution of the transaction tx is valid, then it will include an *unlock* signal and sign it (and possibly include the exchanged information or assets) by its secret share from G_X and sign the signed signature with its own private key. If the execution of the transaction tx is not valid (e.g., conflicted with previous transactions or failed to pass some smart contracts), it will include a *release* signal to abort this transaction, and this *release* signal also will be signed by both secret share from G_X and its own private key. These signed messages will be transmitted to G_Y for verification, and similarly, G_Y will first perform the verification and then release them to trusted gateway G_X if the G_Y got enough valid messages from its own AS.

Unlock/Release: According to the received signal (either *unlock* or *release*) from G_Y, G_X will first verify and then distribute that received message. If a replica in AS X recovers it into a *unlock* signal, the transaction tx will be unlocked and executed as a valid transaction and finally attached into blockchain L_X via consensus protocol. If a replica recovers it into a *release* signal, it indicates the transaction tx is not valid and is aborted at the target AS Y, and this transaction tx will not be included in the blockchain. As we assume the communication between trusted gateways is reliable, if the replica does not receive any message about transaction tx within a predefined time interval (e.g., the time of eventual consensus at target AS plus the transmission time), it considers the transaction to be aborted. And, the AS X will notify the execution result of this cross-AS transaction tx to its client.

A transaction involving *lock* and *unlock* operations should be atomic to ensure the correctness of the ledgers among different ASs. Our atomic cross-AS protocol assembles a two-phase commit (2PC) mechanism and hash-locking scheme to achieve its atomicity with the known lock/unlock operations. In our protocol, we allow the operations in the target AS to confirm first, interleaving with other transactions, then the corresponding operations to be settled later. What is achieved is that once the verification phase in the target AS is confirmed, the transaction in the source AS will be executed eventually. We call such atomicity eventual cross-AS atomicity. We optimistically assume the transaction in the source AS will be eventually picked as long as there exists an eventual consensus protocol in that AS. Theoretically, there may exist bad-behaved participants. As long as the transaction is pre-verified and gets agreed by the majority of participants (the number of "majority" depending on the consensus protocol used), the transaction will finally be appended to its ledger. Otherwise, these transactions will be unlocked.

With the help of a threshold signature scheme and trusted hardware, we can achieve an efficient atomic cross-AS communication protocol and further advance an interoperable blockchain architecture with trusted services.

V. Security Analysis

Our InterTrust scheme focuses on architecture design to facilitate the interoperability among blockchain by utilizing both threshold signature and trusted hardware. In general, each AS can have its own consensus protocol to get an agreement as long as (a) it supports the operations of threshold signature (e.g., signing and verifying), and (b) there exist trusted hardware equipped gateways to transmit messages among distinct ASs. Even though we do not put any constraints on the underlying consensus protocol within each AS, we do prefer each AS utilizing a BFT-based consensus protocol, taking advantage of instant finality and quick responsiveness. Another advantage of utilizing the BFT protocol in each AS is that we can directly apply the underlying threshold signature scheme to scale its consensus process and advance the responsiveness. There exist several works, e.g., SMChain [24] and SBFT [26], on integrating threshold signature with BFT protocol to achieve linear scalability.

We use a leader-based BFT protocol, as an instance, together with our threshold signature scheme to conduct security analysis with regard to its safety and liveness. Following the terminologies in BFT protocols, we consider each participating node as an active replica (either honest or Byzantine) within an AS. *Safety* is the property that if any two honest replicas commit on a decision block for a given sequence number, the two blocks must be the same. *liveness* is to ensure that the consensus protocol makes progress in the current view and moves to a new view, which means clients (aka. other AS's trusted gateways) eventually receive replies to their requests. Besides, we assume $n >= 3f + 2 = (3f + 1) + 1$, where n is the number of replicas, f is the number of Byzantine replicas, and the extra one replica is the trusted gateway. In addition, at most f replicas are passive replicas (e.g., no response) so that the total participating replicas are at least $2f + 2$. In following discussions, $h = H(r||v)$, where r is a request (*req*) from a client and v is the current view number, and $\sigma_{TC}(h)$ is a signature operation on h. Also, we set the threshold to recover a message is $f+1$ in a $3f+1$ system (as this does not include the trusted gateway in a consensus process). In addition, we follow a scalable BFT protocol in SBFT [26] as its consensus protocol for an AS.

A. Safety

If a non-faulty replica commits due to a signature $\sigma_{TC}(h)$ which is induced by *req*, then at least $f+1$ non-faulty replicas have received h as a hash. So at most f replicas can send a different message with $h' \neq h$. Hence, *req* or r is the unique value in the current view v that receives at least $f + 1$ messages. Because of *(f + 1, n)*-threshold signature scheme, the adversary cannot create a signature $\sigma_{TC}(h')$ where $h' \neq h$. This guarantees that, for the current view v, there do not exist honest replicas who will commit to any h' that is not h.

In case of the view-change process to a new view, e.g., $v' > v$, we can prove that the new primary $S_{p'}$ of BFT protocol with a view v' will still pick up the value req. We will use induction to show it. For the base case where the new view is $v' = v + 1$, in order to successfully switch to this new view, at least $f + 1$ non-faulty replicas must work together to participate in the process of reporting the existence of faults to the new primary $S_{p'}$ on the current view v. Otherwise, there is no view-change process due to the $f + 1$ rule in the threshold signature scheme. For the base case, the view v' becomes the only highest candidate view that can be possibly adopted. This view v' will be accepted as a new view during the view-change process. The conditions for the inductive process is almost similar, that is, every new elected primary must accept the request req provided by its previous replicas, since this indicates either there are $f+1$ non-faulty replicas report that fault happened, or there exists a signature $\sigma_{TC}(h')$ where h' consists of *req*. However, due to the setting of the threshold signature, we know that $\sigma_{TC}(h')$ will not be accepted for verification. Therefore, this inductive argument will always be held for the induction processes, and the corresponding proof is to be held for any view $v' > v$ by induction.

Considering both views v and v' in the view-change process, there must have only one primary that adopts the hash h that consists of the request req. From the above proof, as long as the new elected primary with the view v' is honest, it will adopt the request req into its decision block. Otherwise, it may start a cascading view-change process, which does not affect the security. This indicates that only a replica (e.g., a trusted gateway) receives enough signed request shares. It can have the ability to recover and accept the corresponding request.

B. Liveness

The consensus process is supposed to return the reply message to the client to notify that the client's requests have been completed with a correct view. The liveness ensures if the primary is correct, then a view is *stable*. By integrating the threshold signature scheme, we will show that we can obtain a stable state in an AS.

In a stable view, a request *req* from a client will complete. In the consensus protocol, if the primary S_p is honest, a valid message containing *req* will be sent to all the active replicas. If all the active replicas behave correctly (e.g., verifying message and signing it using its secret share), the request *req* can be completed with at least $f + 1$ replicas, and the request will get verified.

If there is a faulty replica, say S_f, it might behave arbitrarily, e.g., keeping silent, or replying with a wrong message or share. If the primary is honest, the faulty replica S_f will be detected, and it will be isolated. When the number of faulty replicas reaches a threshold number, honest replicas will initiate the view-change process, which will continue until an honest primary $S_{p'}$ is selected. Thus, the request *req* from an honest primary will be completed provided that the number of non-primary faulty replicas is no more than f.

If the view is not stable and at least $f + 1$ honest replicas request a view-change, the view will eventually be changed to a stable view. This may happen in three different scenarios.

(a). The new selected primary $S_{p'}$ behave honestly, and all other f active replicas who behave honestly have received a valid new-view message from the new primary. Thus, the system can switch to a stable view successfully.

(b). If no honest replicas receive the valid new-view message from the new primary, another view-change process will be initiated with another new primary. And this process will continue until a stable view can be reached.

(c). There is a case that the number of honest replicas who received the valid new-view message is less than $f + 1$. This case has two effects. (1) The faulty replicas can also behave honestly, e.g., temporarily following the protocol, to make the honest replicas move toward a non-stable view. (2) When detecting faulty messages, other honest replicas will send out a new *REQ-VIEW-CHANGE*, e.g., due to the timeout; however, this round of view-change process will not move forward since the number of view-change messages is less

than $f+1$. When the faulty replicas start to behave maliciously, e.g., deviating from the protocol, the honest replica will trigger a new view-change process to a new view. Eventually, the protocol will reach case (a), and a stable view will be reached.

In all three scenarios, all replicas will eventually go into a stable view, and the clients' requests can complete. If the view is not stable and less than $f+1$ honest replicas request a view change, a stable view can also be achieved. If all active replicas (either faulty or honest) follow the protocol, the request *req* can be completed successfully. Otherwise, the *req* cannot be completed successfully, e.g., due to a timeout. All honest replicas will then launch a view-change process, and the system goes to the case (a) above.

From the above analysis, by integrating the threshold signature scheme into a leader-based BFT protocol, it is easy to achieve safety and liveness. Both properties will guarantee the consistent state within an AS, and via the trusted gateway, the message will forward to other ASs for verification.

C. Trusted Services among ASs

Trusted hardware equipped gateways guarantee trusted services, and the communication between ASs must be via these trusted gateways. When a cross-AS transaction is processed by the source AS, it must first get verified by a trusted gateway before sending it to the destination AS. Suppose the processed cross-AS transaction does not pass the verification process (e.g., without getting enough signed shares from its AS members to verify the validity of messages) on the trusted gateway. In that case, the cross-AS transaction will not be forwarded to the destination AS. Instead, the trusted gateway will attach proof showing that the exchanged message has been verified and certified by its source AS if successfully getting verified. Doing so (e.g., certified by trusted hardware) can establish trusted services between the source AS and destination AS.

In general, there may establish multiple pairs of trusted connections between source AS and destination AS in a decentralized manner for one cross-AS transaction. Each trusted connection works independently. As long as one trusted connection gets established, it can successfully perform the atomic cross-AS swap protocol. We assume the destination AS has the ability to filter out the duplicated messages via the hash of transaction as the hash of transaction can be viewed as a token being processed. Only a valid cross-AS transaction can be forwarded to the destination AS when combined with a threshold signature. This means once the destination AS receives exchanged messages, these messages are trusted and validated. Thus, with the help of threshold signature and trusted hardware, our InterTrust can achieve both interoperability and trusted services among distinct ASs.

VI. Performance

In this section, we evaluate the performance using a prototype of the InterTrust scheme. InterTrust aims to achieve interoperability among multiple blockchains by using a threshold signature scheme and trusted hardware. However, due to the lack of practical blockchain systems, we only evaluate part of its performance. Specifically, we evaluate the overhead when applying a threshold signature scheme in one AS, in which it utilizes a scalable BFT-based consensus protocol. This will account for most of the overhead of our scheme. We develop a robust prototype by reusing and improving the original codebase of PBFT [32], and utilizing some libraries from BLS-based threshold signature scheme [33] [34]. This section shows some simulation results on the overhead of the applied threshold signature scheme to demonstrate its efficiency. Our simulation overhead not only includes the overhead of the threshold signature scheme, but also includes the overhead of a scalable BFT-based consensus protocol. We combine the adopted threshold signature scheme into a BFT consensus protocol, whose communication pattern is linear with the number of participating nodes.

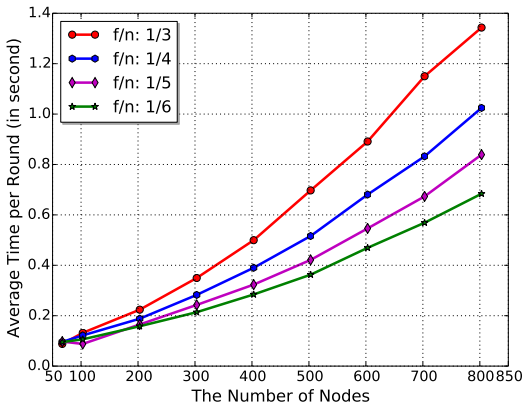

Fig. 4. The average time for each round run with various consensus group sizes and different f/n ratios.

1) Simulation Setup: Especially, our simulation utilizes a generic crypto library. Cryptographic primitives, such as SHA 256, are implemented using the Crypto++ library [35]. For threshold signature, we use Baek and Zheng's scheme [36] to encrypt a 256-bit ephemeral key, followed by ALT_BN128 Elliptic Curve in CBC mode over the actual payload, and the PBC library [33]. The threshold signature scheme requires a symmetric bilinear group; and the adopted Boneh-Lynn-Shacham (BLS) group heuristically offers 128 bits of security [37] [38]. We implement a threshold signature based consensus prototype in Python. We then perform our simulations on an Ubuntu 18.04 LTE system, equipped with Intel Core i7-3520M CPU @ 2.9GHz, 2 cores, and 8GB RAM. All the simulations are run on this system. And, we set 10ms as the network transmission delay.

2) Performance: Our simulations follow the scalable linear message communication patterns and the adopted threshold signature scheme. The implemented prototype simulates the average time for each round run based on the above setting. For each simulation, we run each round 100 times, and then get an average measurement. Note that our current prototype does not include the interaction with other blockchain systems, whose interaction in the prototype will be future work. We only simulate the overhead within one AS to get a verified message with a consensus procedure. In general, the performance of the threshold signature scheme is highly affected by the total number of participating nodes n and the total number of faulty nodes f. Thus, we take the ratio of f/n as one parameter to evaluate the overhead of our scheme. Fig. 4 shows the simulation result with various consensus group sizes and various f/n ratios. The consensus group size ranges from 64 to 800, and the f/n ratio ranges from $1/6$ to $1/3$.

From Fig. 4, we can clearly see with the increase in the number of participating nodes, the average run time for each round increases. This is mainly due to the increased communication messages in each round. For example, when the number of participating nodes is 200, the average time per round is less than 200ms; while when the number of participating nodes is 800, its time can be up to more than 1000ms. With different f/n ratios, the run time also is affected

a lot. In general, with fewer faulty nodes, the run time for each round goes down. For example, when the number of participating nodes is 800 and the f/n ratio is $1/6$, the average time is less than 700ms, while when its faulty ratio is set to $1/3$, the average time can be up to 1340ms. This is mainly caused by the increased communication messages and the operations on the threshold signature scheme. Our threshold signature scheme is set to $f + 1$ out of n scheme, where $f = \lfloor (n - 3)/3 \rfloor$. With the increase of ratio, it takes more time to perform each threshold signature related operation.

In general, for blockchain interoperability, reaching a consensus within one AS would take the most overhead of the whole interoperable process, as the overhead (e.g., transmission time) between trusted gateways can be negligible. From the above simulation, we can get when designing the scheme for a large-scale application, it is necessary to consider both the number of participating nodes and the ratio of f/n. Both parameters highly affect the performance of blockchain interoperability.

VII. Related Work

Blockchain interoperability is a promising and broad research area. Many efforts from both industry and academia have been performed on building interoperable blockchain architecture and protocols which allow cross-blockchain communication between different networks and facilitate the exchange of transactions [39].

Li et al. [40] proposed a multi-blockchain architecture model explicitly devised to meet industrial standards. It utilizes the notion of *Satellite chains* that can privately run different consensus protocols in parallel, thereby considerably boosting the scalability premises of a system. The assets can be transferred between satellite chains through a transaction process based on predefined policies. Cross-chain communication is via some special nodes that facilitate the transactions for the connected satellite chains. Typically, each satellite chain is secured by its own consensus mechanism. The model also accounts for a hands-off *regulator* that oversees the entire networks, which further enforce the policies that are deployed in the form of smart contracts and auditors. By doing so, it can monitor the transaction activities. Wang et al. [17] proposed a cross-chain communication protocol called *blockchain router* to empower blockchains to connect and communicate cross chains, which is similar to the concept of an Internet router. Each independent blockchain network is a sub-chain that connects to the "router" through a connector, and the sub-chain holds a copy of the connected blockchain data. The connector serves as an interface between the sub-chain and the router. The blockchain may take different roles. For example, some blockchains play the role of a router that analyses and transmits communication requests according to the communication protocol, dynamically maintaining a topology structure of the blockchain network.

Kan et al. [41] proposed a component-based framework for exchanging information across arbitrary blockchain systems, called interactive multiple blockchain architecture. In their architecture, a dynamic network of multichain is created for inter-blockchain communication, and an inter-blockchain connection model (or connector) is proposed for routing management and message transferring. The cross-chain communication occurs via an inter-blockchain connector of a routing management system that is used to maintain routing information of the involved blockchain networks. Each blockchain system is required to choose a router to communicate with other systems, in which the router nodes establish and exchange the network information with their neighbors. Once router information is updated, all router nodes consent to the newest routing table. Besides, this framework presents a transaction design that enables heterogeneous blockchains to communicate through standard crossing-chain transactions. HyperService [42] is a platform that delivers interoperability and programmability across heterogeneous blockchains. The platform can execute many customized and decentralized applications (dApps) by interacting with verifiable execution systems (VESes) that process and execute the requests from dApp to the blockchain network. HyperService is powered by two innovative designs: (i) a developer-facing programming framework that allows developers to build cross-chain applications in a unified programming model; and (ii) a secure blockchain-facing cryptography protocol that provably realizes those applications on blockchains. Before HyperService, a similar multichain scheme [43] is proposed as an off-the-shelf configurable platform. However, it can only work with homogeneous networks.

Weber et al. [44] proposed a scalable platform architecture for multi-tenant blockchain-based systems to ensure data integrity while maintaining data privacy and performance isolation. Each tenant has an individual permissioned blockchain to maintain its own data and smart contracts. All tenant chains are anchored into the main chain in a way that minimizes cost and load overheads. Ding et al. [45] proposed a "InterChain" framework that supports blockchain interoperability between blockchain networks. The framework consists of some independent blockchain networks (called sub-chains) and a mother blockchain (called Interchain) that connects all sub-chains together. The participating nodes in Interchain have different roles, e.g., validators to validate the transaction and gateway nodes to relay the cross-chain transactions. This work only provides a very high-level design without providing any detailed information on the involved components. Besides these efforts, some projects are aiming to create an Internet of blockchain that connects different chains, e.g., Cosmos [46] via Cosmos Hub design, Polkadot [47] via a relay chain design, and Quant Overledger [48].

Besides the above-related work, in literature, there exist several excellent review papers on blockchain interoperability, such as Buterin's review paper [9], Wang's SoK paper [8] and Belchior et al.'s survey paper [10].

VIII. Conclusion

This paper proposes an efficient blockchain interoperability scheme, *InterTrust*, which achieves interoperability and trusted services at the same time. InterTrust leverages both a threshold signature scheme and trusted hardware to achieve an atomic cross-AS communication. The threshold signature scheme is used to guarantee the consistency and verifiability of exchanged messages in the target blockchain system, and the trusted hardware is used to guarantee trusted services among distinct blockchain systems. By combining these two techniques, InterTrust can achieve robust blockchain interoperability. As future work, we plan to implement the proposed InterTrust scheme and thoroughly evaluate the throughput and transaction latency performance on real-world applications.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] G. Wang, "Sok: Applying blockchain technology in industrial internet of things," *Cryptology ePrint Archive*, vol. 2021, p. 776, 2021.

[3] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[4] W. Nikolakis, L. John, and H. Krishnan, "How blockchain can shape sustainable global value chains: an evidence, verifiability, and enforceability (eve) framework," *Sustainability*, vol. 10, no. 11, p. 3926, 2018.

[5] P. Wegner, "Interoperability," *ACM Computing Surveys (CSUR)*, vol. 28, no. 1, pp. 285–287, 1996.

[6] E. J. Scheid, T. Hegnauer, B. Rodrigues, and B. Stiller, "Bifröst: a modular blockchain interoperability api," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*. IEEE, 2019, pp. 332–339.

[7] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial internet of things," *IEEE transactions on industrial informatics*, vol. 14, no. 8, pp. 3690–3700, 2017.

[8] G. Wang, "Sok: Exploring blockchains interoperability." *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 537, 2021.

[9] V. Buterin, "Chain interoperability," *R3 Research Paper*, 2016.

[10] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on blockchain interoperability: Past, present, and future trends," *arXiv preprint arXiv:2005.14282*, 2020.

[11] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt, "Sok: communication across distributed ledgers." 2019.

[12] H. Pagnia and F. C. Gärtner, "On the impossibility of fair exchange without a trusted third party," Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Tech. Rep., 1999.

[13] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptology ePrint Archive*, vol. 2016, no. 086, pp. 1–118, 2016.

[14] S. Pinto and N. Santos, "Demystifying arm trustzone: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, pp. 1–36, 2019.

[15] R. Han, H. Lin, and J. Yu, "On the optionality and fairness of atomic swaps," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 62–75.

[16] T. Hegnauer, "Design and development of a blockchain interoperability api," Ph.D. dissertation, CSG@ IFI, University of Zurich, Switzerland, to appear 2019, 2019.

[17] H. Wang, Y. Cen, and X. Li, "Blockchain router: a cross-chain communication protocol," in *Proceedings of the 6th international conference on informatics, environment, energy and applications*, 2017, pp. 94–97.

[18] M. H. Miraz and D. C. Donald, "Atomic cross-chain swaps: development, trajectory and potential of non-monetary digital token swap facilities," *Annals of Emerging Technologies in Computing (AETiC) Vol*, vol. 3, 2019.

[19] B. Pillai, K. Biswas, and V. Muthukkumarasamy, "Blockchain interoperable digital objects," in *International Conference on Blockchain*. Springer, 2019, pp. 80–94.

[20] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM symposium on principles of distributed computing*, 2018, pp. 245–254.

[21] V. Shoup, "Practical threshold signatures," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2000, pp. 207–220.

[22] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *International Workshop on Public Key Cryptography*. Springer, 2003, pp. 31–46.

[23] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *Proceedings of the 6th ACM conference on Computer and communications security*, 1999, pp. 28–36.

[24] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "Smchain: A scalable blockchain protocol for secure metering systems in distributed industrial plants," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, 2019, pp. 249–254.

[25] 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*, 2019, pp. 347–356.

[26] G. G. Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. Reiter, D.-A. Seredinschi, O. Tamir, and A. Tomescu, "Sbft: a scalable and decentralized trust infrastructure," in *2019 49th Annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2019, pp. 568–580.

[27] V. Scarlata, C. Rozas, M. Wiseman, D. Grawrock, and C. Vishik, "Tpm virtualization: Building a general framework," in *Trusted Computing*. Springer, 2008, pp. 43–56.

[28] B. Obada-Obieh and A. Somayaji, "Can i believe you?: Establishing trust in computer mediated introductions," in *Proceedings of the 2017 New Security Paradigms Workshop*. ACM, 2017, pp. 94–106.

[29] S. Bajikar, "Trusted platform module (tpm) based security on notebook pcs-white paper," *Mobile Platforms Group Intel Corporation*, pp. 1–20, 2002.

[30] H. Raj, S. Saroiu, A. Wolman, R. Aigner, J. Cox, P. England, C. Fenner, K. Kinshumann, J. Loeser, D. Mattoon *et al.*, "ftpm: A firmware-based tpm 2.0 implementation," *Microsoft Research*, 2015.

[31] I. Bentov, Y. Ji, F. Zhang, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-time cryptocurrency exchange using trusted hardware," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1521–1538.

[32] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *USENIX Operating Systems Design and Implementation (OSDI)*, vol. 99, 1999, pp. 173–186.

[33] B. Lynn, "On the implementation of pairing-based cryptosystems," Ph.D. dissertation, Stanford University Stanford, California, 2007.

[34] S. Network, "libbls: a library for bls threshold schemes," https://github.com/skalenetwork/libBLS.

[35] "Crypto++ library 5.6.4," . http://www.cryptopp.com/, 2016.

[36] J. Baek and Y. Zheng, "Simple and efficient threshold cryptosystem from the gap diffie-hellman group," in *GLOBECOM'03. IEEE Global Telecommunications Conference (IEEE Cat. No. 03CH37489)*, vol. 3. IEEE, 2003, pp. 1491–1495.

[37] M. A.-A. Khandaker, Y. Nanjo, L. Ghammam, S. Duquesne, Y. Nogami, and Y. Kodera, "Efficient optimal ate pairing at 128-bit security level," in *International Conference on Cryptology in India*. Springer, 2017, pp. 186–205.

[38] P. S. Barreto, B. Lynn, and M. Scott, "Constructing elliptic curves with prescribed embedding degrees," in *International Conference on Security in Communication Networks*. Springer, 2002, pp. 257–267.

[39] B. Pillai, K. Biswas, and V. Muthukkumarasamy, "Cross-chain interoperability among blockchain-based systems using transactions," *The Knowledge Engineering Review*, vol. 35, 2020.

[40] W. Li, A. Sforzin, S. Fedorov, and G. O. Karame, "Towards scalable and private industrial blockchains," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, 2017, pp. 9–14.

[41] L. Kan, Y. Wei, A. H. Muhammad, W. Siyuan, L. C. Gao, and H. Kai, "A multiple blockchains architecture on inter-blockchain communication," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2018, pp. 139–145.

[42] Z. Liu, Y. Xiang, J. Shi, P. Gao, H. Wang, X. Xiao, B. Wen, and Y.-C. Hu, "Hyperservice: Interoperability and programmability across heterogeneous blockchains," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 549–566.

[43] G. Greenspan, "Multichain private blockchain-white paper," *URl: http://www. multichain. com/download/MultiChain-White-Paper. pdf*, pp. 57–60, 2015.

[44] I. Weber, Q. Lu, A. B. Tran, A. Deshmukh, M. Gorski, and M. Strazds, "A platform architecture for multi-tenant blockchain-based systems," in *2019 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2019, pp. 101–110.

[45] D. Ding, T. Duan, L. Jia, K. Li, Z. Li, and Y. Sun, "Interchain: A framework to support blockchain interoperability," *Second Asia-Pacific Work. Netw*, 2018.

[46] J. Kwon and E. Buchman, "Cosmos whitepaper," 2019.

[47] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," *White Paper*, vol. 21, 2016.

[48] G. Verdian, P. Tasca, C. Paterson, and G. Mondelli, "Quant overledger whitepaper," *Release V0. 1 (alpha)*, 2018.