4

Interoperability

The development of real-world blockchain applications based on different platforms (e.g., Bitcoin, Ethereum, Hyperledger Fabric, etc.) shows that blockchain technology is maturing at a fast pace. Blockchain adaptability is a motivating factor for the existence of different blockchains, leading to a heterogeneous ecosystem [92]. With the existence of different blockchains, some questions arose that need to be answered, such as: 1. What if the particular service running on the blockchain becomes vulnerable or is shut down? 2. If user requirements change for which a different blockchain is more appropriate, then how to handle this condition? and 3. What if one blockchain user wants to transfer something (asset/data) to another user of a different blockchain? All these questions can be answered if blockchain technology is enabled with the interoperability feature [28]. Blockchain interoperability can be defined in a simple term as the ability of one blockchain (source blockchain) to change the state of another blockchain (destination blockchain) enabled by cross-chain transactions. A technical report from the National Institute of Standards and Technology (NIST) defines blockchain interoperability as [142]

"*A composition of distinguishable blockchain systems, each representing a unique distributed data ledger, where atomic transaction execution may span multiple heterogeneous blockchain systems, and where data recorded in one blockchain are reachable, verifiable, and referable by another possibly foreign transaction in a semantically compatible manner.*"

There are three categories of blockchain interoperability [96]: first, Interoperability between different blockchains. Second, interoperability between decentralized applications using the same blockchain, and third, interoperability blockchain with technologies (i.e., integration with enterprise systems). In this work, we mainly focused on the first category of interoperability, i.e., interoperability between different blockchains. Therefore, from now on, when we say interoperability, we are referring to the first category of interoperability. While examining the state-of-the-art techniques, we identify the following challenges in existing interoperability solutions:

4.1 CHALLENGES IN INTEROPERABLE BLOCKCHAINS

Interoperability between different blockchain platforms remains one of the most significant hurdles the blockchain faces today. As blockchain technology continues to evolve and find applications across various sectors, the ability for different blockchain networks to seamlessly interact with one another is becoming increasingly crucial. However, achieving this interoperability is fraught with many challenges. These obsta-

cles come from differences in blockchain architectures, consensus mechanisms, the absence of standardized protocols, third-party intermediaries, and the need to satisfy the five essential characteristics that complicate the integration of diverse blockchain systems together.

1. **Different Blockchain Architectures:** Blockchains come in various architectures, each with specific protocols, consensus mechanisms, and security frameworks. For example, Bitcoin and Ethereum substantially differ in structure and function, complicating direct interchain interactions.
2. **Diverse Consensus Mechanisms:** Blockchains operate on varied consensus mechanisms. While Bitcoin uses PoW, others like Cardano use PoS. As the backbone of transaction verification, these mechanisms pose challenges when blockchains try to converse.
3. **Standards and Protocols:** The absence of standardized protocols complicates the development of solutions that enable seamless interaction between different blockchain systems, requiring significant effort and coordination to create interoperable frameworks.
4. **Third Party Intermediary:** Many current solutions for blockchain interoperability rely on third-party intermediaries. This requirement introduces potential vulnerabilities and deviates from the decentralized essence of blockchain technology, making systems more susceptible to attacks or failures centralized around these intermediaries.
5. **Presence of Five Essential Characteristics:** For accomplishing an efficient cross-blockchain transfer, blockchain interoperability must pose the five essential characteristics: non-repudiation, unlinkability, confidentiality, atomicity, and interoperability. The existing solutions have some characteristics, but not all of them.

4.2 *CROSSLEDGER*: A PIONEER CROSS-CHAIN ASSET TRANSFER PROTOCOL

We have proposed *CrossLedger*, a novel cross-chain asset transfer protocol to enable blockchain interoperability while addressing the above-mentioned challenges. It's important to note that, unlike the state-of-the-art approaches, the *CrossLedger* is a heterogeneous protocol, meaning that the source and destination blockchains for asset transfers are not fixed. Furthermore, *CrossLedger* also eliminates the necessity of a third-party intermediary. In contrast to state-of-the-art methods, which specify the source blockchain, *CrossLedger* allows for the use of multiple blockchains on both ends. For each blockchain, *CrossLedger* employs a novel Asset Forwarder Selection (AFS) Algorithm 7 to determine which Asset Forwarder (AF) forwards assets from the source blockchain to the destination blockchain. Between the asset forwarders on both blockchains, *CrossLedger* used the Trust Establishment (TE) Algorithm 8. Furthermore, the Asset Transfer and Confirmation (ATC) Algorithm 9 ensures that the asset is transferred successfully between blockchains. When compared to state-of-the-art approaches, all three of these algorithms maintain consistency across blockchains to give high levels of security, throughput, and scalability. The contributions of *CrossLedger* are as follows:

- We have proposed a generalized Cross-chain asset transfer protocol known as *CrossLedger* for two different types of blockchain platforms. Additionally, *CrossLedger* eliminates the third-party inter-

mediary using the AFS and TE algorithms. As a result, the proposed protocol is more secure and reliable.

- *CrossLedger* introduces a novel Asset Forwarder Selection (AFS) algorithm to select an Asset Forwarder (AF) in each blockchain, i.e., source and destination blockchains. The AF is responsible for transferring the asset from the source blockchain to the destination blockchain. *CrossLedger* uses a novel Trust Establishment algorithm between the AFs of both the blockchains. A novel Asset Transfer and Confirmation (ATC) algorithm ensures that the asset is transferred successfully between blockchains.
- The proposed *CrossLedger* structure supports non-repudiation, unlinkability, confidentiality, atomicity, and interoperability characteristics for Cross-chain based asset transfer.
- We have provided the security analysis of the *CrossLedger*, which illustrates that the *CrossLedger* is secured against the double-spending attack, liveness attack, and Sybil attack.

4.2.1 System Model

This section describes the system model of our proposed scheme. Figure 4.1 depicts the typical process of asset transfer from user U_A to user U_B. There are two blockchains, blockchain A and blockchain B. These blockchains are of any type, i.e., public-private, public-public, and private-private. In other words, blockchain A can be Bitcoin type, and blockchain B can be Ethereum type or Hyperledger fabric type. The *CrossLedger* is a generalized asset transfer protocol where the source and destination blockchains are not fixed. The numbers from 1 to 10 shown in the figure are the sequence of the operations for the asset transfer. Apart from the users U_A and U_B, the other elements of the system are miners and Asset Forwarders AF_A, AF_B. The highlighted blocks B_3 and B_5 are the source and destination blocks, respectively, showing that the asset(s) is/are included in those blocks. When the asset is successfully transferred, the AF_B is responsible for confirming the asset to both the users U_A, U_B. Below, we have defined the system model's main components: User, Asset, Asset Forwarder, and Asset Transfer Confirmation.

1. **User:** A User denoted as U_A/U_B is the one who generates transactions. The actual asset transfer happens between the users, where one user acts as the sender, and another acts as the receiver of the asset. In Figure 4.1, user U_A is the sender, and user U_B is the receiver that belongs to two different blockchains, blockchain A and blockchain B.
2. **Asset:** An Asset *Ast* is a digital representation of arbitrary data, fungible assets, or non-fungible assets. Arbitrary data is often represented via a key-value pair, which is a suitable representation of some blockchains. The fungible assets are usually the payment tokens, and non-fungible tokens are utility tokens such as ERC20 tokens.
3. **Asset Forwarder (AF):** A AF is one of the nodes selected from the distributed network based on the consensus algorithm executed in the blockchain network. The AF is responsible for the asset transfer and confirmation from the source blockchain to the destination blockchain. An AF is selected on both the blockchain, i.e., source blockchain A and destination blockchain B.
4. **Asset Transfer and Confirmation (ATC):** The ATC algorithm verifies the successful transfer of the asset *Ast* from the source blockchain to the destination blockchain. In practice, rolling back or modifying the asset *Ast* after confirmation is computationally difficult.

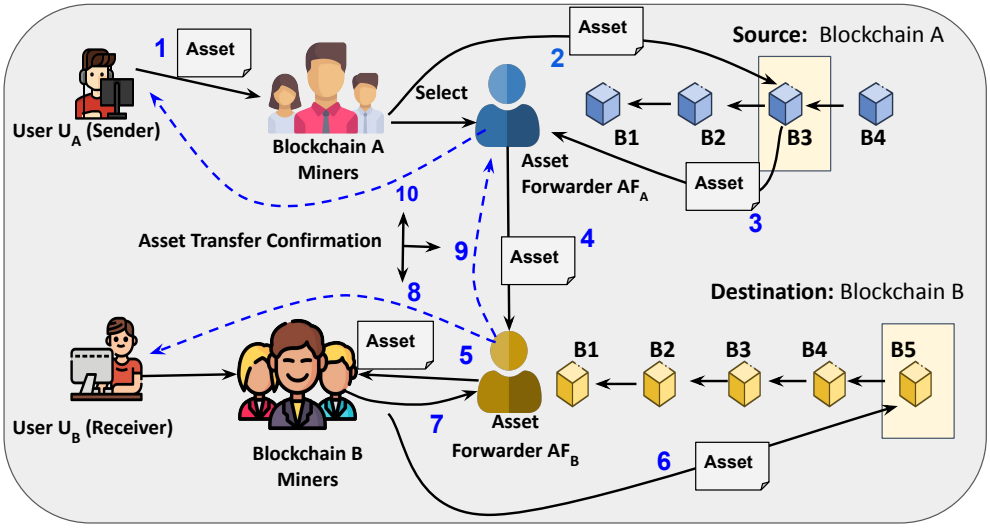


Figure 4.1: *CrossLedger*'s asset transfer mechanism between two blockchains, source blockchain A and destination blockchain B

The *ATC* certifies that the asset *Ast* is successfully committed from the source blockchain to the destination blockchain. In effect, it is impossible to computationally revert or alter the asset *Ast* after the confirmation. In *CrossLedger*, *ATC* is deterministic. When a *Ast* is included in a block and appended to the blockchain, it is termed deterministic confirmation. This is accomplished through the proposal of the block by an elected *Node* with adequate authority. In this instance, an asset forwarder is chosen and accountable for asset transfer and confirmation.

4.2.2 Background

This section briefly describes the taxonomy of the *CrossLedger* scheme, which includes the basic definition of Ownership and the adversary model. In the adversary model, we have defined some assumptions that make it possible for the adversary to attack the system. We have also proved in the later section how the adversary cannot make an attack.

A. Five Essential Properties for Cross-Blockchain Transfer

In CrossLedger, we have considered five essential properties that are crucial for a solution of cross-chain asset transfer. The five properties are as follows: Non-Repudiation, Unlinkability, Confidentiality, Atomicity, and Interoperability. Below, we define these properties and why we consider them essential while designing CrossLedger.

1. **Non-Repudiation:** This property ensures that once a transaction has been completed, i.e., sent by the sender and received by the receiver, neither the sender nor the receiver can deny their involvement in the transaction, which means it provides irrefutable proof of the origin and reception of the asset transfer. This property becomes highly important in a decentralized environment where there is no central authority to enforce the contracts. The proposed CrossLedger is also a protocol to transfer assets across the different blockchains without the involvement of a third party. Therefore, keeping the

non-repudiation property is highly important. Non-repudiation ensures that once an asset is recorded, both chains are accountable for their actions.

2. **Unlinkability:** This property ensures that transactions or assets transferred across blockchains cannot be linked back to their origin or to each other, thus protecting the privacy of users. In cross-blockchain environments, preserving user privacy is a significant concern. Unlinkability prevents adversaries from tracing transactions across different blockchains, which could reveal sensitive information about user behavior or asset ownership. This is particularly important when dealing with public blockchains, where transaction data is visible to all participants.

3. **Confidentiality:** This is an important security property for any digitally secured system, especially when the system includes some sensitive information in the transactions. Confidentiality ensures that the details of the assets or transaction, such as the value of the asset, sender, and recipient, are not accessible to unauthorized parties. Confidentiality is equally essential in cross-chain transfers to protect sensitive information from being exposed during the transfer process. As transactions move between different blockchains, which may have varying levels of security and transparency, ensuring that only the intended parties can view the transaction details is crucial for maintaining trust and security.

4. **Atomicity:** The concept of atomicity is drawn from the database system, which ensures that either the transactions are fully completed or not executed at all. This means that if an asset is transferred from one blockchain to another, the transfer must be entirely successful on both blockchains, or the transaction should be rolled back. Atomicity is vital to prevent partial transfers that could lead to asset loss or inconsistencies between blockchains. In a cross-chain environment, where different blockchains might have different rules and processing times, atomicity ensures that assets are not left in an incomplete state, thereby preserving the integrity of the transfer process.

5. **Interoperability:** This is the core essential property for any cross-chain transfer protocol. Any protocol is said to be interoperable if it ensures the ability of different blockchain networks to work together, allowing assets, data, and transactions to move seamlessly between them without requiring a trusted intermediary. The intent of designing the *CrossLedger* is also the same: there should not be any intermediary or third party to do the task of transferring from one blockchain to another blockchain. Without interoperability, cross-chain transactions would not be possible, as each blockchain would function in isolation. Interoperability ensures that different blockchains, regardless of their underlying technology, can communicate and process transactions in a coordinated manner.

B. Ownership

The ownership of the asset is defined using the following definition.

Definition 5 (Ownership). *The ATC can only be successfully executed if the following conditions are satisfied.*

1. *User U_A (sender) owns the Asset_Value.*
2. *User U_A (sender), U_B (receiver) should be such that ($(U_A, U_B) \in \bigcup_{i=1}^{n} U$), where U is the set of all the legitimate users in both the blockchains.*
3. *Asset_Value should belongs to the legitimate user ($\in U$) and Asset_Value $\in \bigcup_{i=1}^{m} Asset$.*

4. *An asset has exactly one owner at a time* $Owner(U_A, Asset_Value)$ *is true if and only if* U_A *owns* $Asset_Value$.

C. Asset Evaluator

The present value of assets is calculated using book values, fundamental valuation methodologies such as discounted cash flow analyses, option pricing models, or comparables. The following is the definition of the asset evaluator function in terms of $CrossLedger$.

- It begins by verifying that the asset forwarder has enough amount of the asset. This step is known as the balance function, and it produces the following result: If the asset forwarder has sufficient units of the asset, then the value is true; otherwise, the value is false.
- If the prior transaction references the current transaction, it is checked to see if the previous transaction is part of a valid block.

If both of the above steps are true, the asset Verifier function returns true; otherwise, it returns false. After completing all transactions, they return a receipt with all the transaction details.

D. Traceable Ring Signature (TRS)

Unlike ring signatures, *TRS* limits "excessive" anonymity [143]. A *TRS* is a tuple of four elements as $\sum_{TRS} = (\mathbb{G}, \mathbb{S}, \mathbb{V}, \mathbb{T})$. We consider the ordered set of public key $\mathbb{S}|_{pk} = (pk_{N_1}, ..., pk_{N_n})$ as a ring. The four tuples of the *TRS* can be defined as follows.

1. **Generator ($\mathbb{G}$):** It is a function that takes the security parameter, let's say, k as input and generates the output public-secret key-pair (pk_{N_i}, sk_{N_i}).

$$(pk_{N_i}, sk_{N_i}) \leftarrow \mathbb{G}(k) \tag{4.1}$$

2. **Signature ($\mathbb{S}$):** It is a function that takes three parameters as input, which is the secret key sk_{N_i}, tag field $Tg = (issue, pk_{N_n})$, and message Msg, and generate the outputs as signature σ_{Msg}.

$$\sigma_{Msg} \leftarrow \mathbb{S}(sk_{N_i}, Tg, Msg) \tag{4.2}$$

3. **Verifier ($\mathbb{V}$):** It is the function of signature verification that takes the tag field Tg, message Msg and signature σ_{Msg}, as the input and generates the one-bit boolean output as 0/1, indicate the valid or invalid signature.

$$Boolean\ V \leftarrow \mathbb{V}(Tg, Msg, \sigma_{Msg}) \tag{4.3}$$

4. **Tracebility ($\mathbb{T}$):** It is a function that takes the parameter, tag field $Tg = (issue, pk_N)$, and pair of message and its corresponding signature, (Msg, σ_{Msg}), (Msg', σ'_{Msg}), and generates the output one of the following strings "indep," "linked," or pk_{N_i}, where $pk_{N_i} \in \mathbb{S}|_{pk}$. This can be represented as shown in (4.4).

$$T(Tg, Msg, \sigma_{Msg}, Msg', \sigma'_{Msg}) = \begin{cases} indep & if\ N_i \neq N'_i, N_i \in N \\ linked & else\ if\ Msg = Msg' \\ Pubk_{N_i} & otherwise \end{cases} \tag{4.4}$$

For simplicity, we often write $(pk_N, sk_N) \leftarrow Gen(1^k)$ to denote the experiment of $(pk_i, sk_i) \leftarrow Gen(1^k)$ for $i \in N$ and assigning $(pk_N, sk_N) = (pk_i, sk_i)_{i \in N}$. As an ordinary signature scheme, a *TRS* scheme must satisfy the following correctness conditions: For every $k \in N$, every $n \in N$, every $i \in N = 1, ..., n$, every issue $\in 0, 1^*$, and every $m \in 0, 1^*$, if $(pk_N, sk_N) \leftarrow Gen(1^k)$, and $\sigma \leftarrow \mathbb{S}_{sk_i}(L, m)$, where $L = (issue, pk_N)$, it holds with an overwhelming probability (ink) that $\mathbb{V}(L, m, \sigma) = 1$.

E. Why We Need TRS?

TRS is a crucial cryptographic tool used in *CrossLedger* to enhance the security and privacy of cross-chain asset transfers. We have used the *TRS* in *CrossLedger* due to its unique benefits compared to other signature schemes, particularly in the context of decentralized and cross-chain environments. Following are the key benefits of *TRS* over the other signature schemes, which are very important in the context of the cross-chain environment, such as *CrossLedger*.

- **Anonymity with Accountability:** Traditional ring signatures provide a high level of anonymity, allowing a signer to remain indistinguishable within a group of users. However, this anonymity can lead to challenges in accountability, making it difficult to trace malicious activities or prevent double-spending. *TRS*, on the other hand, maintains the anonymity feature but introduces a traceability aspect. This means that while the signer remains anonymous within the ring, their actions can be traced if necessary, enabling the system to link multiple signatures from the same user without revealing their identity. This is particularly important in cross-chain transactions, where accountability must be balanced with privacy.
- **Ensuring the Privacy of Asset Forwarder:** During the trust establishment phase between AFs of different blockchains, *TRS* is used to sign the Asset Forwarder Verification (AFV) message. This message is broadcast across the network of the opposite blockchain to establish trust. The *TRS* ensures that the identity of the asset forwarder remains anonymous within a set of possible signers while still allowing the network to link and verify multiple AFV messages if needed. This enhances the trustworthiness of the asset forwarders without compromising their privacy.
- **Enhanced Security Against Sybil Attacks:** Sybil attacks, where a single entity generates multiple identities to manipulate the network, are a concern in decentralized systems. *TRS* adds a layer of security against such attacks by ensuring that even if a malicious user creates multiple identities, their actions can be traced and linked if they attempt to sign multiple messages, thereby preserving the integrity of the cross-chain transfer process.
- **Ensuring Non-Repudiation and Unlinkability:** *TRS* contributes to non-repudiation by ensuring that once a transaction is signed, the signer cannot deny their involvement, as their signature can be traced if necessary. At the same time, *TRS* supports unlinkability by making it difficult for an adversary to link multiple transactions from the same user unless they attempt malicious activity, such as double-spending. This balance of traceability and unlinkability is crucial for the secure and private transfer of assets between different blockchains.

F. Attack Model

Asset transfer is a very crucial cross-chain communication between two blockchains. A malicious user, say $\mathbb{U}_M$, is always possible to intervene in the communication or present itself as the legitimate user. Here, we consider some assumptions for the security and performance of the system.

- The malicious user $\mathbb{U}_M$ may attempt to exploit the open nature of the communication channel between blockchains. The malicious user might try to compute secret values if a node's private credentials are compromised. However, *CrossLedger* mitigates this threat through the use of secure cryptographic techniques, including *ECC* and *TRS*, which ensure that even if a communication channel is open, the transmitted data remains secure and confidential.

- We have also considered that, the AF could act as malicious asset forwarder denoted as AF_M. The AF_M might attempt to double-spend assets, delay transactions, or intentionally fail to transfer assets to the destination blockchain. However, the *CrossLedger* mitigates this risk through its Asset Forwarder Verification message ($Mesg_{AFV}$), where each AF must obtain verification from a majority of nodes in the opposite blockchain. The *TRS* mechanism ensures that AFs cannot easily forge verification messages, and any discrepancies in behavior are quickly detected and addressed by the network.

- $\mathbb{U}_M$ could attempt to execute a double-spending attack by either spending the same asset twice (discussed in Lemma 6 Case 1) or by creating duplicate/conflicting assets (discussed in Lemma 6 Case 2), i.e., $Owner(\mathbb{U}_M), Asset_Value_A$ is true, where $Asset_Value_A$ belongs to some legitimate user say U_A. *CrossLedger* defends against this by requiring mutual verification and trust establishment between AFs on both blockchains. The *TRS* ensures that only legitimate transactions are processed, making double-spending virtually impossible.

- The $\mathbb{U}_M$ might generate multiple fake identities to manipulate the asset transfer process (discussed in Lemma 8 as Case 1) or attempt to install a fake node as an AF (discussed in Lemma 8 as Case 2). *CrossLedger* counters this by selecting AFs based on their previous performance and verification by other nodes in the network. The use of the *Algorithm* 7 and *Algorithm* 8 ensures that only legitimate nodes can function as AFs, while *TRS* helps verify the authenticity of the AFs.

- The $\mathbb{U}_M$ might attempt to delay the $Mesg_{AFV}$, disrupting the asset transfer process. *CrossLedger*'s security model includes a threshold time parameter, which ensures that any undue delay in $Mesg_{AFV}$ verification results in the transaction being rolled back, thus protecting the system from liveness attacks, i.e., it is not possible due to the atomicity characteristic of *CrossLedger*.

- The attacker model also considered the possibility of collusion among multiple AFs or nodes to manipulate asset transfers. *CrossLedger* addresses this by requiring cross-chain verification of each transaction, where the AFs on each blockchain must independently verify and agree on the transaction's validity. The use of *TRS* further reduces the risk of collusion by linking and tracing transaction signatures, making it difficult for colluding nodes to escape detection.

- **Trust Assumptions:** 1. We have assumed that the majority of nodes in the network act honestly and that cryptographic techniques (such as *ECC* and *TRS*) used are secure. We also assume that initially selected AFs are trustworthy, but this trust is continuously verified through the network's decentralized processes. Any deviation from expected behavior by an AF leads to immediate corrective action, including the possibility of excluding the AF from future transactions.

Additionally, the trust between AFs is established via the *Algorithm* 8, which requires verification from multiple nodes across blockchains. The *TRS* ensures that this process is both secure and traceable, thereby reinforcing the trust assumptions built into the system.

4.2.3 Methodology

The methodology of *CrossLedger* consists of three steps for asset transfer between two different blockchains. The first step is asset forwarder selection, the second step is trust establishment, and the third step is asset transfer and confirmation. In asset forwarder selection, one of the nodes from blockchain networks is selected as an asset forwarder AF in both the blockchains (i.e., blockchain A and blockchain B). After the successful selection of an AF, the next step is establishing trust between these two asset forwarders, i.e., AF_A and AF_B from blockchain A and blockchain B, respectively. To establish trust, the whole network participates in the trust establishment process. When the trust is established, the AF transfers the asset from blockchain A to blockchain B. To describe the proposed scheme, we have considered blockchain A as the source blockchain and blockchain B as the destination blockchain. A detailed description of all three steps is given below.

Step-1: Asset Forwarder Selection (AFS)

The *Algorithm* AFS is straightforward yet very effective. In the proposed asset forwarder selection, we have received help from the blockchain's base consensus algorithm. This means that if blockchain A runs the PoW consensus algorithm, the asset forwarder is selected based on that; similarly, if blockchain B runs the PoS consensus algorithm, the asset forwarder is determined based on the PoS consensus. The *Algorithm* 7 works as follows:

1. The algorithm is simple and takes negligible time to select the AF. Consider AF_A as the asset forwarder of source blockchain A and AF_B as the asset forwarder of destination blockchain B.

2. The current previous miner (CPM) is selected as the AF. There are two conditions to select the CPM as AF.
   1. If the current miner (CM) is not equal to the current previous miner, i.e., if the current miner is not again selected as the miner for the next block generation. Then, the CPM is selected as the AF.
   2. If the CPM is also a CM, then the previous miner (PM) of the CPM (denoted as PM_of_CPM) is selected as the AF. This step can be understood as, let there are three nodes N_1, N_2, and N_3, who generate three blocks B_1, B_2, and B_3, respectively. The blocks are arranged as $B_1 \leftarrow B_2 \leftarrow B_3$. B_3 is the latest block generated by node N_3, and B_2 is the parent block of B_3 generated by the node N_2, in this case the node N_2 is the CPM of node N_3. Now let's suppose block B_2 and B_3 both are generated by the same node N_2 and block B_1 is generated by node N_1. Then, in that case, the node N_1 is PM_of_CPM.

3. When the AF is selected, then a token is generated as $TokenA = createToken(ID_M, T, PBH)$ which includes the ID of CPM, Timestamp, and hash of the previous block PBH. After generating, the token is used for the next step, i.e., trust establishment.

Algorithm 7: Asset Forwarder Selection (AFS) Algorithm

Input: CM: Current Miner, CPM: Current Previous Miner, ID_M: Miner ID of CPM, T: Timestamp, PBH: Previous Block Hash

Output: Asset Forwarder Node

Function `AFS`$(CM, CPM, PM_of_CPM, T, ID_M, PBH)$**:**

- **if** $CM \neq CPM$ **then**
  - $AF \leftarrow CPM$;
  - `/* Current Previous Miner is selected as an Asset forwarder. */`
- **else**
  - $AF \leftarrow PM_of_CPM$;
  - `/* Previous Miner of Current Previous Miner is selected as an Asset forwarder. */`
- **if** $AF.isSelected() = TRUE$ **then**
  - $TokenA = createToken(ID_M, T, PBH)$;
  - $Send_to_BlockchainB(TokenA)$;

Step-2: Trust Establishment and Parameter Exchange

After the selection of the Asset forwarder (AF), the next step is trust establishment between the AFS of both the blockchains, i.e., blockchain A and blockchain B, as described in *Algorithm* 8. To establish trust between both the forwarders, each forwarder broadcasts the signed Asset Forwarder Verification (AFV) message in the network of the opposite blockchain. The AF_A sends the signed AFV message to the network of blockchain B, and AF_B sends the signed AFV message to the network of blockchain A. A TRS is used to sign the AFV message. Each node of the opposite blockchain verifies the signed AFV message. If the AFV message is successfully verified, each node sends the verified message to the AF of the opposite blockchain. If AF_A receives n verification from the network of blockchain B within a timeline, and similarly, AF_B receives the n verification from the network of blockchain A within the timeline, then the trust is established. Here, n is the majority number of honest nodes in each blockchain. Figure 4.2 shows the necessary parameter exchange between asset forwarders after establishing trust between them. This parameter exchange is highly important to understand the block structure and consensus algorithm of both blockchains.

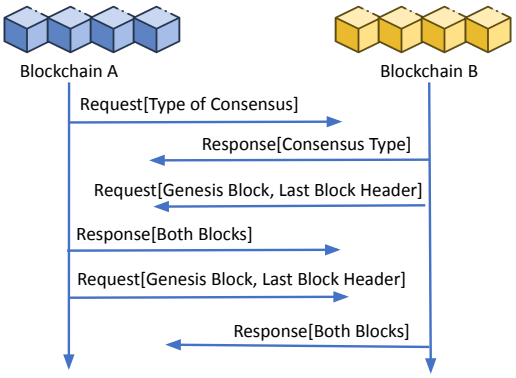


Figure 4.2: Essential parameter exchange between AF_A and AF_B

Algorithm 8: Trust Establishment Algorithm

Input: T: Timestamp, $TokenA, TokenB$: Token of AF_A and AF_B, ID_M: Miner ID of CPM, $send_t$: Sending time of (Msg_{AFV}), $recv_t$: Receiving time of n verification, $thre_t$: Threshold time, $diff_t$: Time difference between $recv_t$ and $send_t$

Output: Trust_Status

Function `TEA(`$T, TokenA, TokenB, ID_M$`)`**:**

```
/* Creation of AFV message                                          */
```
$Msg_{AFV} = Enc_{sk_B}(ID_M||T||TokenA)$;

Blockchain B $\leftarrow send(Msg_{AFV})$;

if $Verification_count(Msg_{AFV}) == n$ ***and*** $diff_t \leq thre_t$ **then**

$Trust_Status.AF = Valid$;

```
/* n is the total number of nodes (sender/receiver)                 */
```

else

$Trust_Status.AF = Invalid$;

Step-3: Asset Transfer and Confirmation (ATC)

This is the third and last step of the scheme. After the successful completion of step-1 (Asset forwarder selection) and step-2 (trust establishment and parameter exchange), the system is ready to transfer an asset from source blockchain A to destination blockchain B. The ATC can only be successfully executed if the sender U_A owns the $Asset_Value$, i.e., An asset has exactly one owner at a time $Owner(U_A, Asset_Value)$ is true if and only if U_A owns $Asset_Value$. Consider U_A and U_B as a domain of users, and asset $Asset_Value_A$, owned by the user U_A, belongs to the set of legitimate assets which can be defined as $Asset_Value_A \in \bigcup_{i=1}^{m} Asset$. Also, consider the $\mathbb{S}|_M$ is the set of miners and any AF is selected if and only if $AF_A/AF_B \in \mathbb{S}|_M$.

We have used Elliptic Curve Cryptography (ECC) and a Secure Hash Algorithm (SHA256) to ensure the confidentiality and authenticity of the asset transfer. The key pair of asset forwarder AF_A is: sk_A, pk_A, and kay pair of asset forwarder AF_B is: sk_B, pk_B. Figure 4.3 shows the sequence of asset transfer communication steps. U_A first sends the asset to its blockchain network. The blockchain network puts the asset in the latest block and also puts that asset in the Asset Transaction Pool (ATP_A). From the ATP_A, the asset is transferred to ATP_B. The AF_B takes the asset from the ATP_B and sends it to the network of blockchain B. Blockchain B first verifies the authenticity of the received asset, and if it is verified, the confirmation is sent to both the users, i.e., sender and receiver. The AF_B also includes the asset transaction in the latest block of blockchain B. This is because, if it is required to track the asset's origin, this block can be used for traceability only with legitimate AF. The node can never trace the block if it is not legitimate AF. Thus, it provides the traceability of the assets. The ATC *Algorithm* 9 works as follows:

1. If the node is AF_A, i.e., asset forwarder of the source blockchain, then it is responsible for the asset transfer. To accomplish this task, AF_A first takes the hash of the asset to be transferred, i.e., $Asset_hash = Hash(Asset_Value)$. After that, it generates the nonce value from the function called $TRNG()$, which is the true random number generator. It also takes the hash of the previous block, i.e., the hash of the previous block of the latest block, which is to be mined. After taking all the values, AF_A create the asset transaction $ATx = (Asset_hash||T||PBH||nonce)$ and send it using ECC encryption and digital signature as $send(\sigma_{ATx}, Enc_{sk_B}(ATx, Asset_Value))$.

Algorithm 9: Asset Transfer and Confirmation (ATC)

Input: PBH: Previous Block Hash, $Hash()$: Hash function using SHA256, ATx: Asset Transaction, B_i: i^{th} Block, T: Timestamp, AF: Asset Forwarder

Output: Asset Transfer

Function `ATC(`ATx, B_i, PM_of_CPM, T, ID_M, PBH`)`**:**

- **if** *node* == *AF(Sender):*AF_A **then**
  - $Asset_hash = Hash(Asset_Value)$;
  - $nonce = TRNG()$; `/* TRNG() is true random number generator */`
  - $PBH = Hash(previous_block)$;
  - $ATx = (Asset_hash||T||PBH||nonce)$;
  - $\sigma_{ATx} = Sign_{sk_A}(ATx, nonce)$;
  - $send(\sigma_{ATx}, Enc_{sk_B}(ATx, Asset_Value))$ to Blockchain B;
- **else**
  - **if** *node* == *AF(Receiver):*AF_B **then**
    - receive$(\sigma_{ATx}, Enc_{pk_B}(ATx, Asset_Value))$;
    - $ATx, Asset_Value = Dec_{sk_B}(ATx, Asset_Value)$;
    - $Asset_hash = Hash(Asset_Value)$;
  - **else if** $Verify(pk_A, ATx, \sigma_{ATx}) == TRUE$ **then**
    - $Asset_hash = Hash(Asset_Value)$;
    - $ATx* = (Asset_hash||T||PBH||nonce)$;
    - **if** $ATx == ATx*$ **then**
      - $Asset_status(valid)$;
    - **else**
      - $Asset_status(not_valid)$;
      - $node.status = bad$;
- **while** $N_i \leq N$ **do**
  - `/* N=total miners, and` $N_i \in \mathbb{S}|_M$ `*/`
  - **if** $count(Asset_status == valid) \geq 75\%$ **then**
    - $blockchain_B.append(CB(Asset_hash, T, PBH, nonce, ATx))$;

2. At the receiving end, the AF_B is responsible for doing all the processing. It first decrypt and verify the received values as $ATx, Asset_Value = Dec_{sk_B}(ATx, Asset_Value)$ and $Verify(pk_A, ATx, \sigma_{ATx})$. If the signature verification is confirmed, then it calculates the asset hash value as $Asset_hash = Hash(Asset_Value)$ and then re-generates the asset transaction as $ATx* = (Asset_hash||T||PBH||nonce)$, after that, it compares both the transactions, i.e., if $ATx == ATx*$, then it is verified that the asset transfer is correct. The AF_B notifies both users about the confirmation of the asset transfer. After the confirmation, the asset is appended into blockchain B as $blockchain_B.append(CB(Asset_hash, T, PBH, nonce, ATx))$. Here, CB is the create block function.

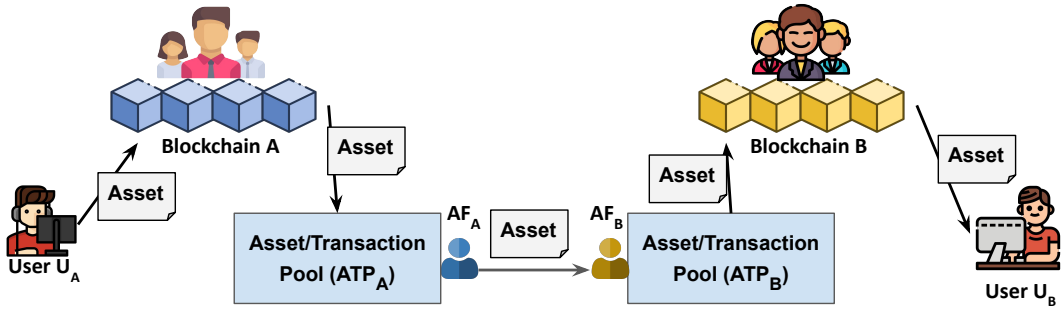


Figure 4.3: Sequence for asset transfer communication from blockchain A to blockchain B

4.2.4 Essential Characteristics of Interoperable Blockchahin

This section explains the characteristics proven by the *CrossLedger*. As mentioned in the previous sections, this scheme supports non-repudiation, unlinkability, confidentiality, atomicity, and interoperability.

Lemma 1 (Non-repudiation)**.** *CrossLedger supports non-repudiation. Legitimate AF are either the sender or the recipient of an asset.*

Proof. *CrossLedger* selects AF_A to transfer the asset from the sender to the receiver blockchain. To transfer the asset, AF_A must go through a few steps in the *Algorithm* 9. The first step is for AF_A to send σ_{ATx} and $Enc_{pk_B}(ATx, Asset_Value)$ to the receiver blockchain's AF_B for asset transfer. The second step is for AF_B to count AF_A's total $asset_status$, which are *valid*. AF_B checks the validity based on signature verification and received parameters. As a result of successful signature verification and a valid $asset_status$ count, AF_A cannot deny asset transfer. If the validity of $asset_status$ is greater than or equal to 75%, then the block is appended to the blockchain, and AF_A gets the acceptance of asset confirmation from the AF_B. So AF_A can not decline because it has not transferred asset to AF_B; similarly, AF_B also can not decline because it has not received asset from AF_A. As shown in Figure 4.1, the sequence number from 1 to 10 shows the complete sequence of asset transfer, where sequence numbers 9 and 10 are the confirmation acknowledgement of asset transfer from AF_B to AF_A and from AF_A to U_A. If the confirmation is not received from the AF_B, the AF_A considers the transfer incomplete due to some error and will be rolled back. Further, after sending the transfer confirmation, AF_B also can not deny the acceptance of the asset. Hence, it is proved that the *CrossLedger* supports non-repudiation. □

Lemma 2 (Unlinkability)**.** *Let there are two distinct assets Ast_{A1} and Ast_{A2} such that $Owner(U_A, Ast_{A1}, Ast_{A2})$. User U_A transfers those assets to U_B and U_C such that $Owner(U_B, Ast_{A1})$ and $Owner(U_C, Ast_{A2})$. Then for any malicious user/adversary U_M, it is impossible to establish a link between both the assets transferred from the same user U_A.*

Proof. In *Algorithm* 9, we have described how the asset is transferred from one blockchain to another, i.e., from source blockchain A to destination blockchain B. As shown in Figure 4.1 there are two users, U_A (sender at blockchain A) and U_B (receiver at destination blockchain B), also $(U_A, U_B) \in \bigcup_{i=1}^{n} U$, meaning that user U_A and U_B both are legitimate users. Let user U_A hold assets Ast_{A1} and Ast_{A2}, which is recorded in the source blockchain A on block B_3. U_A wants to share that asset with U_B and U_C, which is on another

blockchain (blockchain B, our destination blockchain). However, it is not possible for any user to directly transfer the asset because, in *CrossLedger*, only AF is entitled to transfer and receive the asset.

Let some malicious user or adversary U_M want to establish the link between the two distinct assets transferred from the same sender (U_A). It is impossible for U_M for two reasons: first, all the asset transfers are done via AF only, i.e., there is no possibility of direct asset transfer between two users. Therefore, it is impossible to link the two asset transfers from the same sender. Second, it is important to have the address of two distinct assets sent from the same sender to establish the link for U_M. However, in *CrossLedger*, the asset is bundled in the block along with the other information in both the blockchain, i.e., source and destination blockchain. Additionally, as per the *Algorithm* 9, the asset is transferred as follows: First, the hash of the asset is calculated $Asset_hash = Hash(Asset_Value)$. After that, an asset transaction is created as $ATx = (Asset_hash||T||PBH||nonce)$, which is further signed using sk_A as $\sigma_{ATx} = Sign_{sk_A}(ATx, nonce)$ and then it is sent to blockchain B using ECC based encryption as $send(\sigma_{ATx}, Enc_{sk_B}(ATx, Asset_Value))$. It is clear from the above computation that it is impossible to reveal the address of the asset in any phase of the asset transfer. Furthermore, the AF is selected (described in *Algorithm* 7) in every epoch, which further minimizes the possibility of linkability. Thus, it is proved that the *CrossLedger* supports the Unlinkability of two distinct assets from the same sender.

□

Lemma 3 (Confidentiality)**.** *CrossLedger achieves confidentiality while transferring assets between two different blockchains.*

Proof. In Algorithms 7 & 8, we have defined how a AF is selected and how the trust is established between both the AFs of both the blockchains. Also, as mentioned earlier, *Algorithm* 9 described the asset transfer and confirmation process. To transfer an asset while ensuring confidentiality, the AF does the following computation. First, the hash of the asset is calculated ($Asset_hash = Hash(Asset_Value)$). After computing the *nonce* and PBH, the AF create the asset transaction using the earlier computed values (ATx =$(Asset_hash||T||PBH||nonce)$). To ensure the authenticity the AF sign the ATx ($\sigma_{ATx} = Sign_{sk_A}(ATx, nonce)$) and then to ensure the confidentiality, AF encrypt the ATx and $Asset_Value$ using ECC based cryptographic function ($send(\sigma_{ATx}, Enc_{sk_B}(ATx, Asset_Value))$). After that, it sends it to blockchain B. The receiver AF first decrypts the received message and then verifies the signature. If it is verified, it is ensured that the message was not modified during the transfer. Additionally, it is assumed that the private key is not compromised. Thus, the *CrossLedger* supports confidentiality.

□

Lemma 4 (Atomicity)**.** *The CrossLedger is said to have atomicity characteristics if any asset Ast_A belongs to some user U_A is either successfully transferred from one source blockchain A to destination blockchain B or aborted if some error occurs.*

Proof. There are two ways to understand the atomicity in *CrossLedger*.

1. As shown in Figure 4.1 and *Algorithm* 9, when AF_B receives asset transaction from AF_A, it broadcasts this transaction to $Blockchain_B$ network and executes two comparison statements.

(a) The first comparison operation checks the equality between ATx and $ATx*$. If this statement returns true, then the *Asset_status* is marked as valid.

(b) The second comparison operation checks the validity of *Asset_status* is more significant than equal to $(3/4)*n$.

If both comparison statements return true, the transaction is committed in the block and appended to the blockchain. AF_B forward transaction confirmation to AF_A in case of commit otherwise rolled back the transaction. This statement proves that *CrossLedger* is atomic.

2. The atomicity of *CrossLedger* can also be understood as per the ownership property. According to the definition of ownership property, a unique asset belongs to only one owner at a time. Let asset Ast_A belongs to user U_A such that $Owner(U_A, Ast_A)$. The ownership of the asset can be changed if and only if the asset is successfully transferred from source blockchain A to destination blockchain B, for which we followed the *Algorithm* 9. When the asset is successfully transferred, then the asset belongs to only U_B ($Owner(U_B, Ast_A)$). Thus, *CrossLedger* ensures the Atomicity in asset transfer.

□

Lemma 5 (Interoperability)**.** *The cross-chain asset transfer protocol CrossLedger is called interoperable across different types of blockchains if and only if it allows the transfer of assets efficiently between two heterogeneous blockchains.*

Proof. Interoperability is an important challenge when we deal with cross-chain value transfer. *CrossLedger* provides interoperability for the cross-chain asset transfer. Unlike the state-of-the-art schemes where one of the blockchains is fixed (either source or destination), in *CrossLedger*, both the blockchains are not fixed and are of heterogeneous type. Usually, the federation approach is the most preferable approach to enable interoperability. In this approach, both the blockchains (source and destination) rely on the trusted third party to confirm the asset value transfer of one chain to another [144]. However, in *CrossLedger*, the asset transfer in two different blockchains is done following the three novel algorithms (*Algorithm* 7, 8, and 9). In *Algorithm* 7, first, we select the AF out of the existing blockchain network nodes in both the blockchains. After that, using *Algorithm* 8, a trust is established between both the asset forwarders, i.e., between AF_A and AF_B. Therefore, instead of relying on a third party, the *CrossLedger* establishes trust between two selected AFs. Thus, the *CrossLedger* eliminates the dependency on a third party to provide interoperability, which also assures higher security and privacy, as no third party is involved in the communication. The parameter exchange shown in Figure 4.2 strengthens the interoperability. According to Figure 4.2, both the blockchains share the type of consensus, the header of the genesis block, and the header of the last block. It is important to understand the information shared for the asset transfer. Thus, *CrossLedger* provides interoperability. □

4.3 SECURITY ANALYSIS

In this section, we provide a security analysis of our scheme and discuss how *CrossLedger* mitigates or eliminates some well-known attacks against the blockchain.

Lemma 6. *[Double-Spending Attack] The double-spending attack occurs in the cross-chain asset transfer when a* ***Case 1:*** *User U_A wants to spend the same asset twice to two different users, U_B and U_C.* ***Case 2:*** *Malicious user $\mathbb{U}_M$, wants to send a duplicate or conflicting asset,*

Proof. **[Solution] Case 1:** *CrossLedger* is secure against a double-spending attack because trust has been built between the asset forwarders of both blockchains, i.e., blockchain A and blockchain B, which require n verification from opposite blockchains. Here, n is the number of nodes in each blockchain. To get the n verification, the asset forwarder (AF) from each blockchain broadcasts the Asset Forwarder Verification (AFV) message to the opposite blockchain's network to create trust. To validate the legitimacy of the asset forwarder and build trust, a *TRS* is employed to sign the AFV message. As shown in *Algorithm* 8, $if(Verification_count(Msg_{AFV}) = n)$ then $Trust_Status.AF = Valid$. If the AF of blockchain A obtains n verification from the network of blockchain B, and the AF of blockchain B receives n verification from the network of blockchain A, trust is established. Since the trust is established between both forwarders after receiving n verification, the value of Pz is negligible. Therefore, it is clear that the *CrossLedger* is secure against the double-spending attack.

[Solution] Case 2: As we have described above, the definition of ownership of the asset is the leading cause of the prevention of the double-spending attack. Let the user $U_A \in \bigcup_{i=1}^{n} U$ and the asset which the user U_A owns is $Asset_Value_A \in \bigcup_{i=1}^{m} Asset$, for any malicious user, say $\mathbb{U}_M$, such that $\mathbb{U}_M \notin \bigcup_{i=1}^{n} U$ tries to spend asset $Asset_Value_A$ which does not belong to it, then it is not possible because An asset has exactly one owner at a time $Owner(U_A, Asset_Value_A)$ is true if and only if U_A owns $Asset_Value_A$.

□

Lemma 7. *[Liveness Attack] Using liveness attacks, Msg_{AFV} can be delayed as much as possible during the broadcast in the network of the opposite blockchain.*

Proof. The *CrossLedger* is secured from the liveness attack. Let's consider AF from blockchain A, which sends the AFV message to blockchain B's network, and AF of blockchain B, which sends the AFV message to blockchain A's network. To validate the legitimacy of the asset forwarder and build confidence, the AFV message is signed using a *TRS*. If AF of blockchain A gets n verification from the network of blockchain B in the threshold time frame, and AF of blockchain B receives n verification from the network of blockchain A within the timeline, trust is established. If $Verification_count(Msg_{AFV}) == n \&\& diff_t \leq thre_t$ from *Algorithm* 8 satisfied then the $Trust_Status$ of AF is declared as valid. The liveness attack can not affect AF because if the value of $diff_t$ is greater than $thre_t$, there is no trust established between both the AFs. The AF assumes that there has been a liveness attack between two blockchains if the value of $diff_t$ is higher than $thre_t$. Now, AF of one blockchain again forwards the AFV to other blockchains to establish the trust. □

Lemma 8. *[Sybil Attack] The Sybil attack is a major concern in almost every blockchain network. In cross-chain also, the Sybil attack is a big issue to be addressed. There are two cross-chain cases where this attack can take place.* ***Case 1:*** *When a malicious user creates multiple fake accounts and tries to transfer the assets which do not belong to them.* ***Case 2:*** *When a malicious user creates a fake node as asset forwarder AF. We have addressed both cases in the CrossLedger.*

Proof. **[Solution] Case 1:** We have given the ownership definition of any asset to its user. The *Algorithm* 9 also works according to the ownership of the legitimacy of the asset. The ownership of an asset de-

fines who (user) owns the asset. Let the user $(U_A, U_B) \in \bigcup_{i=1}^{n} U$ and the asset which the user U_A owns is $Asset_Value_A \in \bigcup_{i=1}^{m} Asset$. If a malicious user $\mathbb{U}_M$ wants to transfer some asset say $Asset_Value_A$ which belongs to the user U_A, is possible if and only if $Owner(\mathbb{U}_{\mathbb{M}}, Asset_Value_A)$ is true, where $\mathbb{U}_{\mathbb{M}} \notin \bigcup_{i=1}^{n} U$. However, this is not possible because only one user owns any asset at a time. For example, before the transfer of the asset, $Asset_Value_A$ belongs to the user U_A, and after the successful transfer and confirmation, the $Asset_Value_A$ belongs to the user U_B. However, it is mandatory that both the users, U_A and U_B, should be part of legitimate users of the blockchain network, i.e., $(U_A, U_B) \in \bigcup_{i=1}^{n} U$.

[Solution] Case 2: Another use case for the Sybil attack is that if any malicious node tries to become a AF in place of the legitimate node. To address this issue, the two algorithms of *CrossLedger*, *Algorithm* 7 and *Algorithm* 8, are used.

In the first step, the *Algorithm* 7 selects the AF as described in the algorithm. It is to be noted that the selected AF is the one node that has previously successfully added the legitimate block to the blockchain. This can be understood as follows. Let there are three nodes N_1, N_2, and N_3, who generate three blocks B_1, B_2, and B_3 respectively. The blocks are arranged as $B_1 \leftarrow B_2 \leftarrow B_3$. B_3 is the latest block generated by node N_3, and B_2 is the parent block of B_3 generated by the node N_2. Here, the N_2 is the CPM and N_3 is the CM. In this case, the N_2 is selected as the AF. We have selected the CPM as the AF due for two reasons. First, it reduces the load from the CM to do the task of asset transfer. Second, when a block is added successfully by the CPM, and no malicious activity is found, then it ensures the legitimacy of the N_2. Now let's suppose some malicious node, say $\mathbb{N}_M$ poses itself as the AF. However, before initiating the asset transfer, the AF_B of the destination blockchain first establishes the trust with AF of the source blockchain A, where $\mathbb{N}_M$ claims to the AF of it. To establish the trust, the AF_B broadcasts the Asset Forwarder Verification (AFV) message in blockchain A. A TRS is used to sign the AFV message to verify the authenticity of the asset forwarder and establish the trust. If the asset forwarder of blockchain B receives the $3/4^{th}$ verification from the network of blockchain A, then the trust is established. Due to this process, the malicious node $\mathbb{N}_M$, which poses itself as the AF, is identified and discarded by the AF of the destination blockchain B. This way, the *CrossLedger* is secured from the Sybil attack. □

Concluding Remark

This Chapter introduced *CrossLedger*, a novel protocol designed to facilitate seamless interoperability between distinct blockchain networks while ensuring robust security. This solution addresses the critical challenge of enabling cross-chain asset transfers and communication, which is essential for the future of decentralized applications operating across multiple blockchain ecosystems. The chapter demonstrated how *CrossLedger* effectively bridges the gap between different blockchains, laying the groundwork for a more interconnected and versatile blockchain environment. In the subsequent chapter, we shift our focus to the practical application of blockchain technology in smart cities, exploring how blockchain can be leveraged to enhance IoT, ITS, and smart grid systems through tailored solutions.