# CEV Framework: A Central Bank Digital Currency Evaluation and Verification Framework with Focus of Consensus Algorithms and Operating Models

Si Yuan JIN, Yong Xia*
HSBC Lab, HSBC Technology China
Guangzhou, Guangdong, China

***Abstract*—We propose a general framework (CEV Framework) for recommending and verifying technical solutions in central bank digital currency(CBDC) system, in which we build two sub-frameworks: an evaluation sub-framework that analyzes current problems about CBDC and provides solutions in terms of consensus algorithms and operating models, and a verification sub-framework that proves the feasibility of recommended solutions. Our framework provides a workflow and can be used as a reference to design CBDC systems based on related national economic and regulatory conditions. The working procedure to generate customized solutions is to split consensus algorithms into different components and analyze their impact on CBDC systems. Furthermore, we propose two operating models to cover operational aspects. Then we build a verification sub-framework to verify potential solutions through empirical experiments and formal logic proof. By iterating with our framework, CBDC designers can find a balanced point in designing a CBDC system. To the best of our knowledge, we are the first to propose a framework to recommend and verify CBDC related technical solutions.**

***Index Terms*—Central Bank Digital Currency, Evaluation Sub-framework, Consensus Algorithm, Operating Model, Verification Sub-Framework**

## I. Introduction

The recent development in cryptography and distributed ledger technology (DLT) has seen a new form of currency known as Central Bank Digital Currency (CBDC) [1]. CBDC is a digital representation of legal currency regulated by nations' monetary authority or central bank. There are many differences between jurisdictions in terms of national conditions and the focuses of regulators. So CBDC designers across different jurisdictions have varying approaches in designing a CBDC system. Based on previous research, we conclude several CBDC dimensions to measure a CBDC system and propose an evaluation sub-framework that provides holistic CBDC solutions covering these dimensions. Moreover, we build a verification sub-framework to verify the feasibility and rationality of proposed solutions and see if they match initial expectations. Finally, we integrate both sub-frameworks into one framework, called CEV Framework.

Consensus algorithms are vital components in a CBDC system. They directly impact many dimensions, including performance and privacy. Therefore, it is essential to evaluate consensus algorithms and undertake the related calibration properly. We analyze diverse consensus algorithms and build a theoretical framework, the evaluation sub-framework, that splits consensus algorithms into different components, such that the impacts of each of them on CBDC dimensions can be derived.

We also propose two operating models to cover operational aspects in which business secrecy is one of the critical criteria for CBDC systems. Most CBDC pilots have shown a tiered architecture, bringing many potential problems, especially business secrecy issues. Current technical solutions to keep business secrecy, such as homomorphic encryption [39], could not satisfy non-functional requirements, especially performance. Therefore, we propose new operating architectures to cover these aspects in our evaluation sub-framework, like helping institutions retain their business secrecy.

After proposing CBDC solutions, we then propose a verification sub-framework to guide CBDC designers to verify proposed solutions. The sub-framework needs CBDC designers to build a mathematical model for its choice and verify if it can meet initial expectations on diverse CBDC dimensions, like performance, security, privacy. By leveraging this framework, they can judge whether proposed consensus algorithms and operating models are consistent with their preferable evaluation dimensions.

In summary, our CEV framework includes an evaluation sub-framework to provide CBDC solutions and a verification sub-framework to prove the feasibility of proposed solutions. Depending on different national economic and regulatory conditions, CBDC designers can create various consensus algorithms to address specific CBDC needs.

The remainder of this paper is organized as follows. In Section II, we present the background of our research. Section III introduces our CBDC framework, including an evaluation sub-framework and a verification sub-framework. We mainly discuss the evaluation sub-framework in Section III and the verification sub-framework in Section IV. Section IV also gives an example of leveraging our framework to develop a solution and verify it. Finally, we conclude the paper in Section V.

*Corresponding Author: Yong Xia (yong.xia@hsbc.com)

## II. Background

### A. Central Bank Digital Currency

According to the research of the Bank for International Settlements (BIS) [4], more than 85% of central banks worldwide have already started research on CBDC. Since there are many differences regarding national economic and regulatory conditions between them, such as population and internet penetration, CBDC designers have diverse focuses to design their CBDC systems. For example, a country with a large population would pay more attention to the performance of its CBDC system. These different background conditions and the focus areas of regulators will lead to different CBDC designs.

Based on previous research [4]–[16], [38], we conclude following dimensions to cover most of CBDC-related considerations for CBDC designers and regulators. Note that there may be different definitions of the following categories, so we use specific sub-categories to define them.

*1) Performance:* In a country, millions, even billions of customers, may use CBDC in future. Therefore, performance is an essential criterion to handle millions of requests. In current CBDC projects around the world, blockchain has shown many benefits [27], [31]. In cross-border business, peer-to-peer payment could save liquidity and improve efficiency [33]. However, blockchain has been widely used in the wholesale CBDC rather than retail CBDC [40]. One key factor is that its weak scalability cannot meet the need for high performance in the CBDC system. Therefore, we conclude that we should consider the following features to measure performance.

1) User Scalability: the cost of adding a new customer to a CBDC system.
2) Network Scalability: the capability to handle larger transaction volumes per second(TPS), which is related to the time complexity of consensus algorithms.
3) Latency: the time to complete one transaction.

*2) Security:* Security in a distributed system involves various aspects, including cryptography, secure channels, key management, prevention of double-spending attacks, and so on [18]. Among them, the prevention of double-spending is one of the basic requirements in a CBDC system. Therefore, we would use double-spending prevention as an example in the verification sub-framework.

*3) Privacy:* Privacy is related to the user's identity and transaction information [32]. We divide it into two aspects, customer privacy and business secrecy. For customer privacy, it is related to the anonymity of transactions. For business secrecy, it is related to preventing data leakage to competitors, which is one of the essential design principles of CBDC. In BIS's report [43], it presents a two-tier CBDC model. This model will work if participating institutions are not commercial competitors with each other. If not, tier-two institutions would be worried about data leakage to tier-one institutions and may even give up CBDC services to protect business secrecy. In the CBDC system, we define business secrecy as safeguarding data of tier-two institutions from tier-one institutions.

*4) Others:* Other dimensions do not conflict with these CBDC dimensions, or there are already mature solutions in these areas. Examples include resilience, governance [19], functionality, interoperability and offline payments. For example, resilience can be related to many problems [32], such as hardware issues, power or network outages, or cloud service interruption. We consider the following features for resilience.

1) Fault-tolerance: the capability of tolerating accidents in a CBDc system, such as single-point failure.
2) Availability: the time or percentage that a CBDC system will operate satisfactorily in an ideal environment.

We believe these requirements can be met in the current financial system, or they can be solved independently, so we do not discuss them in any detail. In future, there could be more dimensions involved in our CEV framework if needed.

### B. Consensus algorithm

With blockchain development, new research topics appear one after another, especially in the performance part. The performance of blockchain, especially bitcoin [20], cannot meet today's commercial needs. To solve this problem, many blockchain platforms spring up one after another [22]–[25]. Besides performance, other dimensions, like resilience, security and privacy, are also undergoing research. Consensus algorithms play a key role in many dimensions. Fabric [22] once used Practical Byzantine Fault Tolerance (PBFT) [21] which provides fault tolerance while sacrificing part of performance. Corda blockchain protocol aims to satisfy finance and regulatory requirements [23]. Transactions in the Corda platform are recorded only by participants rather than the entire network, which provides high performance and protects privacy to some extent. To fully consider consensus algorithms, we use our evaluation sub-framework to analyze most of the current consensus algorithms.

### C. Tiered Model

BIS's paper [13] mentions a tiered CBDC model. Until now, a tiered CBDC model has been selected in most CBDC projects, including China's E-CNY [30], Sweden's E-Krona [?], [41], [42]. There are several strong reasons to follow a tiered model in CBDC designs. Firstly, a tiered model can maintain the system stability of the current banking system. After many years of operation, the tiered model shows financial stability and has advantages in functionality and facilities. Furthermore, a tiered model can accelerate market competition and improve banking and payment services.

## III. Evaluation Sub-framework

The CEV framework includes two sub-frameworks: an evaluation sub-framework that provides consensus algorithms and operating models and a verification sub-framework that proves the feasibility and rationality of recommended solutions. We will discuss the evaluation sub-framework in detail in this section.

Figure 1 shows the working procedures of the CEV framework. First, CBDC designers can determine preferable CBDC dimensions based on their economic and regulatory conditions. Then they can determine consensus algorithms and operating models according to the evaluation sub-framework. After that, CBDC designers can build a theoretical model for their choices and carry out experiments and proofs to verify original CBDC dimensions. Finally, if they are not satisfied with the verified result, they can go back to adjust their expectations on CBDC dimensions, leverage the evaluation sub-framework to update their solutions, and verify proposed solutions again.

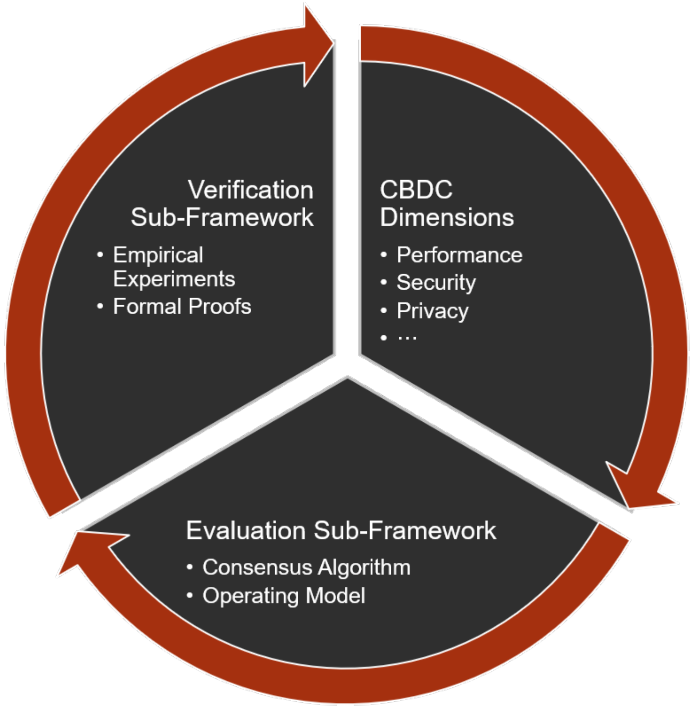

Fig. 1. **A closed-loop workflow of CEV Framework for CBDC designers.**

We mainly discuss the evaluation sub-framework in Section III. The evaluation sub-framework includes two parts: consensus algorithm part which splits consensus algorithms into different components and analyses their impacts on CBDC systems, operating model part which provides two options to overcome business secrecy issues.

*1) Consensus algorithm:* Consensus algorithms play a significant role in CBDC systems. It has direct impacts on many dimensions mentioned in Section II. We split them into different components to better analyze consensus algorithms and their impacts on CBDC systems. Then we can estimate the impacts of individuals. By combining different components into one algorithm and analyzing its effect on CBDC dimensions, we can better propose CBDC-related consensus algorithms to adapt national economic and regulatory conditions.

figure 2 introduces components of consensus algorithms. In a CBDC consensus network, a client will send a request to the system, and the system will process it until reaching an agreement inside the network. The following steps describe details about this process:

1) Leader Election / Proposer: a network requires one representative to lead the bookkeeping process before a client sends a transaction request.
   a) Voting: a leader is elected via a voting mechanism. There are many extensive options for the voting mechanism, such as defining the percentage of votes needed to become a leader. RAFT [26] adopts a voting mechanism with election timeout to determine the network's leader.
   b) Predetermination: a leader is predetermined in this network. For example, leaders would be the notary node in Corda blockchain platform [23], which is determined before network deployment.
   c) Round-robin: a leader is chosen by an arrangement in a group equally in some rational order. PBFT [21] uses this approach to choose the primary (leader).
   d) Proposer: There could be no leader in the network. For example, in some public blockchain systems, a proposer collects transactions from users and proposes them to the network via Proof of Work [20], Proof of Stake [35], etc.
2) Client - Request: a client would submit its payment request to the network in a transaction.
   a) To one: a client sends the request to one node. A node is a connection point in a communication network. In some algorithms, like RAFT, if no response from the leader is received, the client sends the request to another node in the network. We classify this situation as the "To one" option. Furthermore, we consider node sharding [28], [29] to improve system performance. Sharding means multiple nodes process transactions in parallel and interact with each other in a specific manner. This may improve the system's capability to handle high concurrent transactions.
   b) To all: a client sends the request to all nodes in the network to ensure the request is accepted in time.
3) Leader / Proposer - Pre-prepare: the leader node or proposer would process the request locally after receiving it.
   a) Proof of X: if "proposer" is chosen in the first step, the proposer would leverage one of Proof of X, such as Proof of Work [20], to publish transactions.
   b) Verification: A leader would verify proposed transactions and send them to other nodes.
   c) Inter-communication & Verification: a leader would first communicate with other nodes and filter transactions into the next phase. Filtration can potentially reduce the number of illegal transactions in the early phase, which can help increase the system's security.
4) Fixed / Dynamic Validators: validators are other nodes than the leader node. In this step, some consensus algorithms require all nodes to carry out some operations while others just need selected or random dynamic nodes. For example, traditional RAFT [26] and

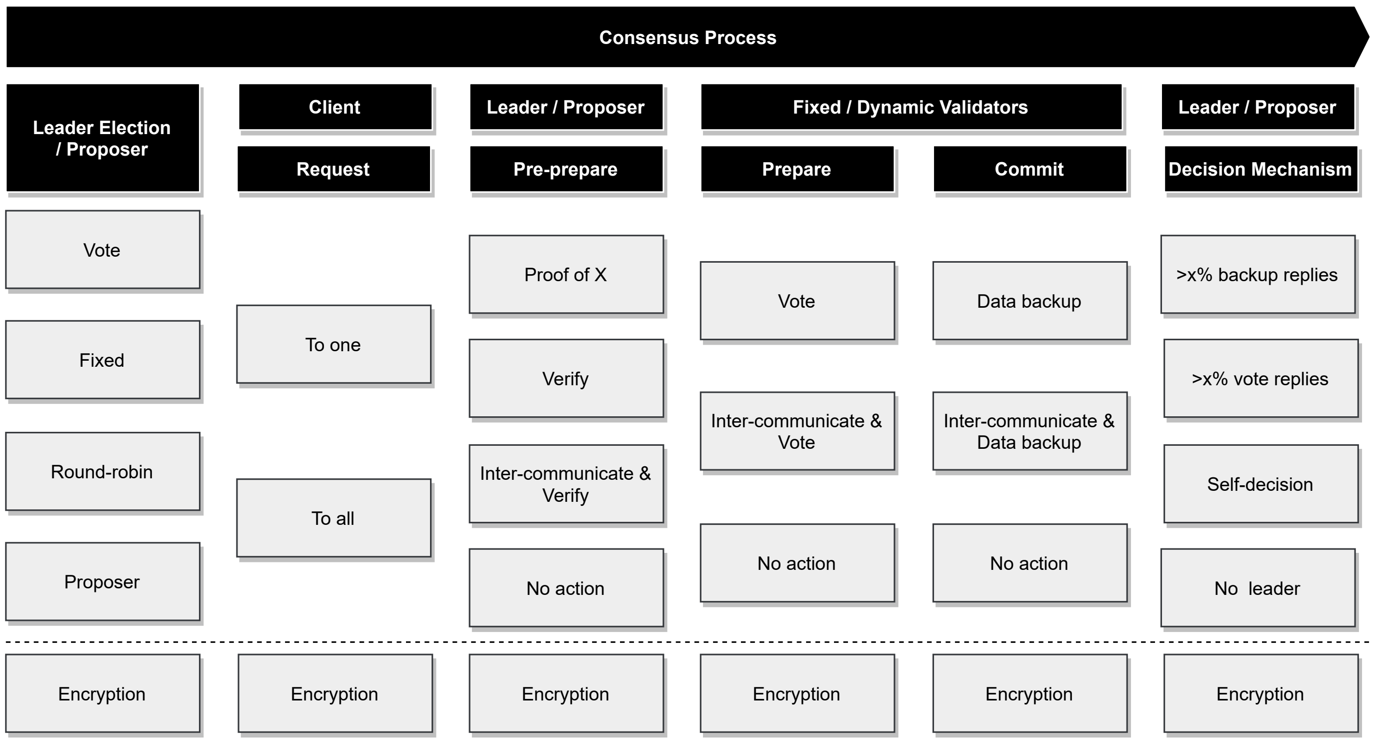

Fig. 2. **Consensus Algorithm Process Map. We break consensus algorithms into several parts to better analyze each one.**

PBFT [21] need all nodes to participate data backup, while IBFT [34] adopts a dynamic set of validators to deal with transactions. A dynamic set of validators can provide a higher performance due to a more flexible number of participants. In contrast, a fixed set of validators would be easier to implement and more secure.

5) Validators - Prepare: validators can vote and communicate with others to verify the request from the leader node.
   a) Voting: validators would vote for the request.
   b) Inter-communication & Voting: validators would vote for the request after inter-communication between each other. PBFT [21] adopts inter-communication between nodes to prevent malicious behaviors.
   c) No action: No action is taken in this step. For example, validators (called followers) in RAFT [26] would not vote or communicate.
6) Validators - Commit: commit means how validators record transactions in their database. They could directly undertake data backup, like RAFT [26], or communicate with other nodes, like PBFT [21] and then determine whether to record the proposed request.
   a) Data backup: validators would make a backup in its local database without other operations.
   b) Inter-communication & Data backup: validators would communicate with each other before backup.
7) Leader / Proposer - Decision Mechanism: decision mechanism describes the conditions of a success notice sent to the client that the transaction has finalized.
   a) x% backup replies: the transaction would be finalized if the leader node receives more than x% backup replies. For example, in RAFT [26], the leader node needs to receive 50% replies before responding to its client.
   b) x% vote replies: the transaction would be finalized if the leader node receives more than x% vote replies. If there is no action in the "commit" step, there could be possible that the validator node only collects votes.
   c) Self-decision: a leader node decides the transaction by itself. For example, the notary node in Corda [23] verifies proposed transactions by itself.
8) Encryption: encryption can be applied in every step to secure transmitted information. It is independent of previous options. In most algorithms, consensus algorithms are more related to achieving consistency in a network and do not cover encryption. However, our consideration of consensus algorithms is more relevant to CBDC, where encryption plays an important role, so we include encryption in this part.

Note that each component has many possibilities and extensions. Our evaluation framework would be more like a guide to CBDC designers to consider related factors. Besides, there are also some constraints between different components. For example, based on our knowledge, choosing "proof of X" in the third step must choose "proposer" in the first step. Therefore, we need to verify to ensure that proposed consensus algorithms are valid, which will be done in the verification sub-framework.

We have referenced PBFT and RAFT several times. Here we describe two consensus algorithms by our Consensus Process Map. RAFT [26] (figure 3) starts from voting and election timeout mechanism in the leader election. Then a client sends a request to the leader in the network. After the leader's Verification, validators would make a backup and respond to the leader node. After receiving more than half of the notices, this transaction would be regarded as valid. PBFT [21] (figure 4) selects a leader in a round-robin manner. Suppose a client sends a request to all nodes in the network, the leader node would undertake simple Verification and publish the request. Then validators would communicate to ensure more than 67% of nodes pass Verification. Then a new round of communication between them would ensure more than 67% of them undertake a data backup. Finally, the client waits for 33% replies from validators to ensure no malicious behaviours.

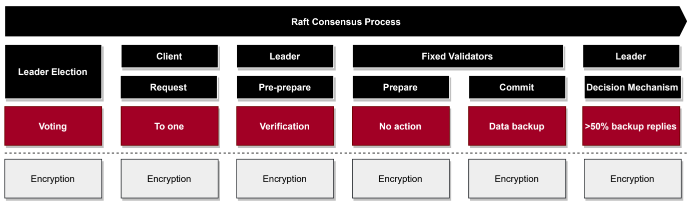

Fig. 3. **Consensus Process Map of RAFT**

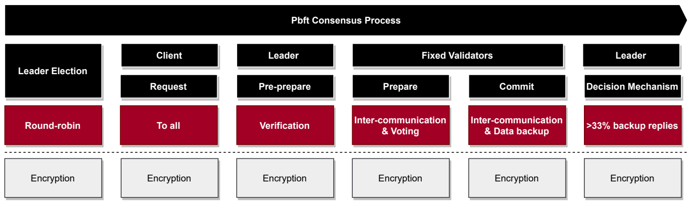

Fig. 4. **Consensus Process Map of PBFT**

Due to diverse national economic and regulatory conditions, CBDC designers have different requirements for their CBDC systems. Therefore, to make the CEV framework widely applicable, we further analyze connections between consensus algorithm components and CBDC dimensions. We now conclude a table according to our experience on industrial CBDC research and work. Table I shows the impact of the above components on mentioned dimensions. We measure the impact through categories of High, Medium and Low. High means the module is helpful to the dimension. Medium means the module has no impact or relative medium impact on the dimension. Low means the module has a relatively bad influence on the dimension. TBA indicates that the impact needs to be further analyzed. For example, the "To one" option can provide higher user scalability and network scalability due to the sharding method. CBDC designers can better choose consensus components according to their national requirements by this table.

This table needs further refinement in the future via the CEV framework workflow. Our objective, for now, is to build a framework for future research. Based on this framework, CBDC designers could choose different components individually according to their focus areas and analyze different combinations until they find a suitable consensus algorithm.

*2) Operating Model:* We propose several operating models to cover operational aspects, especially business secrecy, that can not currently be solved from a technical standpoint alone. According to the BIS's paper [43], a two-tier CBDC model is proposed where tier-one institutions are directly connected to the central bank (tier-one), and the tier-two part includes household and business. If tier-two institutions and tier-one institutions are not competitors, this model can mitigate business secrecy-related concerns. For example, if all tier-one institutions are banks, other banks in tier-two would be worried that tier-one banks monopolize their customer data.

However, it is nearly impossible in a tiered model to let all commercial institutions become tier-one ones and responsible for CBDC distribution and circulation because of a potential single point (central bank) failure and performance bottleneck. Distribution means that an institution connected to the central bank helps issue CBDC and manages CBDC authentication work. Tier-one institutions take the responsibility of distribution in a two-tier CBDC model.

Circulation means that an institution provides CBDC-related transfer services. General commercial institutions can carry out circulation in order to improve service coverage. However, if tier-two institutions provide CBDC services, they have to connect to tier-one institutions and ask them to provide authentication services. Tier-two institutions are mostly competitors with tier-one institutions, and they are reluctant to provide transaction and customer information to their competitors, making two-tier architecture hard to implement in a CBDC system.

Overall, we found that two problems in the current two-tier CBDC model:

1) It is not possible that every institution is a tier-one institution.
2) If tier-two institutions want to provide CBDC related services to their customers, they would have to connect to tier-one institutions. However, they are commercial competitors in most cases, and tier-two institutions would be worried about customer data leakage.

A three-tier CBDC model (figure 5) can better describe these two problems in which commercial institutions would be divided into two tiers. We regard higher-tier institutions as privileged institutions and lower-tier institutions as non-privileged ones. The tier-one institutions in the two-tier model are privileged in the three-tier model. The three-tier model describes how non-privileged institutions provide CBDC services to their customers. Due to privileged institutions operating the ledgers, non-privileged institutions must provide transaction information to privileged ones for bookkeeping, which means transactions from non-privileged institutions' customers are regarded as valid only

TABLE I
**IMPACT OF INDIVIDUAL CONSENSUS COMPONENTS. THE FIRST COLUMN IS CONSENSUS COMPONENTS. THE FIRST ROW IS CBDC-RELATED DIMENSIONS.**

| Modules | | User Scalability | Network Scalability | Latency | Fault-tolerance | Account-ability | Security | Customer Data Protection | Business Secrecy |
|---|---|---|---|---|---|---|---|---|---|
| Leader Election / Proposer | Voting | TBA | Medium | Medium | TBA | Low | High | TBA | TBA |
| | Predetermination | TBA | High | High | TBA | High | TBA | TBA | High |
| | Round-robin | TBA | High | High | TBA | TBA | TBA | Low | Medium |
| | Proposer | High | High | Low | TBA | Low | TBA | TBA | TBA |
| | Encryption | TBA | TBA | Low | TBA | Low | TBA | TBA | TBA |
| Client - Request | To one / Sharding | High | High | Medium | TBA | TBA | TBA | Medium | TBA |
| | To all | Low | Medium | High | TBA | TBA | TBA | Low | TBA |
| | Encryption | TBA | TBA | Low | TBA | TBA | TBA | TBA | TBA |
| Pre-Prepare | Verification | High | High | High | TBA | TBA | TBA | TBA | TBA |
| | Inter-communication & Filter | Low | Low | Low | TBA | TBA | High | Low | Low |
| | Encryption | TBA | TBA | Low | TBA | High | TBA | High | High |
| Validators | Predetermination | TBA | Low | Low | High | High | High | TBA | TBA |
| | Dynamic Selected | TBA | High | High | Medium | Medium | Medium | TBA | TBA |
| Prepare | Voting | Medium | Medium | Medium | TBA | TBA | TBA | Low | TBA |
| | Inter-communication & Voting | Low | Low | Low | TBA | TBA | TBA | Low | Low |
| | Encryption | TBA | TBA | Low | TBA | Low | TBA | TBA | High |
| Commit | Data Backup | Medium | Medium | Medium | High | TBA | TBA | TBA | TBA |
| | Inter-communication & Data Backup | Low | Low | Low | High | TBA | TBA | TBA | TBA |
| | Encryption | TBA | TBA | Low | TBA | TBA | TBA | TBA | TBA |
| Decision Mechanism | >x% Vote Replies | Medium | TBA | TBA | High | Low | High | TBA | Low |
| | >x% Backup Replies | Meidum | TBA | TBA | High | High | Medium | TBA | Medium |
| | Self-decision | High | High | TBA | Low | High | Low | TBA | High |
| | Proof of X | High | Low | TBA | High | Low | High | TBA | Medium |

when recorded in privileged institutions' ledgers. Non-privileged institutions are particularly reluctant to provide their customer data to privileged ones because they are competitors with interest conflict. Therefore, their customer data should not be shared with their competitors (privileged institutions).

For commercial institutions, they can give up providing CBDC-related services or become non-privileged institutions and connect to privileged ones in which they are worried about business secrecy issues.

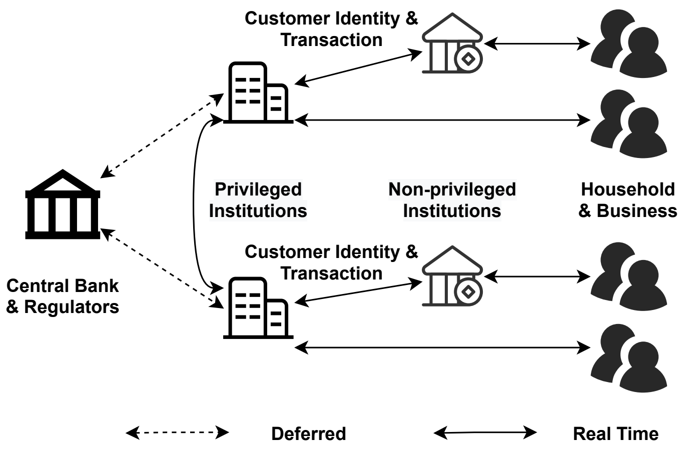
Customer Identity & Transaction
Privileged Institutions
Non-privileged Institutions
Household & Business
Customer Identity & Transaction
Central Bank & Regulators
Deferred
Real Time

Fig. 5. **Three-tier CBDC model**

To safeguard non-privileged institutions from data monopoly, we propose several operating models:

1) Use dynamic virtual addresses to keep the identities of participants secret from privileged institutions;
2) Set up an independent operating organization that has no conflict of interest;

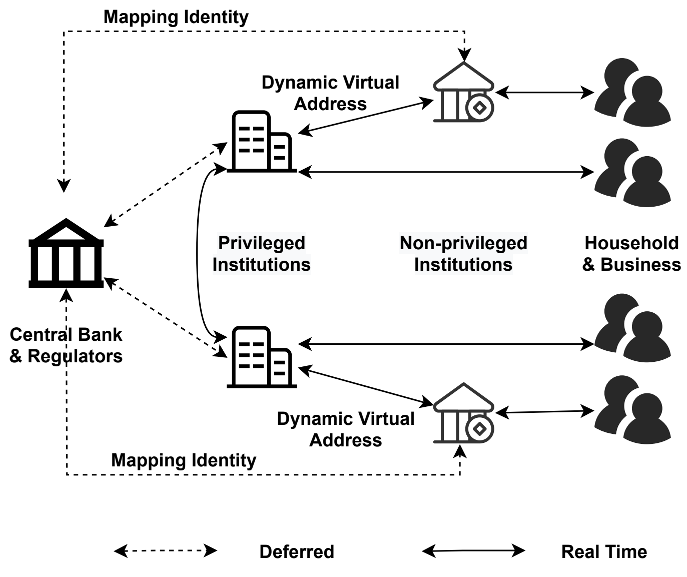

Fig. 6. **Operating Model Option One leverages dynamic virtual address to avoid privileged institutions from knowing the real identities of customers.**

For option one (figure 6), non-privileged institutions would create virtual addresses for their customers in the ledger of privileged ones who would not know identity information. Note that privacy includes identity and transaction information. Virtual addresses ensure that privileged institutions do not know the payee and payer of each transaction. In this model, non-privileged institutions would provide a mapping table between virtual addresses and real identities to regulators. Only regulators and non-privileged institutions can know the mapping relation of virtual addresses to real identities. One benefit of this option is that non-privileged institutions only need to inform the central bank of the identity mapping relationship, then the central bank could get all transaction information by combining identity mapping and ledger transaction information.

However, privileged institutions can infer virtual addresses by analyzing token flow even though they only know transaction information. To further protect the business secrecy of non-privileged institutions, we can make virtual addresses dynamic, which means new virtual addresses are created to collect changes in transactions. This technical solution can prevent privileged institutions from further accessing non-privileged institutions' critical customer data.

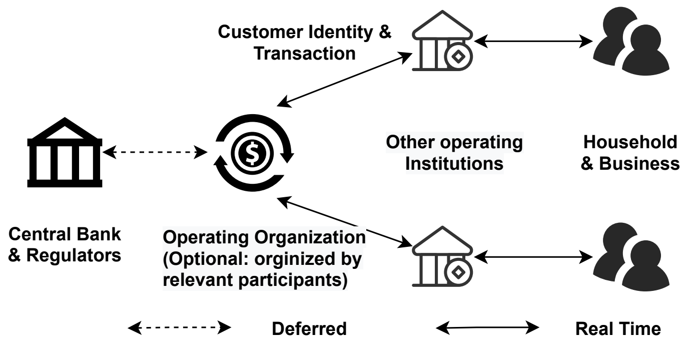
Customer Identity & Transaction
Other operating Institutions
Household & Business
Central Bank & Regulators
Operating Organization (Optional: orginized by relevant participants)
Deferred
Real Time

Fig. 7. **Operating Model Option Two builds an operating organization rather than privileged institutions to operate the CBDC system without interest conflict.**

For option two (figure 7), the operating organization could be operated by a third-party institution to ensure no competition with other tier-one and tier-two. In this model, an operating organization could know the transaction data of other operating institutions without business secrecy issues. However, the business model of operating organizations needs to be further created to ensure its stable operation. For example, all participating institutions could invest and organize the operating organizations, protecting their interests and supporting rational CBDC operation.

Both options can be applied in a CBDC system to ensure business secrecy and a fair market. CBDC designers can choose either of them based on their preference.

## IV. Verification Sub-Framework

By the evaluation sub-framework, CBDC designers can develop a customized solution. Then the verification sub-framework works to ensure the proposed solution's feasibility and rationality. Figure 8 shows how CBDC designers can verify proposed solutions. The procedures to apply verification sub-framework:

1) Build a model based on proposed solutions, including state machine, ledger, transaction. A theoretical model is essential for further Verification. The consensus algorithm and operating model have been developed in the evaluation sub-framework, so we need to further build related state machines and ledger to describe the proposed solution.
2) Translate this model into mathematical languages, including notations, definitions. Math is a perfect tool to provide further rational Verification.
3) Empirical experiments are used to verify dimensions that need a real test, like performance.
4) Formal proofs are used to prove the rationality of proposed solutions, like preventing double-spending. We would follow the rules built in previous steps and prove related theories with formal logic proof.

Next, we will use the verification sub-framework to analyze potential solutions proposed from the evaluation framework and build related models for Verification.

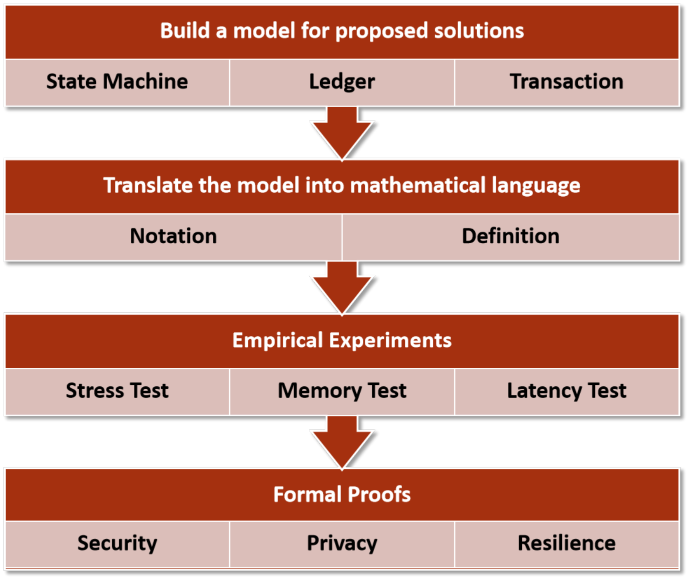

Fig. 8. **Verification Sub-Framework**

The choice of operating models would influence the design of consensus algorithms. So we first discuss the overall operating architecture. Figure 9 shows potential consensus networks in our proposed operating model (option one). We would follow this model to verify CBDC dimensions. There are wholesale consensus networks and retail consensus networks in the figure. The wholesale consensus network involves central banks and privileged institutions responsible for transactions involving different privileged institutions. For this kind of transaction, there would be interactions between the wholesale consensus network and retail consensus networks. On the other hand, retail consensus networks involve retail clients and service providers. Transactions between clients in the same privileged institutions would be finished inside retail consensus networks.

In the wholesale consensus network, there could be many possible implementations. For example, if the central bank does not want to record retail transactions, it can control the wholesale balance of issued CBDC to different privileged institutions. The central bank is only responsible for issuance and redemption transactions by this method.

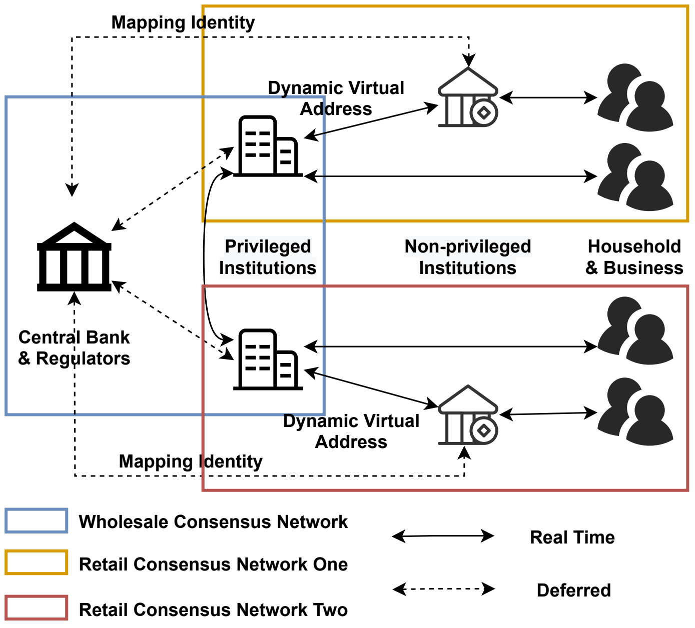

Fig. 9. **Consensus Networks**

If any issue exists in retail transfer transactions, corresponding privileged institutions should be accountable. On the other side, if the central bank wants to regulate every retail transaction, privileged institutions can provide detailed transaction information asynchronously in this model. Asynchronously means transactions do not need the central bank to participant in the process of consensus in real-time, and this transaction information can be provided to the central bank after transactions are finished.

On the other hand, there could be different consensus algorithms in retail consensus networks. Consensus algorithms in the retail consensus network one and two work to protect retail users at different levels. For example, privileged institutions can involve other corporations to form a blockchain network to ensure each transaction recorded in the ledger is verified by each participant in the network. However, this method would influence performance a lot. Alternatively, privileged institutions can determine every transaction by themselves, ensuring high performance, but it may bring potential security issues. Therefore, the implemented consensus algorithms in different consensus networks could be different.

Next, we provide an example by leveraging the framework. Assume a country with a large population and well-developed technology and communication. The CBDC designers focus on privacy and performance, especially network scalability, latency, business secrecy. Assume they choose operating model option one and build consensus algorithms based on it.

There would be different consensus algorithms in operating model option one in different consensus networks. For wholesale consensus network, we propose a new consensus algorithm in figure 10. For retail consensus networks one and two, we propose a new consensus algorithm in figure 11.

The recommended consensus algorithm in wholesale consensus network (consensus algorithm one) works as following procedures:

1) Leader nodes are predetermined in the algorithm that the central bank would run.
2) Clients send cross-shard transactions (defined later) to its sharded leader node, which would resend it to the leader node (central bank) in the wholesale network.
3) The leader node would verify the transaction and finish the related issuance and redemption transactions for different sub-networks.

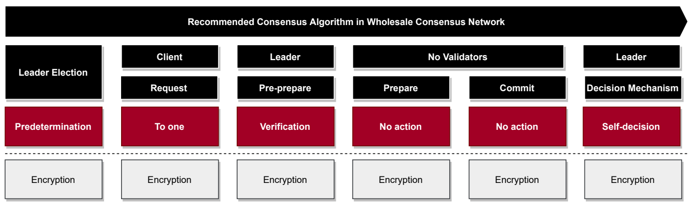

Fig. 10. **Recommended Consensus Algorithm in the Wholesale Consensus Network**

The recommended consensus algorithm in retail consensus networks (consensus algorithm two) works as following procedures:

1) Leader nodes are predetermined in this algorithm that privileged institutions run.
2) Clients send transactions to its sharded leader node.
3) The leader node encrypts transactions before sending to validators in the network (possible competitors).
4) Dynamic set of validators would be chosen to make an encrypted data backup.
5) The Leader node would send a success notice to the client after receiving 50 per cent of notices from dynamic validators.

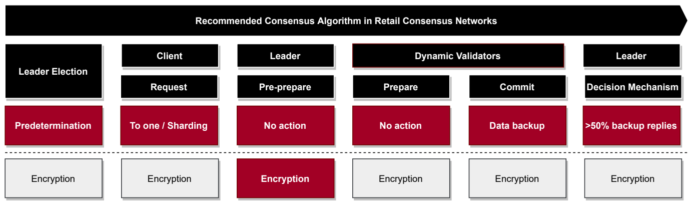

Fig. 11. **Recommended Consensus Algorithm in the Retail Consensus Networks**

Finally, we would leverage the verification sub-framework to show how we can verify the proposed solution's performance, security, and privacy.

## *A. Model*

We build a theoretical model to describe the proposed solution above, including state machine, ledger and transaction types in a token-based CBDC system. Furthermore, we use 'regulated cryptocurrency' to replace 'CBDC' in some parts because regulated cryptocurrency includes CBDC and stablecoin, which means the CEV framework can also be used to design a stablecoin system.

Different components in the consensus process map can be used flexibly. For example, recommended consensus algorithm two in figure 11 chooses "To one / Sharding" in the "client-request" step. This choice brings many possibilities to system design and implementation.

There are two types of transactions in the CBDC system: cross-shard transaction, which describes a transaction with two ledgers involved, single-shard transaction, which describes a transaction within one ledger. Figure 12 shows a cross-shard transaction on the left side. We discussed three-tier operating models in Section III, in which privileged institutions can also directly provide services to end-users. Therefore, we only show two-tier in this figure. In a CBDC system, cross-shard transactions would influence the system's performance compared to single-shard transactions because cross-shard transaction needs the currency issuer (Central Bank) to redeem a token in one ledger and issue a new token in another ledger. Moreover, with the increasing number of operating institutions, it would be frequent that cross-shard transactions happen in an account-based sharding system.

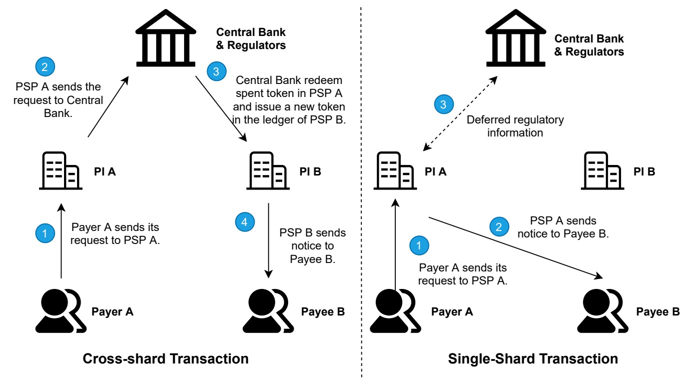
Central Bank & Regulators
2
PSP A sends the request to Central Bank.
3
Central Bank redeem spent token in PSP A and issue a new token in the ledger of PSP B.
PI A
PI B
1
Payer A sends its request to PSP A.
4
PSP B sends notice to Payee B.
Payer A
Payee B
Cross-shard Transaction
Central Bank & Regulators
3
Deferred regulatory information
PI A
PI B
2
PSP A sends notice to Payee B.
1
Payer A sends its request to PSP A.
Payer A
Payee B
Single-Shard Transaction

Fig. 12. **Cross-Shard Transaction vs Single-shard transaction. PI refers to privileged institution.**

Sharding is a method used to improve performance. The account-based sharding method divides users by accounts. Traditionally, privileged institutions have different wallets and divide users by their accounts. Specifically, PI (privileged institution) A's customers will come to PI A when using CBDC because it created its account from PI A. If the customer transfers its money to PI B's customers, a cross-shard transaction happens.

On the contrary, we may divide users by tokens. If PI A issues a token to one retail customer, the customer will come to PI to initiate CBDC related services because of the token he used rather than its account. Figure 13 shows token data structure, in which a string of the operator could be used to determine its service provider. In a token-based sharding, users must contact their token service provider. Therefore, users can transfer the token to everyone without a cross-shard transaction. However, token-based sharding needs the central bank with operating institutions to provide a uniform interface to distribute transactions based on tokens. Because if a customer use PI A's token, he needs to come to PI A to transfer the token. If he comes to PI B's interface, he should be distributed to PI A based on the uniform interface.

Another important thing is that the central bank should provide uniform wallet standards to privileged institutions. For example, if the transaction receiver is not PI A's customer before, there should be a wallet address to record the token from it. In this case, if PI B provides KYC and onboarding services to its customers, it does not expect to see data leakage of its customers to PI A. So virtual addresses could help design wallet addresses. Figure 13 shows that a virtual address for its holder or wallet. Note that token-based sharding is only a new direction to explore. It may be not an excellent solution to solve all related pain points.

| | | | | |
|---|---|---|---|---|
| .... | PI A | 100 | asdadc3423sdad... | .... |
| | **Operator** | **Amount** | **Holder** | |

Fig. 13. **Token Data Structure**

Overall, we try to leverage a token-based sharding system to reduce the number of cross-shard transactions and verify related features.

Besides, in consensus algorithm two, encryption in the third step is used to protect business secrecy from validators, including non-privileged institutions. Therefore, validators in the algorithm would only make a backup of encrypted data, which will be used for tampering with proof.

Next, we build a theoretical model for the recommended algorithm and operating model. Figure 14 shows ledger state machine. The central bank and privileged institutions would operate the state machine together. This model provides a formal description of finite state machine M = (S, V, t). The state machine can describe any state $s \in S$ for every moment. It can read input token $\tau \in V$ and proceed to the next state by different transitions t(s, $\tau$). There are two types of transactions. Single-shard transactions will only need consensus in the retail consensus network, while cross-shard transactions need the central bank participant. Leaders are privileged institutions in the retail consensus network. The central bank is the leader node in the wholesale consensus network.

Assuming an initial machine state $s_0$, a token-based sharding system would distribute transactions to different leaders. The central bank institutions could provide a uniform system interface to the public to handle transactions and process them based on token-based sharding. In a single-shard transaction, the shard leader checks transaction signatures and input tokens $\tau_0$. After verifying the signature, the tokens become locked. If involved tokens have not previously been locked, the output becomes $\tau_{locked}$ and the machine moves to the locked state $s_1$. Otherwise, it will exit ( a rolled-back transaction to the initial state). Next, the leader verifies $\tau_{locked}$ available in its ledger. If $\tau_{locked}$ are available, the output token will be $\tau_{verified}$, and the machine moves to the verified state $s_2$. Otherwise, it will exit. Finally, the leader writes the transaction with inputs

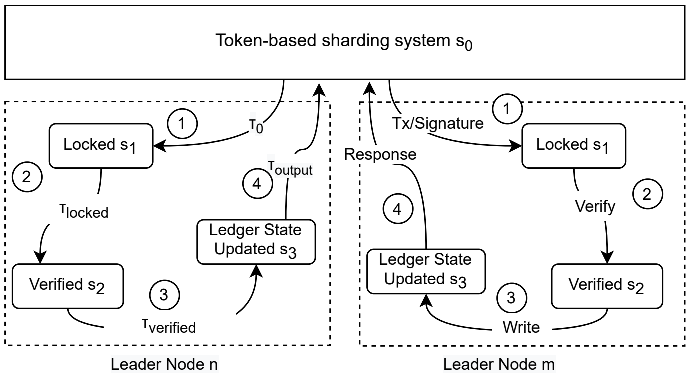

Fig. 14. **State Machine describes how a transaction is being executed, including token's state on the left side and operations on the right side. If choosing sharding in the consensus process map, there could be many leader nodes at the same time.)**

TABLE II
**TABLE OF NOTATION**

| Notation | Meaning |
|---|---|
| $x, y, z$ | time |
| $xRy$ | x before y |
| $\tau$ | Token |
| $V$ | Token Set |
| $p(\tau)$ | $\tau$ has been recorded by one leader |
| $Tx(\tau, x)$ | Transaction with input $\tau$ at time x |
| $H(\tau, x)$ | $\tau$ has been spent before x |
| $F(\tau, x)$ | $\tau$ can be spent after $x$ |
| $r(\tau)$ | received token in a transaction using $\tau$ |
| $c(\tau)$ | change token in a transaction using $\tau$ |
| $f(\tau)$ | received token in a cross-shard transaction using $\tau$ |
| $d_G^+(\tau)$ | The out-degree of vertex v |
| $d_G^+(\tau, x)$ | The out-degree of vertex v at time x |
| H | Leaders execute transactions by state machine flow |
| EE | Existential Elimination |
| AE | And Elimination |
| AI | And Introduction |
| UE | Universal Elimination |
| UI | Universal Introduction |
| IE | Implication Elimination |
| II | Implication Introduction |
| BE | Biconditional Elimination |
| Ind | Induction |
| AS | Assumption |

and outputs. If successful, the output will be $\tau_{output}$, and the machine moves into the state $s_3$. Otherwise, it will exit. These steps ensure the token is recorded on the ledger before noticing clients.

Note that we do not cover validators' operations here, like data backup. We will discuss it in Section C.

Figure 15 shows data model of leaders' ledgers. Transaction records are subject to the leader's ledger. Once a leader updates its ledger, the transaction becomes legal and immutable. If traders want to revert the transaction, they must initiate a new transaction to return the token.

**Definition 1.** ***(Ledger State)*** *The ledger state of the regulated cryptocurrency is defined as a directed graph D=<V(D), E(D), φ> that the elements of V(D) are vertices (tokens) and the elements of E(D) are edges (token flow) in the model. φ is ordered mapping from token set V to token flow set E.*

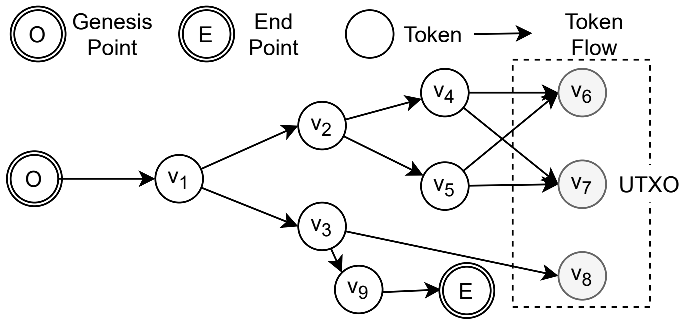

Fig. 15. **Ledger State. v are vertices (tokens), belonging to V, edge represents token flow. On the right side, there are some tokens that no edge coming from, which we call unspent token set (UTXO).**

**Definition 2.** ***(UTXO)*** $UTXO = \{\tau | \tau \in V(D) \wedge d_G^+(\tau) = 0\}$. *UTXO is an unspent transaction token set. In the model, an unspent token means there is no edge coming from it (the out-degree of it is 0).*

**Definition 3.** ***(Transaction Graph)*** *A transaction graph is a directed graph TD=<V(TD), E(TD), φ>. In a transaction graph, the in-degree of an input token is 0 and the out-degree of an output token is 0. Leader's ledger will be updated when a transaction is finished by the state machine (∀x. ∀ τ.(Tx(τ,x) ⇒ D = D + TD)).*

Figure 16 shows all types of transaction graph. The Initial Issuance Transaction can only be initiated by the currency issuer (Central Bank), who can generate a new token from the genesis point. The Final Redemption Transaction has to be checked by the central bank (transaction receiver). Besides, our model needs the central bank to carry out real-time Initial Issuance Transactions and Final Redemption Transactions. Other types of transactions can be regulated in a deferred manner for central banks, which reduces the central bank's performance stress. A valid transaction in in figure 16 would produce new tokens.

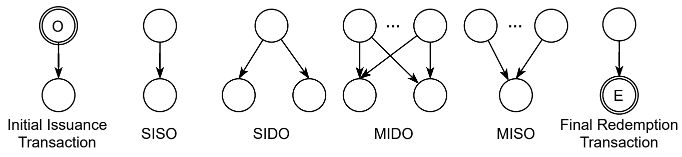

Fig. 16. **Different types of transaction graphs. SISO refers to a single-input single-output transaction. SIDO refers to a single-input double-output transaction. MIDO refers to a multi-input double-output transaction. MISO refers to a multi-input single-output transaction.**

We can use the following mathematical expressions to show token flow. Here we add an assumption that the leader nodes are non-faulty (H) and would follow the model procedures. Faulty nodes may behave arbitrarily and be vulnerable to inside and outside attacks. With non-faulty nodes, we can ensure tokens are recorded in the ledger by every transaction:

1) $\forall \tau.(H \wedge p(\tau) \Rightarrow p(r(\tau)) \wedge p(c(\tau)))$
2) $\forall \tau.(H \wedge p(\tau) \Rightarrow p(f(\tau)))$

The state machine ensures that every transaction is valid. $\forall\tau.(H \wedge p(\tau) \Rightarrow p(r(\tau))$ means that if input $\tau$ has been recorded in the ledger, a valid transaction graph using $\tau$ as inputs and received token $r(\tau)$ and change token $c(\tau)$ as outputs will be added to the ledger by a non-faulty leader(H). If the change is 0, then $c(\tau) = null$ and p(c($\tau$)) means no token recorded. Note that the expressions above show the token flow rather than the transaction.

$\forall\tau.(H \wedge p(\tau) \Rightarrow p(f(\tau)))$ means non-faulty leaders(H) would ensure that output tokens in a cross-shard transaction would be recorded in another shard's ledger. Cross-shard transactions would not happen in our model if we split one transaction into several concurrent sub-transactions. However, cross-shard transactions would match some business scenarios more. For example, CBDC users may pay tokens in different ledgers and achieve an atomic transaction. Alternatively, CBDC designers want to control the number of tokens and consolidate tokens from different ledgers to one new token.

Figure 17 shows a client has different tokens allocated in three different shards and uses them to initiate a transaction. The transaction first turns the input tokens to the endpoint via the Final Redemption Transaction and issues new tokens in the new shard via the Initial Issuance Transaction. In a CBDC system, the central bank is responsible for issuing and redeeming tokens, regulating both transactions and ensuring that output tokens are recorded in the new shard.

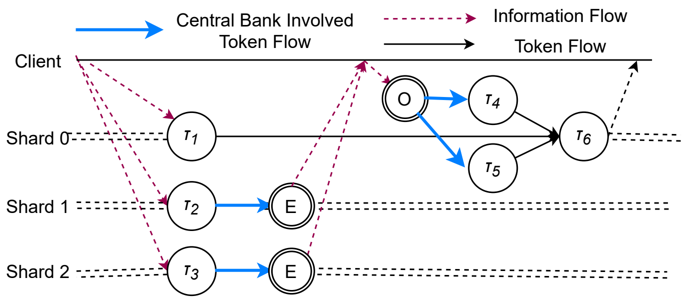

Fig. 17. **Cross-shard Transaction Example**

## B. Performance

For performance verification, we need to test its user scalability, network scalability and latency. These dimensions can be tested by empirical experiments, like a stress test. To ensure experiments are valid, we could choose specific scenarios and operations which should be as close to reality as possible. In our experiments, we initiate random transactions by random users, and we also consider different payment methods, like face-to-face transfer, collecting, etc.

In the recommended algorithm (figure 11), we use sharding in the retail consensus networks to improve network scalability and user scalability. Commercial institutions, including privileged and non-privileged ones, could take the role of leader nodes or validator nodes in the retail consensus networks and undertake customer due diligence. To improve performance, CBDC designers could leverage sharding to increase their capability of handling transactions. Sharding provides horizontal scalability to a CBDC network. However, traditional account-based sharding may bring extra cross-shard transactions due to interactions between multiple operating institutions. Therefore, we use token-based sharding to reduce the number of cross-shard transactions. We leverage AWS EC2 and Corda node to carry out a series of experiments. In our experiments, privileged institutions would take the role of leader nodes when processing their single-shard transactions.

The result in figure 18 shows a linearly increasing TPS, which presents excellent user scalability and network scalability. Furthermore, sharding has little impact on latency shown on the right side in the figure. Unfortunately, since the Corda open-source version has limitations on performance, we could not demonstrate an extremely large TPS in the experiment due to cost control. However, by our experiment, we can see that our method improves the performance of a CBDC system a lot.

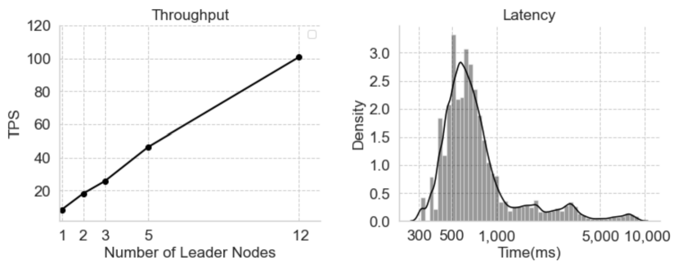

Fig. 18. **Results of performance test, including system throughput and latency.**

If CBDC is a non-fungible token, Token-based sharding can map every non-fungible token to one leader node since the token's creation. Then tokens can be circulated in different ledgers in parallel, increasing the performance and achieving efficient token circulation. However, CBDC is more like a fungible token. Due to changes produced by transactions, the account user might use several tokens in a transaction which causes additional concurrent volume. Moreover, if a CBDC user wants to use two tokens circulating in different ledgers simultaneously, the cross-shard transaction may happen to ensure no double-spending.

In our recommended consensus algorithms, sharding improves performance by parallel running ledgers. This is because cross-shard transactions are relatively less frequent in the token-based sharding method than the account-based one. However, if we consider the performance in the wholesale consensus network, say that each transaction in the retail consensus network needs Verification from all parties in the wholesale networks, more shards may cause worse performance.

Overall, we prove that the proposed algorithms can increase the system's TPS while not sacrificing latency. There are many other kinds of performance tests, in which we use two of them as an example.

## *C. Privacy*

We provide two options as operating models to protect business secrecy. For option two, we designate one operating organization to distribute CBDC because it is not a competitor with other operating institutions such that they would not be worried about their data being monopolized.

For option one, we use dynamic virtual addresses to prevent privileged institutions from knowing customer data of non-privileged ones. The method is similar to the bitcoin schema. In the bitcoin [20] system, there are some essential facts: 1) new addresses are used to collect change in transactions, 2) users could have many addresses. Bitcoin uses this method to protect customer privacy from data leakage to the public, while our model protects non-privileged institutions' data from privileged institutions.

However, in the current model, similar to the bitcoin schema, privileged institutions can still obtain secret information, depending on different transaction types. Besides SISO transactions, SIDO, MIDO, MISO transactions may expose relationships between inputs and outputs. For example, figure 19 shows a SIDO transaction, in which one of the addresses in $v_6$ and $v_9$ would be the address for the change. The relationship between payees and payers could be inferred when collecting enough extra data, like goods, transaction places.

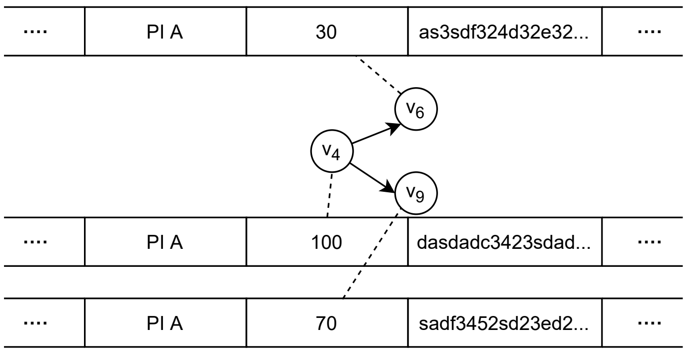

Fig. 19. **Privacy in a SIDO Transaction**

However, in a SISO transaction, the payee and payer are not the same people in most of the cases. If we only have SISO transactions in the network, privileged institutions would not know the relationship between payees and payers. However, in a MISO transaction, tokens coming from different virtual addresses are usually paid from one same client. Similarly, in a SIDO transaction, one of the output tokens should be the change token back to the payer. Moreover, in a MIDO transaction, payers would be the same person in most cases. Therefore, we conclude that only SISO transactions can protect privacy.

This problem is not solved in the bitcoin system, which has been proved that bitcoin can not protect user privacy completely [37]. Nevertheless, in our CBDC operating model, transaction data would be processed by non-privileged institutions before they are sent to privileged institutions. Non-privileged institutions can add virtual entities with virtual addresses to the network and use these virtual entities to create SISO transactions for customers. For example, in a SIDO transaction, non-privileged institutions can use virtual entities as the receiver to avoid the connection between payer and payee and then send it to the actual receiver via a SISO transaction. This can also be applied in other transactions. With enough virtual entities by operators, non-privileged institutions can hide the direct relationship between the inputs and outputs.

## *D. Security*

Here we discuss double-spending as a security example, where we try to prove no double-spending in single-shard and cross-shard transactions separately. Note that the premises below come from lemmas or definitions in the model.

**Lemma 1.** *When a non-issuance transaction is finished, the out-degrees of the input tokens in the ledger state become non-zero.*

$$\forall x.\forall \tau.(Tx(\tau,x) \Rightarrow \forall y.(xRy \Rightarrow d^+_G(\tau,y)! = 0).$$

*Proof.* For a non-issuance transaction in definition 3, the out-degrees of input tokens become non-zero after the transaction is finished. Since (∀x. ∀ $\tau$.(Tx($\tau$,x) ⇒ D = D + TD)), a new transaction graph is added into the ledger where the out-degrees of input tokens would not change. □

**Lemma 2.** *∀x. ∀ $\tau$.(Tx($\tau$,x) ⇒ ∀y.(xRy ⇒ H($\tau$, y)))*

*Proof.* According to lemma 1, we get ∀x. ∀ $\tau$.(Tx($\tau$,x) ⇒ ∀y.(xRy ⇒ $d^+_G(\tau,y)! = 0$).

| | | |
|---|---|---|
| 1. | $\forall x.\forall \tau.(Tx(\tau,x) \Rightarrow \forall y.(xRy \Rightarrow d^+_G(\tau,y)! = 0))$ | Premise |
| 2. | $\forall y.\forall \tau.(H(\tau,y) \Leftrightarrow d^+_G(\tau,y)!=0)$ | Premise |
| 3. | $Tx(\tau,x) \Rightarrow \forall y.(xRy \Rightarrow d^+_G(\tau,y)!=0)$ | UE: 1 |
| 4. | $H(\tau,y) \Leftrightarrow d^+_G(\tau,y)!=0$ | UE: 2 |
| 5. | $Tx(\tau,x)$ | AS |
| 6. | $\forall y.(xRy \Rightarrow d^+_G(\tau,y)!=0)$ | IE: 3,5 |
| 7. | $xRy \Rightarrow d^+_G(\tau,y)!=0$ | UE: 6 |
| 8. | $xRy$ | AS |
| 9. | $d^+_G(\tau,y)!=0$ | IE: 7,8 |
| 10. | $d^+_G(\tau,y)!=0 \Rightarrow H(\tau,y)$ | BE: 4 |
| 11. | $H(\tau,y)$ | IE: 9,10 |
| 12. | $xRy \Rightarrow H(\tau,y)$ | II: 8,11 |
| 13. | $\forall y.(xRy \Rightarrow H(\tau,y))$ | UI: 12 |
| 14. | $Tx(\tau,x) \Rightarrow \forall y.(xRy \Rightarrow H(\tau,y))$ | II: 5,13 |
| 15. | $\forall x.\forall \tau.(Tx(\tau,x) \Rightarrow \forall y.(xRy \Rightarrow H(\tau,y)))$ | UI: 14 |

□

The temporal logic proofs of Lemma 3, 4, 5 and 6 are shown in appendix A. The logic proofs (temporal logic [36]) have been checked by a proof-editor from Stanford University [44].

**Lemma 3.** *∀x.∀ $\tau$.(*Tx*($\tau$,x)) ⇒ ∀ z.(zRx ⇒ F($\tau$, z)))*

**Lemma 4.** *∀x.∀ $\tau$.(F($\tau$,x) ⇔ ¬H($\tau$,x))*

**Lemma 5.** *A recorded token can not be spent twice in different transactions in the network.*

$$\forall x.\forall y.(Tx(\tau_1, x) \wedge Tx(\tau_2, y) \wedge xRy \Rightarrow \tau_1 \neq \tau_2) \qquad (1)$$

**Lemma 6.** *If leaders are non-faulty(H), there is no double-spending in single-shard transactions.*

**Lemma 7.** *If leaders are non-faulty(H), there is no double-spending in cross-shard transactions.*

*Proof.* According to the model, a non-faulty leader with the central bank ensures a cross-shard token is recorded in the ledger. According to lemma 5, we prove a recorded token without double-spending problems. Since the tokens are recorded in the ledger, there can be no double spending in a cross-shard transaction. □

To be mentioned, there are different operating architectures mentioned in BIS's report [43], in which the central bank can record wholesale balance or retail balance. If the currency issuer (Central Bank) secures the Initial Issuance Transaction and the Final Redemption Transaction, there would be no double-spending in cross-shard transactions, no matter central bank records wholesale balance or retail records. However, recording balance can help the central bank in many other ways, like controlling the volume of circulating CBDC in different ledgers.

**Lemma 8.** *If leaders are non-faulty(H), there is no double-spending in all transactions.*

*Proof.* There are two kinds of transactions in the network: transactions in one shard and transactions between two shards. Lemma 6 and 7 prove no double-spending with non-faulty leaders in these two kinds of transactions. Therefore, we conclude no double-spending problem in the network if leaders are non-faulty. □

We have shown that with a non-faulty leader, we can prevent double-spending. However, the assumption is the weakest point in our system. We can believe that the central bank would not perform malicious behaviour. Then in a cross-shard transaction, the central bank can ensure no fault in the wholesale consensus network. However, in the retail consensus network, privileged institutions are responsible for their ledger and decide each transaction on its own without validation. There is no mechanism to ensure them non-fault. As a result, double-spending may happen in single-shard transactions. We leverage mathematical tools to find the weakest point in the system. Math would be a perfect tool for describing a complex system and finding problems inside it.

We sacrifice part of security to increase performance in this example. As we discussed before, the design of CBDC is a trade-off between different dimensions, including performance, security and privacy. For example, leveraging a single institution responsible for all transactions could ensure high performance but bring security issues.

Although we can not avoid double-spending in real-time in our recommended consensus algorithm, we can increase the cost of malicious behaviours. The recommended consensus algorithm in the retail consensus network leverages data-backup from validators to ensure that the leader node would be non-faulty. A leader would send encrypted transactions to validators in the network after Verification. By encryption, customer data would not be shared with validators. Encryption also ensures business secrecy in the CBDC system.

Validators would undertake data backup. Once the leader node changes the original Data, encrypted data from validators can be used to check data consistency. This can help mitigate malicious behaviours from the leader node. If the leader node performs malicious behaviour, they would be punished. Moreover, since the central bank controls issuance and redemption transactions, it will know the balance of money on each ledger so that no extra money would come from retail networks. By data backup, all transactions become immutable in real-time. By checking data consistency, malicious behaviours would be found.

On the other side, we can ensure latency is not being overly influenced because validators in the network would be randomly selected so that not all validators in the network need to join the process of a transaction.

Therefore, leader nodes in our model are motivated to be non-faulty. If validators are competitors, these data should be encrypted to ensure business secrecy. In other cases, validators could be run by third-party auditors. In this scenario, it is not necessary to encrypt the data. Besides, we use 50 per cent as the threshold to make encrypted data backup. Since most validators could be competitors or auditors of the leader node, they should be motivated to keep the encrypted data recorded. To avoid the leader node from being set up by its competitors, there should be at least two validators in this process.

If CBDC designers want a real-time check for fault, we can add auditors in the consensus process and let them vote for each transaction. Then there would be no double-spending even though the leader node is faulty.

## *E. Framework Iterations*

Figure 20 shows how we can iterate in this example. CBDC design involves many trade-offs between different dimensions. In our example, we start from a country with a large population, focusing on performance and privacy. Then we use the evaluation sub-framework to propose solutions. Finally, we come back to the original dimensions in our verification framework, like performance, privacy, and security. We try to verify these dimensions in diverse ways. As shown above, the proposed solution presents an excellent performance and privacy, but when we try to prove there is no double-spending, we find that it is possible. Our verification sub-framework can find the weakest link in the model. We only discuss three dimensions in this example. If CBDC designers have other specific focuses, this framework could have more dimensions.

After Verification, if CBDC designers want to improve system security, they can come back to CBDC dimensions

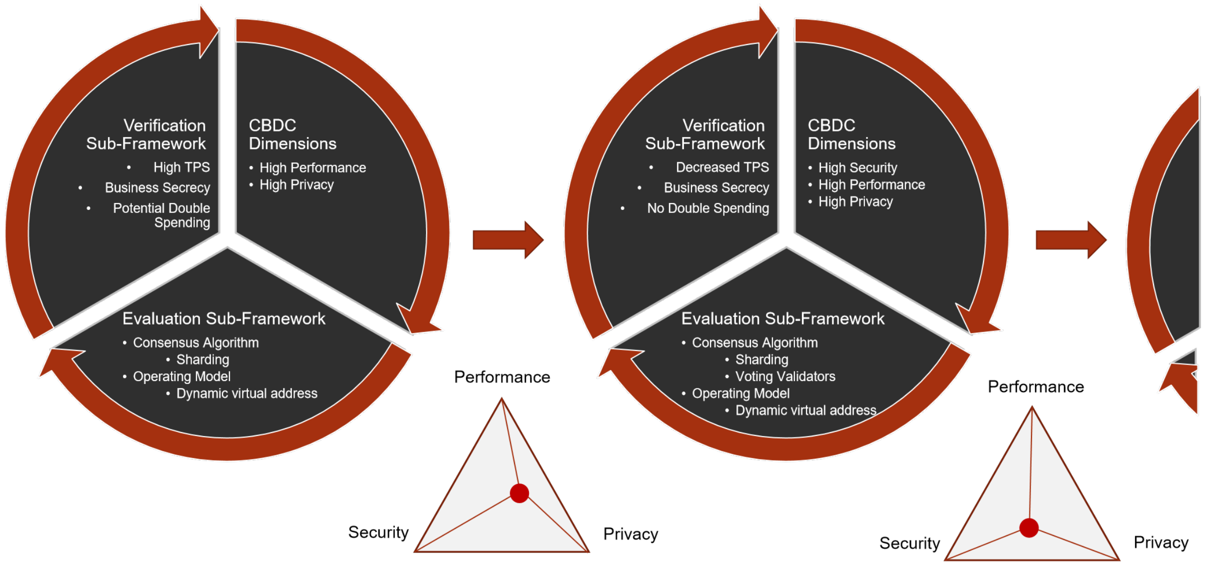

Fig. 20. **Iterations of the CEV Framework until an acceptable balance point**

to determine what they expect again. If they need a more secure system, they could continually use the evaluation sub-framework to propose new solutions and use the verification framework to verify diverse dimensions. In our example, the CBDC designer can leverage the evaluation sub-framework to choose another component in the consensus process and form a Byzantine fault-tolerant algorithm. The newly proposed solution can involve more participants to vote for single-shard transactions, which is helpful to ensure no fault in the retail consensus networks. However, it may potentially influence the system's performance. Afterwards, the CBDC designer can return to CBDC dimensions again to adjust its expectations on different dimensions until finding a balance point.

Our framework presents potential trade-offs in CBDC designs and helps CBDC designers see what they expect with iterations. Finally, after many times of iterations, there would be a balance point for the CBDC designers.

## V. Conclusion

Our paper proposes a CBDC framework (CEV Framework), including an evaluation sub-framework and a verification sub-framework to design central bank digital currency. This work is of significant importance to the evolution of CBDC and proposes an original approach. We provide a holistic solution for CBDC designers who can have a method to analyze potential solutions according to the economic and regulatory conditions of their jurisdictions.

To the best of our knowledge, we are the first to propose a framework to analyze CBDC related technical solutions by splitting consensus algorithms into different components and proposing operating models to solve CBDC related issues. Most importantly, we build a verification sub-framework to prove the feasibility of the recommended algorithms and operating models with rigorous and professional mathematical proofs. Besides, our framework would not bring any new issues.

Our framework could be continuously updated and improved by iterating with the workflow. In addition, there are diverse central bank digital currency projects worldwide. These projects can leverage our framework to design the consensus algorithms better and adopt reasonable operating models. To handle the future need for regulated cryptocurrency design, our framework needs further refinement in practice. We have listed all considerations in CBDC in the CEV framework. The main future work is to include more dimensions and solutions into the framework and simplify the process of using this framework.

## Acknowledgment

The authors gratefully acknowledge the help of Bo Tong Xu on fruitful discussions on some aspects of the present work. This paper is partially based on our work in Global CBDC Challenge [3], in which we were shortlisted into finalists. Based on our framework in this paper, we built an evaluation and recommendation platform, which advocates consensus algorithms for different CBDC designers based on national economic and regulatory conditions. The authors wish to acknowledge the other two teammates, Mark Liu and Bing Qu. This paper is reviewed by Dr Philip Installura. The authors wish to express great appreciation for his valuable input.

## Appendix A
## Formal Logic Proof

Here are temporal logic proofs. Please see notations in section III.

### A. *Lemma 3*

$$\forall x.\forall \tau.(\mathrm{Tx}(\tau, x)) \Rightarrow \forall z.(zRx \Rightarrow F(\tau, z)))$$

*Proof.* A token $\tau$ keeps unspent status when it has not been used in any transaction. For any $\tau \in V, F(\tau, y) \Leftrightarrow (\tau \in$ UTXO at time y) ⇔ $d^+_G(\tau, y) = 0$. From our definition in transaction, we get ∀x. ∀ $\tau$.(Tx($\tau$,x) ⇒ ∀y.(yRx ⇒ $d^+_G(\tau, y) = 0$).

| | | |
|---|---|---|
| 1. | $\forall x.\forall \tau.(Tx(\tau, x) \Rightarrow \forall y.(yRx \Rightarrow d^+_G(\tau, y) = 0))$ | Premise |
| 2. | $\forall y.\forall \tau.(F(\tau, y) \Leftrightarrow d^+_G(\tau, y)=0)$ | Premise |
| 3. | $Tx(\tau, x) \Rightarrow \forall y.(yRx \Rightarrow d^+_G(\tau, y)=0)$ | UE: 1 |
| 4. | $F(\tau, y) \Leftrightarrow d^+_G(\tau, y)=0$ | UE: 2 |
| 5. | $Tx(\tau, x)$ | AS |
| 6. | $\forall y.(yRx \Rightarrow d^+_G(\tau, y)=0)$ | IE: 3,5 |
| 7. | $yRx \Rightarrow d^+_G(\tau, y)=0$ | UE: 6 |
| 8. | $yRx$ | AS |
| 9. | $d^+_G(\tau, y)=0$ | IE: 7,8 |
| 10. | $d^+_G(\tau, y)=0 \Rightarrow F(\tau, y)$ | BE: 4 |
| 11. | $F(\tau, y)$ | IE: 9,10 |
| 12. | $yRx \Rightarrow F(\tau, y)$ | II: 8,11 |
| 13. | $\forall y.(yRx \Rightarrow F(\tau, y))$ | UI: 12 |
| 14. | $Tx(\tau, x) \Rightarrow \forall y.(yRx \Rightarrow F(\tau, y))$ | II: 5,13 |
| 15. | $\forall x.\forall \tau.(Tx(\tau, x) \Rightarrow \forall y.(yRx \Rightarrow F(\tau, y)))$ | UI: 14 |

□

## *B. Lemma 4*

$$\forall x.\forall \tau.(F(\tau, x) \Leftrightarrow \neg H(\tau, x))$$

*Proof.* $\neg H(\tau, x)$ means $\tau$ has not been spent before x. Therefore, F($\tau$,x) ⇔ $d^+_G(\tau, x)! = 0$ ⇔ $\neg H(\tau, x)$.

| | | |
|---|---|---|
| 1. | $\forall x.\forall \tau.(F(\tau, x) \Leftrightarrow d^+_G(\tau, x)!=0)$ | Premise |
| 2. | $\forall x.\forall \tau.(H(\tau, x) \Leftrightarrow d^+_G(\tau, x)!=0)$ | Premise |
| 3. | $\forall \tau.(F(\tau, x) \Leftrightarrow d^+_G(\tau, x)!=0)$ | UE:1 |
| 4. | $F(\tau, x) \Leftrightarrow d^+_G(\tau, x)!=0$ | UE:3 |
| 5. | $\forall \tau.(H(\tau, x) \Leftrightarrow d^+_G(\tau, x)!=0)$ | UE:2 |
| 6. | $H(\tau, x) \Leftrightarrow d^+_G(\tau, x)!=0$ | UE:5 |
| 7. | $F(\tau, x) \Rightarrow d^+_G(\tau, x)!=0$ | BE:4 |
| 8. | $d^+_G(\tau, x)!=0 \Rightarrow F(\tau, x)$ | BE:4 |
| 9. | $H(\tau, x) \Rightarrow d^+_G(\tau, x)!=0$ | BE:6 |
| 10. | $d^+_G(\tau, x)!=0 \Rightarrow H(\tau, x)$ | BE:6 |
| 11. | $F(\tau, x)$ | AS |
| 12. | $d^+_G(\tau, x)!=0$ | IE:7,11 |
| 13. | $H(\tau, x)$ | IE: 10, 12 |
| 14. | $F(\tau, x) \Rightarrow H(\tau, x)$ | II:11,13 |
| 15. | $H(\tau, x)$ | AS |
| 16. | $d^+_G(\tau, x)!=0$ | IE:9,15 |
| 17. | $F(\tau, x)$ | IE:8,16 |
| 18. | $H(\tau, x) \Rightarrow F(\tau, x)$ | II:15,17 |
| 19. | $F(\tau, x) \Leftrightarrow H(\tau, x)$ | BI:14,18 |
| 20. | $\forall \tau.(F(\tau, x) \Leftrightarrow H(\tau, x))$ | UI:19 |
| 21. | $\forall x.\forall \tau.(F(\tau, x) \Leftrightarrow H(\tau, x))$ | UI:20 |

□

## *C. Lemma 5*

$$\forall x.\forall y.(Tx(\tau_1, x) \wedge Tx(\tau_2, y) \wedge xRy \Rightarrow \tau_1 \neq \tau_2)$$

*Proof.* Time is continuous that given any two timestamps, there is one timestamp between them. we assume a double spending transaction possible as one premise to find a contradiction.

| | | |
|---|---|---|
| 1. | $\forall x.\forall \tau.(\mathrm{Tx}(\tau, x) \Rightarrow \forall y.(xRy \Rightarrow H(\tau, y)))$ | Premise |
| 2. | $\forall x.\forall \tau.(\mathrm{Tx}(\tau, x)) \Rightarrow \forall z.(zRx \Rightarrow F(\tau, z)))$ | Premise |
| 3. | $\forall x.\forall \tau.(F(\tau, x) \Leftrightarrow \neg H(\tau, x))$ | Premise |
| 4. | $\forall x \forall y.(xRy \Rightarrow (\exists z.(xRz \wedge zRy)))$ | Premise |
| 5. | $\exists x.\exists y.(xRy \wedge Tx(\tau, x), Tx(\tau, y))$ | goal |
| 6. | $\exists y.([x]Ry \wedge Tx(\tau, [x]), Tx(\tau, y))$ | EE: 5 |
| 7. | $[x]R[y] \wedge Tx(\tau, [x]), Tx(\tau, [y])$ | EE: 6 |
| 8. | $[x]R[y]$ | AE: 7 |
| 9. | $Tx(\tau, [x])$ | AE: 7 |
| 10. | $Tx(\tau, [y])$ | AE: 7 |
| 11. | $\forall y.([x]Ry \Rightarrow (\exists z.([x]Rz \wedge zRy))$ | UE: 4 |
| 12. | $[x]R[y] \Rightarrow (\exists z.([x]Rz \wedge zR[y]))$ | UE: 11 |
| 13. | $\exists z.([x]Rz \wedge zR[y])$ | IE: 8,12 |
| 14. | $[x]R[z] \wedge [z]R[y]$ | EE: 13 |
| 15. | $[x]R[z]$ | AE: 14 |
| 16. | $[z]R[y]$ | AE: 14 |
| 17. | $\forall \tau.(\mathrm{Tx}(\tau, [x]) \Rightarrow \forall y.([x]Ry \Rightarrow H(\tau, y)))$ | UE: 1 |
| 18. | $\forall \tau.(\mathrm{Tx}(\tau, [y]) \Rightarrow \forall z.(zR[y] \Rightarrow F(\tau, z)))$ | UE: 2 |
| 19. | $\mathrm{Tx}(\tau, [x]) \Rightarrow \forall y.([x]Ry \Rightarrow H(\tau, y))$ | UE: 17 |
| 20. | $\mathrm{Tx}(\tau, [y]) \Rightarrow \forall z.(zR[y] \Rightarrow F(\tau, z))$ | UE: 18 |
| 21. | $\forall y.([x]Ry \Rightarrow H(\tau, y))$ | IE: 9,19 |
| 22. | $\forall z.(zR[y] \Rightarrow F(\tau, z))$ | IE: 10,20 |
| 23. | $[x]R[z] \Rightarrow H(\tau, [z])$ | UE: 21 |
| 24. | $[z]R[y] \Rightarrow F(\tau, [z])$ | UE: 22 |
| 25. | $H(\tau, [z]))$ | IE: 15,23 |
| 26. | $F(\tau, [z]))$ | IE: 16,24 |
| 27. | $\forall \tau.(F(\tau, [z]) \Leftrightarrow \neg H(\tau, [z])$ | UE: 3 |
| 28. | $F(\tau, [z]) \Leftrightarrow \neg H(\tau, [z])$ | UE: 27 |
| 29. | $F(\tau, [z]) \Rightarrow \neg H(\tau, [z])$ | BE: 28 |
| 30. | $\neg H(\tau, [z])$ | IE: 26,29 |
| 31. | $Contradiction$ | 25,30 |

With proof by contradiction, we get that a recorded token in the validator's ledger can not be spent twice in different transactions. □

## *D. Lemma 6*

Assume a leader is non-faulty(H), there is no double-spending in its shard.

*Proof.* In a token chain, tokens (a) with in-degree 0 are created and issued by an issuer. For a valid payment transaction, a non-faulty leader would ensure the received token and change token is recorded, after which the transaction will be announced as valid.

| | | |
|---|---|---|
| 1. | $p(a)$ | Premise |
| 2. | $\forall\tau.(H\wedge p(\tau)\Rightarrow p(r(\tau))\wedge p(c(\tau)))$ | Premise |
| 3. | $H\wedge p(\tau)\Rightarrow p(r(\tau))\wedge p(c(\tau))$ | UE: 2 |
| 4. | $H$ | AS |
| 5. | $p(\tau)$ | AS |
| 6. | $H\wedge p(\tau)$ | AI: 4,5 |
| 7. | $p(r(\tau))\wedge p(c(\tau))$ | IE: 3,6 |
| 8. | $p(r(\tau))$ | AE: 7 |
| 9. | $p(\tau)\Rightarrow p(r(\tau))$ | II: 5,8 |
| 10. | $\forall\tau.(p(\tau)\Rightarrow p(r(\tau))$ | UI: 9 |
| 11. | $H\Rightarrow(\forall\tau.(p(\tau)\Rightarrow p(r(\tau))))$ | II: 4,10 |
| 12. | $H$ | AS |
| 13. | $p(\tau)$ | AS |
| 14. | $H\wedge p(\tau)$ | AI: 12,13 |
| 15. | $p(r(\tau))\wedge p(c(\tau))$ | IE: 3,12 |
| 16. | $p(c(\tau))$ | AE: 15 |
| 17. | $p(\tau)\Rightarrow p(c(\tau))$ | II: 13,16 |
| 18. | $\forall\tau.(p(\tau)\Rightarrow p(c(\tau))$ | UI: 17 |
| 19. | $H\Rightarrow(\forall\tau.(p(\tau)\Rightarrow p(c(\tau))))$ | II: 12,18 |
| 20. | $H$ | AS |
| 21. | $\forall\tau.(p(\tau)\Rightarrow p(r(\tau)))$ | IE: 11,20 |
| 22. | $\forall\tau.(p(\tau)\Rightarrow p(c(\tau)))$ | IE: 19,20 |
| 23. | $\forall\tau.(p(\tau))$ | Ind: 1, 21, 22 |
| 24. | $H\Rightarrow(\forall\tau.p(\tau))$ | II: 20,23 |

□

## References

[1] Adrian, Tobias and Tommaso Mancini-Griffoli (2019), 'The Rise of Digital Money', Fintech Notes no. 19/01, IMF.

[2] Boar, Codruta, Henry Holden, and Amber Wadsworth. "Impending arrival–a sequel to the survey on central bank digital currency." BIS paper 107 (2020).

[3] Monetary Authority of Singapore, "Global CBDC Challenge Problem Statements," 2021, [Online] Available: https://hackolosseum.apixplatform.com/hackathon/globalcbdcchallenge

[4] R Auer, G Cornelli and J Frost (2020), "Rise of the central bank digital currencies: drivers, approaches and technologies", BIS working papers, No 880, August.

[5] Boar, Codruta, and Andreas Wehrli. "Ready, steady, go?-Results of the third BIS survey on central bank digital currency." (2021).

[6] Kiff, Mr John, et al. "A survey of research on central bank digital currency." (2020).

[7] Mancini-Griffoli, Tommaso, et al. "Casting light on central bank digital currency." IMF staff discussion note 8 (2018).

[8] Chapman, James, et al. "Project Jasper: Are distributed wholesale payment systems feasible yet." Financial System 59 (2017).

[9] Monetary Authority of Singapore. "Project Ubin: Central Bank digital money using distributed ledger technology." (2017).

[10] Deloitte, Monetary Authority of Singapore and Singapore Exchange (2018): Delivery versus payment on distributed ledger technologies: Project Ubin, August.

[11] Fernández-Villaverde, Jesús, et al. "Central bank digital currency: Central banking for all?." Review of Economic Dynamics 41 (2021): 225-242.

[12] Bank, De Nederlandsche. "Central Bank Digital Currency. Objectives, preconditions and design choices." Occasional Studies 20 (2020).

[13] Auer, Raphael, and Rainer Böhme. "The technology of retail central bank digital currency." BIS Quarterly Review, March (2020).

[14] Kiff, Mr John, et al. "A survey of research on retail central bank digital currency." (2020).

[15] Auer, Raphael A., Giulio Cornelli, and Jon Frost. Rise of the central bank digital currencies: drivers, approaches and technologies. No. 8655. CESifo Working Paper, 2020.

[16] Lannquist, Ashley, S. Warren, and R. Samans. "Central bank digital currency policy-maker toolkit." Insight Report, World Economic Forum, Geneva. 2020.

[17] Calle, George, and Daniel Eidan. "Central Bank Digital Currency: an innovation in payments." R3 White Paper, April (2020).

[18] Kshemkalyani, Ajay D., and Mukesh Singhal. Distributed computing: principles, algorithms, and systems. Cambridge University Press, 2011.

[19] Kiff, Mr John, et al. "A survey of research on retail central bank digital currency." (2020).

[20] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." Decentralized Business Review (2008): 21260.

[21] Castro, Miguel, and Barbara Liskov. "Practical byzantine fault tolerance." OSDI. Vol. 99. No. 1999. 1999.

[22] Cachin, Christian. "Architecture of the hyperledger blockchain fabric." Workshop on distributed cryptocurrencies and consensus ledgers. Vol. 310. No. 4. 2016.

[23] Brown, Richard Gendal, et al. "Corda: an introduction." R3 CEV, August 1 (2016): 15.

[24] Eyal, Ittay, et al. "Bitcoin-ng: A scalable blockchain protocol." 13th USENIX symposium on networked systems design and implementation (NSDI 16). 2016.

[25] Poon, Joseph, and Thaddeus Dryja. "The bitcoin lightning network: Scalable off-chain instant payments." (2016).

[26] Ongaro, Diego, and John Ousterhout. "In search of an understandable consensus algorithm." 2014 USENIX Annual Technical Conference (USENIXATC 14). 2014.

[27] G30 Working Group on Digital Currencies.Digital Currencies and-Stablecoins: Risks, Opportunities, and Challenges Ahead. 2020.

[28] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding", in IEEE Symposium on Security and Privacy (SP), IEEE, 2018, pp. 19–34.

[29] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: A fast blockchain protocol via full sharding.", IACR Cryptology ePrint Archive, vol. 2018, p. 460, 2018.

[30] Allen, Sarah, et al. Design choices for central bank digital currency: Policy and technical considerations. No. w27634. National Bureau of Economic Research, 2020.

[31] Fatás, Antonio, ed. "The Economics of fintech and digital currencies." London: CEPR Press, 2019.

[32] Kiff, Mr John, et al. "A survey of research on central bank digital currency." (2020).

[33] Bank of Thailand (BoT) and Hong Kong Monetary Authority(HKMA). "Inthanon-lionrock leveraging distributed ledger technology to increase efficiency in cross-border payments," January 2020.

[34] Y.-T. Lin, Istanbul byzantine fault tolerance (08 2017). URL https://github.com/ethereum/EIPs/issues/650

[35] Wood, Gavin. "Ethereum: A secure decentralized generalized transaction ledger." Ethereum project yellow paper 151.2014 (2014): 1-32.

[36] Rescher, Nicholas, and Alasdair Urquhart. Temporal logic. Vol. 3. Springer Science & Business Media, 2012.

[37] Androulaki, Elli, et al. "Evaluating user privacy in bitcoin." International conference on financial cryptography and data security. Springer, Berlin, Heidelberg, 2013.

[38] Agur, Itai, Anil Ari, and Giovanni Dell'Ariccia. "Designing central bank digital currencies." Journal of Monetary Economics (2021).

[39] Darbha, Sriram, and Rakesh Arora. Privacy in CBDC technology. No. 2020-9. Bank of Canada, 2020.

[40] Auer, Raphael, Philipp Haene, and Henry Holden. "Multi-CBDC arrangements and the future of cross-border payments." (2021).

[41] Sveriges Riksbank (2018), 'The Sveriges Riksbank's e-krona project: Report 2', October.

[42] Sveriges Riksbank, The Riksbank to test technical solution for the e-krona, Riksbank press release, 20 Feb. 2020.

[43] Auer, Raphael, and Rainer Boehme. Central bank digital currency: the quest for minimally invasive technology. No. 948. Bank for International Settlements, 2021.

[44] Standford University. "Logica Formal Logic Proof", 2021, [Online] Available: http://logica.stanford.edu/logica/homepage/index.php