BANK OF CANADA
BANQUE DU CANADA

Staff Discussion Paper/Document d'analyse du personnel—2025-9

Last updated: June 13, 2025

A Retail CBDC Design for Basic Payments: Feasibility Study

Ram Darbha
Banking and Payments Department
Bank of Canada
SDarbha@bankofcanada.ca

Rakesh Arora
Banking and Payments Department
Bank of Canada
RakeshArora@bankofcanada.ca

Cyrus Minwalla
Information Technology Services Department
Bank of Canada
CMinwalla@bankofcanada.ca

Dinesh Shah
Banking and Payments Department
Bank of Canada
dshah@bankofcanada.ca

Bank of Canada staff discussion papers are completed staff research studies on a wide variety of subjects relevant to central bank policy, produced independently from the Bank's Governing Council. This research may support or challenge prevailing policy orthodoxy. Therefore, the views expressed in this paper are solely those of the authors and may differ from official Bank of Canada views. No responsibility for them should be attributed to the Bank.

DOI: https://doi.org/10.34989/sdp-2025-9 | ISSN 1914-0568

© 2025 Bank of Canada

Acknowledgements

We are grateful for the technical advice from Sam Stuewe and other colleagues at the Massachusetts Institute of Technology Digital Currency Initiative (MIT DCI), with whom the Bank of Canada has conducted collaborative research into digital assets and fintech over the past two years.

Abstract

We frame the wide spectrum of possible system architectures for an online retail central bank digital currency (CBDC) and identify a promising architecture well-suited for basic payments. We select OpenCBDC 2PC, a representative system design that fits this architecture and analyze it using a range of criteria to assess the feasibility of such system designs. Our analysis, augmented with lab experiments, focuses on retail payment systems with two-tier deployment and includes a detailed assessment of non-repudiation, integrity of the monetary supply, privacy, compliance, scalability of performance and resilience of the system state. It suggests that such system designs can be fast and cheap for basic payments, with high privacy, although some areas such as integration with retail payments systems, performance of auditing and resilience of the core system state require further investigation. Our framing highlights other promising architectures for an online retail CBDC, whose analysis we leave as an area for further exploration.

Topics: Central bank research, Digital currencies and fintech

JEL codes: E, E4, E42, E5, E51, O, O3

Résumé

Nous répertorions le large éventail d'architectures de système possibles pour une monnaie numérique de banque centrale de détail utilisable en ligne et cernons une architecture prometteuse qui serait adaptée pour les paiements de base. Nous choisissons le système OpenCBDC avec validation à deux phases, qui correspond bien à l'architecture recherchée, et l'analysons en utilisant une série de critères pour évaluer la faisabilité d'un tel système. Notre analyse, qui inclut des expériences en laboratoire, porte principalement sur les systèmes de paiement de détail avec un protocole de validation à deux phases et comprend une évaluation détaillée de la non-répudiation, de l'intégrité de l'offre de monnaie, de la confidentialité, de la conformité, de l'extensibilité et de la résilience du système. Il en ressort que ce genre de système peut effectuer des paiements de base rapidement et à faible coût et offrir une grande confidentialité. Toutefois, certains aspects, comme l'intégration avec les systèmes de paiement de détail, la vérification de l'offre de monnaie et la résilience du système de base, doivent faire l'objet d'un examen plus poussé. Nous mettons aussi en lumière d'autres architectures prometteuses pour une monnaie numérique de banque centrale de détail, qui devront être examinées plus à fond.

Sujets : Recherches menées par les banques centrales; Monnaies numériques et technologies financières

Codes JEL : E, E4, E42, E5, E51, O, O3

Executive summary

Darbha (2022) develops **five archetypes,** or common system design patterns, for retail CBDC: the centralized, the leaderless, the micro- and macro-partitioned, and the direct. In this paper, we analyze combinations of the archetypes with **three funds models**, or representations of money, to identify one system architecture that is, in our opinion, well-suited for basic payments in an online retail CBDC system. This is a micro-partitioned system based on an unspent transaction output (UTXO) funds model.

We select a representative system design, **OpenCBDC 2PC** (Massachusetts Institute of Technology Digital Currency Initiative 2022), that fits this architecture for basic payments and analyze it using a range of criteria to assess the feasibility of such system designs for an online retail CBDC. From our findings, summarized below, we do not anticipate serious obstacles to achieving feasibility, although some areas require further investigation.

Designs like OpenCBDC 2PC align well with a **two-tier system**, allowing a separation of central bank (CB) and intermediary functions. We define three types of wallets and show they can coexist in a single system, enabling both user custody and institutional custody of funds as well as payments to users who are either present or non-present.

A key attribute of micro-partitioned designs like OpenCBDC 2PC is that a transaction involves two legs—a core system update and a wallet-to-wallet transfer—making it more complex than a transaction in a traditional account-based system, which involves only one leg. We describe a payment flow and necessary system functions that would enable the system to confirm the completion of both legs and thereby achieve strong **non-repudiation guarantees**. Further, the current **retail payments ecosystem** is underpinned by account-based systems. The additional complexity and different funds model of designs such as OpenCBDC 2PC mean that design effort is needed to integrate such systems into the retail payments ecosystem. We note this as an open area for further exploration.

We assess two variants of OpenCBDC 2PC, both with and without privacy-preserving techniques, to **audit the money supply**, i.e., to aggregate the supply and detect discrepancies. Our assessment indicates that these mechanisms can feasibly be used to detect a discrepancy in the CBDC supply. The performance impact of auditing needs to be investigated for a production-size supply, especially for the privacy-preserving variant that is compute-intensive. Further, a production system must not only be capable of detecting a discrepancy but also identifying the root cause and reversing it. The complexity and performance impact of reversal mechanisms are unknown, especially for a monetary supply of national size, the investigation of which we leave for further exploration.

Our **scalability** experiments with three variants of OpenCBDC 2PC—the baseline and two auditability variants—suggest that the performance of the baseline scales efficiently, with low response times suitable for retail use. The two auditability variants do not scale as efficiently, although they can likely be improved over time. Nevertheless, our observations highlight the efficiency of such system designs as even the least performant configurations demonstrate throughput in excess of 10,000 transactions per second.

Micro-partitioned systems such as OpenCBDC 2PC that support user custody naturally support high levels of **privacy** from the CB and, optionally, from intermediate institutions too. A user could be non-registered or keep custody of their own funds to achieve high privacy. These ideas could be combined with the use of privacy-enhancing technologies to hide amounts even

from the settlement system, to achieve a very high level of privacy exceeding what is available in today's electronic payment systems.

Compliance in systems such as OpenCBDC 2PC would be the responsibility of intermediaries, in alignment with the two-tier model. Both rule-based and principle-based compliance would be feasible for registered users that operate wallets in the custody of intermediaries. The system would have to impose controls (e.g., a transaction amount) to prevent illicit usage for non-registered users or users that have custody of their own funds, should those options be permitted by policy-makers.

The core system state of a system such as OpenCBDC 2PC constitutes its "crown jewels" and must never be lost. Hence, the **resilience of the core state** is critical, and its recovery must be guaranteed, even in the event of a total system outage. While this is theoretically feasible, engineering effort is needed to make it efficient and practical. We outline the enhancements needed to achieve recovery of the core state in such a system. We leave to future work how to achieve these goals, which would depend heavily on the state replication mechanisms and database technologies used. The **resilience of wallets** would be the responsibility of wallet custodians such as money services businesses (MSBs) and wallet providers.[1] We note that institutional backup and recovery mechanisms are widely used by enterprises already. Further, technical solutions already exist for mobile content backup and recovery targeted to end users, which could be employed by providers of CBDC wallets.

From the above, some possible areas of **future work** within micro-partitioned systems such as OpenCBDC 2PC include integration with the retail payments ecosystem; auditability of a production-size monetary supply, while preserving privacy; identification of the root-causes and reversal of monetary supply discrepancies; preservation of scalability with auditability; and resilience and timely recovery of the core system state from persistent storage.

Our framing highlights **other promising architectures** for an online retail CBDC. For example, the centralized or leaderless archetypes combined with the account model would support arbitrarily complex, full-featured programmable arrangements while providing a strong coherence guarantee. These arrangements arguably would be well-suited for contingent payments. The feasibility analysis of alternative architectures remains an open area for further exploration.

[1] Money services businesses are regulated institutions authorized to provide CBDC services to end users. These businesses include deposit-taking banks but could also include other entities, such as financial technology businesses (fintechs).

Section 1: Introduction

A wide variety of technology designs have been, and continue to be, proposed to underpin retail central bank digital currency (CBDC) systems. By considering the fundamental perspective of how information, or state, related to a CBDC instrument is organized among participating entities, Darbha (2022) develops five archetypes, or common patterns, that recur in system designs. He then analyzes them using a range of criteria such as privacy, compliance, scalability, resilience and offline payments. Given the variety of possible system designs for a retail CBDC, these archetypes provide a common framework and terminology to analyze and evaluate technical designs, independent of vendor, platform and technology.

In this discussion paper, we extend our analysis by combining the archetypes (*how* the state is organized) with three funds models (*what* the state is) and analyze the resulting architectures for their relative trade-offs. Among them, we identify one architecture for basic payments that is a promising candidate for further analysis and experimentation. We provide a detailed analysis of that architecture.

It is useful to clarify the purpose of this work. Firstly, this should not be interpreted as a recommendation to issue a retail CBDC, now or sometime later. That decision is outside the scope of this analysis. It is also not a recommendation in favour of a particular technical system, product or vendor. Instead, we assess the feasibility of a set of design ideas that are, in our opinion, well-suited for basic payments and have been realized in a functioning research system. This analysis is a step in the iterative refinement of technical design options for a retail CBDC. We anticipate that, should a future phase of CBDC development be undertaken, similar analyses of other promising architectures would be conducted before identifying optimal technical architectures that meet the policy goals of a retail CBDC in a particular jurisdiction.

The rest of this paper is organized as follows:

In Section 2, we present a brief background on retail CBDC archetypes and funds models and discuss their combinations to identify a promising architecture for basic payments.

In Section 3, we analyze this architecture using a set of core requirements and applying these to OpenCBDC, a leading system design from the Massachusetts Institute of Technology's (MIT) Digital Currency Initiative (DCI), as a representative example. We augment our analysis with experimental observations where applicable.

In Section 4, we conclude with a summary of findings and lessons learned on how well such system designs for basic payments align with the core requirements for an online retail CBDC system.

We present a proposed transaction flow in Appendix A and details of the scalability test results in Appendix B.

Section 2: Background on archetypes and funds models

2.1 Funds models

We follow the organization of funds models—the way in which funds are represented—described by the Eesti Pank and Guardtime (Buldas et al. 2021) as three properties of value, ownership and identity.[2] Each property may be fixed or variable, as outlined in **Table 1**.

As Buldas et al. (2021) show, there are eight possible enumerations, of which some are unfeasible, leading to three primary funds models: account, unspent transaction output (UTXO) and bill (**Table 2**).

Table 1: Three properties of funds models, each with two flavours

Property	Fixed	Variable
Value	The monetary value is unchanged over many transactions.	The monetary value can change on every transaction.
Ownership	Owner of the instrument doesn't change across transactions.	Owner changes on each transaction.
Identity	The instrument preserves its identity across transactions, i.e., it is long-lived.	The instrument changes its identity, e.g., on every transaction, i.e., it is short-lived.

Table 2: Three feasible funds models and their properties

Property / Funds model	Value	Ownership	Identity
Account	Variable	Fixed	Fixed
Unspent transaction output[3]	Variable	Variable	Variable
Bill	Fixed	Variable	Fixed

An **account** is long-lived and has a fixed owner, while its value or balance changes over time. A **bill** is long-lived and has a fixed value, while its owner changes over time. A **UTXO** has a value and an owner, but its distinguishing feature is that it is short-lived. It is created as an output of a transaction with a fixed owner and value, and only exists until it is spent as an input of a subsequent transaction, which creates other outputs, and so on.

Among the three funds models, the UTXO and bill models are often considered more scalable than the account model since their processing is more easily parallelizable. This is because a single user can make multiple payments independently when using bills or UTXOs, while payments from their account must be serialized. However, Buterin (2016) and George, Dryja

[2] Buldas et al. (2021) use the term *money scheme,* while we use the term *funds model*.

[3] Value is variable because, in the set of UTXOs, values are not preserved across a transaction. Ownership is variable because a UTXO's owner changes from one before the transaction to zero after it. Identity is variable because there is no identification number, serial number or account number that can be traced across multiple transactions.

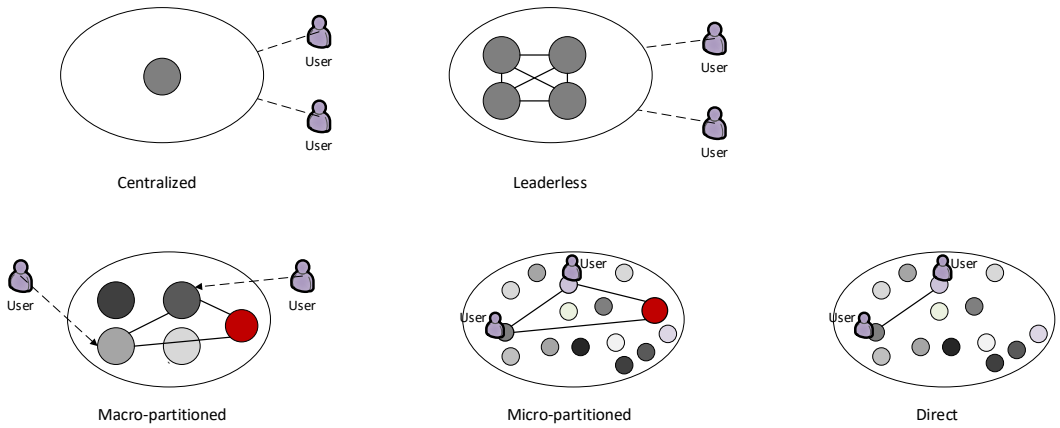
and Narula (2023) suggest the account model can better express complex contingent payment arrangements than UTXO and bill models.

2.2 Archetypes of a retail central bank digital currency

Darbha (2022) develops five archetypes, or common patterns, that recur in system designs: the centralized, the leaderless, the micro- and macro-partitioned, and the direct (**Figure 1**). He analyzes them using a range of criteria such as privacy, compliance, scalability, resilience and offline payments. These archetypes are general enough to collectively cover a wide range of possible CBDC designs.

Briefly, in **centralized** systems, one entity stores and controls the system state. In **leaderless** systems, the system state is replicated to multiple identical entities without a leader. An update changes all replicas the same way. The crowd of replicas provides oversight by collectively approving the update. In **macro**- and **micro-partitioned** systems, the system state is divided into partitions where no single partition represents the total system state. In macro-partitioned systems, partitions are large and owned and operated by institutions, while in micro-partitioned systems, they are small and owned by end users. An update changes only some partitions and is overseen by a trusted authority. In **direct** systems, the system state is divided into partitions that transact directly with each other without involving other parties. An update involving two parties does not require approval by a third party. The **centralized**, **leaderless** and **macro-partitioned** archetypes represent systems in which only institutions—the central bank (CB) and private entities—maintain state. The **micro-partitioned** and **direct** archetypes represent systems in which some state information about funds is maintained in, and transferred between, end-user devices.

Figure 1: Archetypes of a central bank digital currency



Note: Each oval represents a CBDC system. Each dot is a distinct entity that stores the system state, partially or fully. Dots with the same colour denote replicas of the same state information, while dots coloured differently differ in the information they hold. The red dot denotes an authority (entity or protocol) that has the power to approve or deny a state change such as a transaction. Solid lines show the entities involved in finalizing one state change, e.g., settling a transaction. Users outside the CBDC system hold no state (their access is shown with dotted lines), while those inside hold some state.

The progression from **centralized** to **partitioned** and finally to **direct** systems can be visualized as a migration of the system state outward from the core. At one end of this continuum, the

state is fully in the centre, solely in the CB's control; then, some of the state moves outward to non-CB entities. Finally, it resides entirely in the periphery, having migrated away from the CB.

2.3 Combinations of archetypes and funds models

We consider a system architecture as a combination of an archetype and a funds model. Our goal is to identify system architectures that have particular strengths and are likely to be optimal choices for a retail CBDC.

We can make five key inferences about the combinations of these archetypes and funds models (summarized in **Table 3**):

- No archetype scores highly across all criteria, so a design based on a single archetype is not likely to satisfy all policy goals (Darbha 2022).
- Only system designs based on the direct archetype are capable of offline settlement, i.e., a settlement between two parties that doesn't involve a third party. However, the lack of third-party oversight presents risks that make such designs sub-optimal for an online CBDC.[4]
- The ability to access and reason over the full global state permits contingent payments that are more expressive (George, Dryja and Narula 2023). The two archetypes that present a global state, the centralized and leaderless, are best suited for complex contingent payments.
- Leaderless systems appear to offer a weak fit for governance since they involve multiple parties sharing authority, while a retail CBDC is issued by and is the liability of a single authority.[5] Further, these systems' high overhead inhibits their ability to scale, arguably making them sub-optimal for a retail CBDC, as is the case for designs aligned closely with cryptocurrency systems such as Bitcoin and Ethereum.[6]
- Of the two partitioned archetypes, the micro and the macro, the micro has two advantages. Firstly, it is more private than the macro-partitioned since it allows end users to keep custody of their funds, if they choose, while the macro archetype permits only institutional custody. Secondly, it is more flexible than the macro-partitioned since every system designed for users to have custody of their funds can also support institutional custody, while the reverse is not necessarily true.

[4] The authority, a third party, would have no visibility into transactions *at the time of settlement*. Even if the system permits an authority to learn transaction details at a later time, it cannot stop fraudulent transactions before they are settled. This is a serious, and arguably unnecessary, risk in an online system.

[5] The argument regarding a weak governance fit applies to a retail CBDC system operating *within* a jurisdiction where a single authority (the CB) oversees the fiat currency instrument. If, however, a system is intended to span several jurisdictions, say a cross-border system involving multiple CBDCs, then sharing authority as per the leaderless archetype could be a strength and, indeed, a necessity.

[6] Some leaderless system designs claim to achieve throughput in thousands of transactions per second (TPS). While they may satisfy retail CBDC needs in some jurisdictions, they are, nevertheless, generally incapable of linear scaling, i.e., achieving higher throughput by adding more capacity.

Table 3: Conceivable system architectures for an online retail central bank digital currency

Funds model / Archetype	Account	UTXO	Bill[7]
Centralized	*The account model fits well with centralized systems.*		
Leaderless	*Leaderless systems have a weak governance fit to the central bank's role and lack the ability to scale for an online CBDC.*		
Macro-partitioned	*Macro-partitioned systems are less flexible than micro-partitioned systems because they do not support custody by end users.*		
Micro-partitioned		*Micro-partitioned systems based on UTXO are well-suited for basic payments.*	
Direct	*Direct systems offer the central bank limited visibility for an online CBDC.*		

Note: UTXO means unspent transaction output. The shaded architecture is a promising candidate for an online retail CBDC system and the subject of the analysis in this report.

Among the combinations, we highlight one that is a promising candidate for an online retail CBDC since it is well-suited for basic payments (shown as the shaded box in **Table 3**). It is based on the UTXO funds model combined with the micro-partitioned archetype. Micro-partitioned systems can be scalable if designed to minimize the core state and optimize updates to it. We expect such systems to be fast and cheap for basic payments, with high privacy. Further, they could evolve to support micropayments, should such novel uses experience uptake in the future. [8] These systems, however, would have limited, if any, support for complex contingent payments.

In Section 3, we select a representative research system design that fits the architecture for basic payments and analyze it using a range of criteria.

First, we assess how the architecture would fit in a **two-tier model** where the CB would be responsible for issuance, validation and settlement functions and money services businesses (MSBs) for providing customer-facing functions such as know-your-customer (KYC) and account management. We then analyze how a **retail payment flow** could be realized with such an architecture and how it would differ from today's typical retail payment flow. We discuss how such an architecture could provide guarantees for **non-repudiation** to resolve questions about a disputed payment. We discuss experimental results on the ability of such a system to **scale** throughput (the number of transactions settled, measuring the speed of core system updates) as demand increases. We assess the level of **privacy** users would have from the CB and non-CB entities in such a system, and how legal obligations related to **compliance** could

[7] Other differences exist between the UTXO and bill models. Anecdotally, UTXOs are used more widely in online systems, while bills appear more in offline store-of-value systems. This may be because UTXOs require proving only validity, while bills require proving both validity and uniqueness. We leave investigation of these models' relative trade-offs for future work.

[8] Micropayments are payments of low value (less than a dollar) and high volume.

be satisfied. Lastly, we assess how the system could guarantee the **integrity of the supply**, maintain the **resilience of the system state,** and **recover funds** in case of loss.

Note that our framing as in Table 3 suggests there are other promising architectures to be explored. For example, the centralized and leaderless archetypes that permit access and reasoning over a global system state, when combined with the account model, would support arbitrarily complex, full-featured arrangements for contingent payments with a strong coherence guarantee (George, Dryja and Narula 2023). They would also support basic payments, of course, although probably at a higher cost owing to the overhead of a programmable platform. We leave the analysis of other architectures to future work.

Section 3: Analysis of a system for basic payments

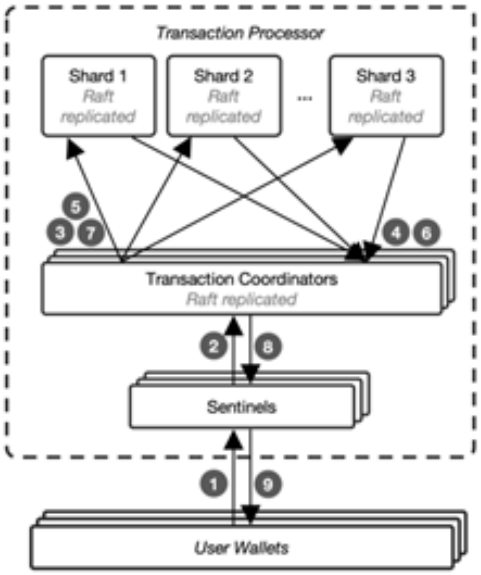
3.1 Choice of system design

We know of several system designs that would fit the micro-partitioned archetype based on a UTXO-like funds model. Two such examples are OpenCBDC (MIT Digital Currency Initiative 2022) and TODA (Gravitis, Goh and Toliver 2019). For our analysis, we choose the OpenCBDC platform for a few reasons. Firstly, it is purpose-built from first principles for a retail CBDC application, with design choices to optimize the system for that purpose. Secondly, it has a strong research focus, which allows us to consider and explore design ideas with the MIT DCI team in a way that might not have been possible in a production-ready system. Lastly, despite being an experimental research system, OpenCBDC comes with a flexible and powerful test framework that enables its deployment at scale and collection of performance metrics in a public cloud environment.

Two OpenCBDC architectures are based on the UTXO model: **two-phase commit (2PC)** and **Atomizer**. Briefly, Atomizer is built to materialize a global transaction order, while 2PC isn't. We choose the 2PC architecture for our analysis and experiments since it is not clear that a global transaction order is necessary for a retail CBDC intended for basic payments.[9] By relaxing that requirement, 2PC achieves higher performance. See **Figure 2**.

[9] Several researchers have shown that a global ordering isn't necessary for basic asset transfers as with UTXOs (Baudet, Danezis and Sonnino 2020; Guerraoui et al. 2019). However, if the account funds model is used, then an ordering becomes important. For example, two transaction orderings may differ in whether a balance drops below some threshold, incurring a charge. Also, if conditional logic is employed, as in contingent payments, a transaction order may be needed to achieve guarantees.

Figure 2: OpenCBDC system diagram for the two-phase commit architecture



Note: This figure is reprinted from Figure 3 in the Project Hamilton paper presented in 20th USENIX Symposium on Networked Systems Design and Implementation.
Source: Lovejoy et al. (2023)

The OpenCBDC 2PC architecture consists of three system components, each deployed in a variable number of instances. **Sentinels** act as the entry points to the system, accepting transactions submitted by wallets and validating them. Sentinels are stateless.[10] A **coordinator** takes a validated transaction, determines what changes must be performed to the system state to settle it and ensures that those changes are done atomically (i.e., they all happen, or none of them do) across a set of shards. Each **shard** maintains and updates some subset of the system state (known as the UTXO hash set, or UHS), as directed by coordinators. The set of all shards form the core of the system.[11]

We deploy the OpenCBDC test framework in an Amazon Web Services (AWS) environment:

- The core components of shards, coordinators and sentinels are deployed across three geographically dispersed AWS regions, so as to closely reproduce the network distances in a production deployment of a national system.
- Shards are configured with a replication factor of 3, i.e., each logical shard's information is replicated in real time across three replicas, one in each region. This too is chosen to mimic the extent of replication in a production system.
- All servers are set up as eight virtual central processing units (vCPUs), for uniformity and ease of testing. In a production system, the number of vCPUs would be adjusted up or down based on the processing load of each server type.

[10] Sentinels do not maintain information related to the monetary supply. As implemented, however, they do maintain some short-lived information for efficiency, such as open connections to wallets and recently seen transaction IDs.

[11] Some familiarity with OpenCBDC system design and transaction format is assumed for the following subsections of this paper.

- Traffic between data centres uses the default connectivity of the cloud provider that is shared with other customers. A production system could likely improve on this by using dedicated links.

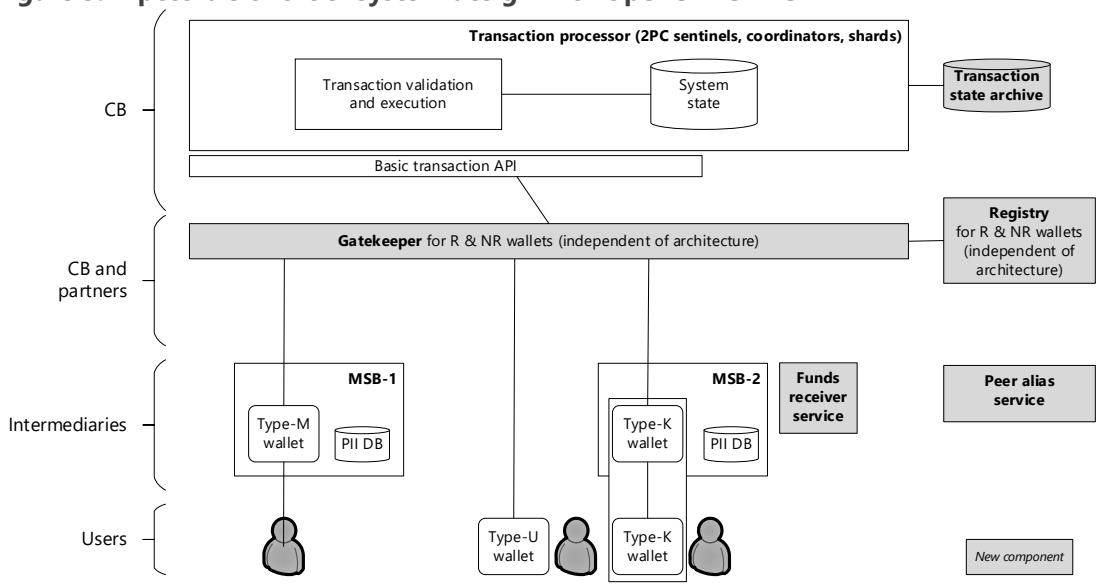
3.2 Two-tier model

In a two-tier model, the CB would be responsible for issuance, validation and settlement functions, while MSBs would provide customer-facing functions such as KYC and account management.

Figure 3 presents one possible system design built around OpenCBDC 2PC. The core transaction processor comprising the shards, coordinators and sentinels would be owned solely by the CB.[12] It would combine the validation and settlement engine as well as the core system state. The CB would provide access to the core through an application programming interface (API) that presents capabilities for basic payments to end-user wallets.

We describe below a few other components that would be necessary in a two-tier retail CBDC system.

Figure 3: A possible two-tier system design with OpenCBDC 2PC



Note: 2PC is two-phase commit, and CB means central bank. R means registered and NR means unregistered. Type-U wallets are fully in the user's custody, type-M wallets are fully in the money services business's (MSB's) custody. Type-K wallets keep keys in user custody and funds in MSB custody. API is application programming interface, and PII is personally identifiable information.

[12] One variation explored by the MIT DCI team is sentinels being operated by non-CB regulated entities, which is useful if their number is large (Stuewe et al. 2024). Such a design would have to contend with compromised sentinels outside the CB trust zone performing fraudulent validations. We assume in this analysis that sentinels are operated solely by the CB.

Wallets. We posit three types of end-user wallets, U, M and K, based on their custody arrangements for the spending keys and funds.[13] We then discuss how the wallets relate to registered (R), non-registered (NR) and semi-registered (SR) users (**Table 4**).[14]

Table 4: Wallet types

Wallet type	Spending keys	Funds	Transaction history	R user	NR/SR user
U	User	User	User		*
M	MSB	MSB	MSB	*	*
K	User	MSB	MSB	*	*

A type-U wallet is fully in the custody of a user device. It stores the user's spending keys and funds, so it has enough information to spend those funds. But if the device is lost, the funds are lost. Being outside an MSB's control or visibility, this wallet offers the highest privacy of the three types but is only suitable for NR/SR users since an R user is, by definition, known to an MSB.

A type-M wallet is fully in the MSB's custody. It too stores the user's spending keys and funds, so it can spend those funds. The user accesses it through an interface provided by the MSB. If a user loses their device, their funds and keys are protected (owing to the MSB's enterprise-grade safe storage, backup and recovery of wallet contents). But they must trust the MSB to not spend the funds without their approval since the MSB has custody of spending keys and funds.

A Type-K wallet is split into two parts—the user has custody of spending keys, while the MSB has custody of funds. To spend the funds, both entities must cooperate since neither has enough information to act alone. The user's funds are protected if they lose their device. However, the user retains authority over spending.[15]

All three wallet types could be supported in a feasible system design based on such an architecture. Regardless of the type, it is the wallet custodian's responsibility to maintain the necessary information about the wallet and its owner, such as the funds, full transaction history, etc., to allow the custodian to serve the owner's needs and satisfy any legal obligations.

Type-K and M wallets are suitable for R users because an MSB can perform compliance checks on them. But an MSB may offer those wallets to NR/SR users, too, by charging them fees despite not knowing their identities, as long as there are no compliance-related obligations.[16]

[13] In OpenCBDC 2PC, funds are the UTXO pre-images, while spending keys are cryptographic secrets to which funds are encumbered. An entity that has possession of both the funds and their spending keys can spend the funds.

[14] A **registered** user is one who has onboarded via a KYC process of the MSB, so their identity is fully known to the MSB; A **non-registered** user has onboarded anonymously, so their identity is not known; A **semi-registered** user has onboarded with minimal identity information, such as an email address or phone number.

[15] If a user loses their spending keys, funds encumbered to those keys would become unspendable. To recover them, the MSB could provide the funds to the CB, who would delete the corresponding UHS entries, mint new funds of equal value encumbered to the user's new keys, provide them to the MSB, and add their entries to the UHS.

[16] For example, an MSB could pay itself a monthly fee from a type-M wallet's funds, much like traditional banks automatically deduct fees, or lock a type-K wallet's funds until a monthly fee is paid by the owner.

Transaction status archive. We posit that the CB could maintain and publish an anonymous transaction status archive (TS-archive) through an API. The TS-archive would keep the transaction status by transaction identification (ID), but no other details, such as transacting endpoints, keys, amount, etc. Its sole purpose would be to allow queries for non-repudiation (see Section 3.6). Note, the TS-archive would not know a particular user's transaction history. That would only be known to the user's wallet custodian.

Gatekeeper and registry. We posit two services that are independent of the core architecture, gatekeeper and registry, for protecting the system against invalid endpoints or wallets. Every wallet, R or NR, would have its details (without personally identifiable information, or PII) recorded in the registry at onboarding time. The registry could also store per-wallet limits such as the number of transactions per day, maximum transaction value, etc. Transactions submitted to the system would first reach the gatekeeper, which would check with the registry that the wallet was onboarded and not blacklisted, and then apply limits, if any. More generally, the gatekeeper could perform checks that a single MSB would not be able to do, such as detecting fraud, preventing a single wallet operating from multiple MSBs, stopping distributed denial of service attacks, etc.

Both these services could be operated by the CB or its trusted partners. If the gatekeeper validates long-lived wallet credentials such as certificates, then ideally it would be operated by a non-CB partner so that the CB could not create transaction graphs by associating that credential with transactions.[17] If transaction values are hidden from system components, the ability to apply limits on amounts by the gatekeeper could be curtailed.

We note that the cost of the gatekeeper and registry components and the latency introduced would impact all system designs equally, and similar front-end services are common in high-throughput web-scale systems.

If architecture-specific checks are needed, e.g., limiting the number of UTXOs per transaction, they could be implemented in an architecture-specific component, such as a 2PC sentinel.

Peer alias (PA) service.[18] In a CBDC system, a sender must be able to determine a receiver's address. For example, the sender may provide an email or phone number of the receiver to get back the latter's public key or an address to make the payment to. The PA service would serve this need, independent of core architecture, and could return ephemeral (one-time use) or long-lived addresses. Its use would be optional since there can be alternate ways for a receiver to share their information with a sender, e.g., a QR code.

A PA service must be hosted by non-CB entities so that PII about users is not visible to the CB. Further, the service should ideally not be a centralized store because that would be an attractive target for hackers. Lastly, queries should be as private as possible, since a query by Alice for Bob's address is an indication that Alice is likely planning to pay Bob and, combined with other such queries, could probabilistically lead to the creation of a transaction graph, eroding privacy.

These goals can be achieved by partitioning a peer database between multiple MSBs, where each MSB is responsible for information about its hosted wallets. Advanced techniques such as

[17] The gatekeeper could validate anonymous credentials allowing users to prove that they are onboarded, without divulging their public keys or other attributes.

[18] The ideas paraphrased here are taken from a personal note shared by Sam Stuewe, "Payment Routing for CBDCs" (2024). It is a discussion of peer discovery and FR services.

private information retrieval (PIR) can be employed on each shard, so that a shard doesn't know which receiver is being queried.

More work is needed to understand how privacy-preserving techniques like PIR would work within a PA service that handles ephemeral information, e.g., single-use public keys.

Funds receiver (FR) services. In OpenCBDC 2PC, in addition to a settlement being recorded in the UHS, a sender must communicate the UTXO pre-images out of band to the receiver. More generally, funds information must be transferred wallet-to-wallet in micro-partitioned systems. Hence, such systems must consider how a payment is completed when a receiver is not present.

An easy, if sub-optimal, approach is to require the receiver to always be present for a payment. If the receiver is not present, payment cannot happen. This is overly restrictive and, as we describe below, it is possible to do better.

A second approach is to queue the out-of-band information for an absent receiver, so that when they are present later, they can retrieve the information and complete payment. The drawback of queuing messages is that completion of a payment may be delayed for an arbitrary period, awaiting an absent receiver. Further, a sender could use the same funds to issue payments to multiple receivers and, even though the system would guarantee that only one payment would succeed, it wouldn't be known until much later which receiver got paid (and which others saw their transactions fail). Hence, this approach doesn't achieve real-time settlement.

A third approach is for a receiver to designate an FR service in their absence to receive payments on their behalf. The sender would send a full payment to the FR, who would verify it and accept it on behalf of the receiver, completing the payment immediately. The receiver would retrieve their funds from the FR when present later. However, this introduces a risk that an FR could receive payments and then default on its obligation to a receiver. This risk is similar to that of a type-M wallet custodian defaulting and could be mitigated by regulation. We consider if and how an FR service relates to the three wallet types.

First, we note that it is incongruous for an NR user with a type-U (self-hosted) wallet to delegate custody of their funds to another entity. Hence, a type-U receiver must be present to receive payment.

A type-M receiver would be operated by an MSB with high availability, online except for planned or unplanned outages. The receiver would have custody of all the keys and funds needed to issue payments and receive funds, and hence wouldn't need an FR service.

A type-K wallet would fit well with an FR service operated by an MSB. Indeed, multiple FR services could be operated by multiple MSBs.

We note that there are similarities in a type-M hosted wallet and a type-K FR service. For example, both would hold their customers' funds and be highly available, so they could be developed together while maximizing code reuse.[19] But there are differences as well, so it is useful to keep them distinct in case future policy decisions require one of them to be

[19] One idea is to explore the FR service of each MSB acting as a fallback for other MSBs, increasing system resilience.

decommissioned without impacting the other. For example, a type-M wallet has to manage a large number of keys while a type-K FR has to manage certificates to prove it is authorized.[20]

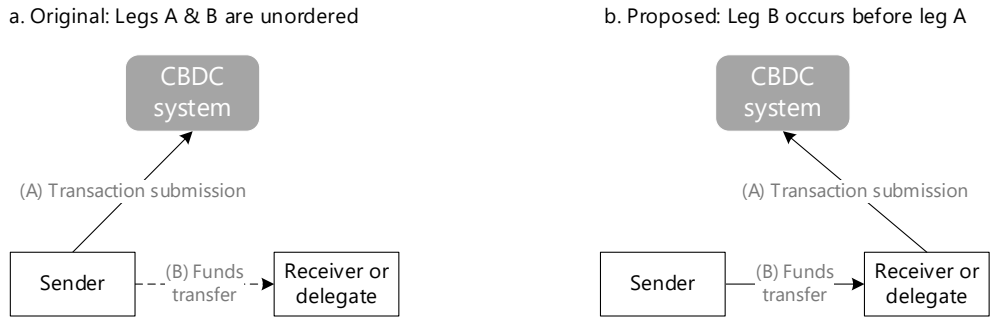
PII databases. Each MSB would maintain a database of PII about its retail customers. This information would only be known to the MSB, not the CB. Importantly, this database would include details of each user's full transactions along with other information the MSB needs (e.g., dates, times, IP addresses and geo locations, as needed for fraud detection or other purposes) to report on its customers for compliance purposes, and to provide them services such as their transaction history. The design and contents of the PII database are outside the scope of the CBDC system.

3.3 Payment flow and integration with retail payment systems

In this section we discuss a possible payment flow in a system such as OpenCBDC 2PC.

As noted earlier, in this type of micro-partitioned system, settlement requires two legs: (A), an update to the system state, and (B), a transfer of funds from the sender wallet to the receiver wallet. The wallet-to-wallet transfer, or leg B, is crucial since without it the receiver would not have possession of funds. The OpenCBDC system design does not specify when leg B will occur relative to leg A, only that it should happen out of band at some point (see **Figure 4**, panel a). We propose an ordering that would require leg B to occur before leg A (see **Figure 4**, panel b).

Figure 4: Two legs of a transaction in a system such as OpenCBDC 2PC



Note: CBDC is central bank digital currency. Leg A represents an update to the system state, and leg B is a transfer of funds from the sender's wallet to the receiver's wallet.

With this ordering, the sender wallet would send the full transaction to the receiver wallet instead of submitting it to the system. The receiver wallet (or a delegate acting on its behalf, such as an FR service) would validate it, store the funds, sign it to prove receipt of funds and submit it to the system for settlement.[21] The rationale of this ordering is to achieve strong non-repudiation guarantees, described later.

[20] Keys and certificates are discussed in the context of non-repudiation in Appendix A.

[21] A variation would be the receiver signing and returning the transaction to the sender to submit to the system. This would satisfy the ordering we intend to achieve, i.e., leg B occurring before leg A.

Figure 5 depicts a possible payment flow with this ordering, at a high level, generalized to type-K, type-M or type-U wallets from senders to receivers. Some steps below are applicable only to some wallet types.

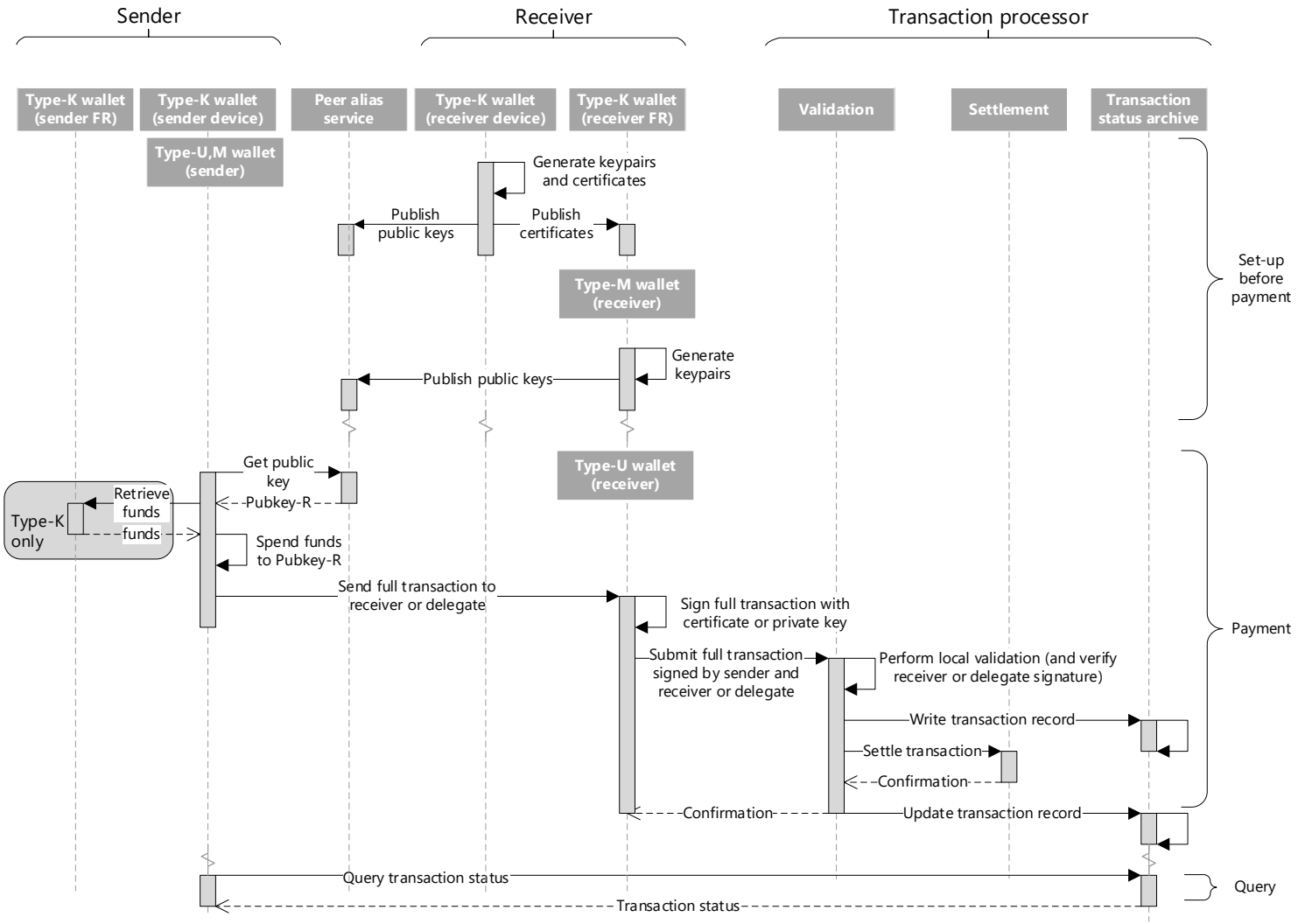
A type-U or type-M wallet has a single lifeline, labelled "sender" or "receiver." A type-K wallet has two lifelines labelled "device" and "FR," the former being in the user's custody and the latter in the MSB's custody.

First, consider the block labelled "Payment." The sender wallet queries the PA service to determine which public key to make the payment to. It then signs the funds to that key (a type-K wallet has to first retrieve its funds stored remotely). The signed funds are sent in a full transaction to the receiver wallet (or its delegate, such as an FR service), which validates the details, stores the funds, and signs and submits the transaction for settlement to the system. On settlement, the anonymous TS-archive is updated so that it can be queried later.

The block labelled "Set-up before payment" shows how the FR and PA services would be prepared by the receiver, which could occur weeks or months prior to payment.

Finally, the "Query" block shows how an authorized entity could, at any time after settlement, query the transaction status by its ID.

Figure 5: Payment flows from type-K, type-M and type-U sender wallets to type-K, type-M and type-U receiver wallets



Note: Type-U wallets are fully in the user's custody; type-M wallets are fully in the custody of a money services business (MSB); and type-K wallets keep keys in user custody and funds in MSB custody. FR is a funds receiver service.

Currently, in a typical retail payment scenario involving bank accounts, a funds transfer occurs between banks in the back end, not between a user and a merchant device.[22] In contrast, in a system such as OpenCBDC 2PC, funds must be transferred from the sender wallet (e.g., a smartphone) to the receiver wallet (e.g., the point-of-sale, or PoS, terminal or another entity), which would have to receive and safeguard funds. This arrangement is much like a cash register accepting and storing physical currency. This is more complex than how account-based systems function and raises several questions that need investigation, e.g., how to cancel and roll back a partially completed transfer.

It is certainly feasible to perform retail payments in a system based on wallet-to-wallet funds transfers, as shown in the sequence in Figure 5. However, it would entail integration work with payment service providers (PSPs), to update PoS terminals and entities in the payments flow to transfer funds through standard messaging formats such as ISO20022, put them into custody, and gracefully handle the boundary conditions, e.g., transaction timeouts, cancellations, missed messages, etc. We leave investigation of PSP integration to future work.

3.4 Integrity

3.4.1 Integrity of funds and wallets

In a micro-partitioned system, a single copy of the funds exists locally in the user's wallet (regardless of custody). Moreover, the wallet is responsible for construction and submission of a well-formed transaction to the core system. Since funds are sensitive assets and transactions are sensitive processes, they require protection while at rest and during processing. Standard techniques such as encryption and trusted execution environments can be employed. Use of secure hardware can further bolster protections in smartphones and support debit-card-style form factors.

Funds representation and transfer protocols rely heavily on cryptography and cryptographic primitives for confidentiality and integrity. The following properties must be satisfied to secure funds information at the periphery of a micro-partitioned system:

- **Funds and transaction authenticity**—Provided through digital signatures based on asymmetric cryptography to ensure that only the appropriate party with authorization to spend the funds can spend them.
- **Channel security**—Established through a combination of asymmetric and symmetric cryptography to negotiate a session key that is unique to every interaction and used only once.
- **Message confidentiality**—Provided through symmetric key algorithms to encrypt messages so that other entities spying on the channel cannot observe the contents.
- **Message integrity and tamper protection**—Provided through cryptographic one-way functions that generate hashes and message authentication codes so that the receiver can detect if messages are modified, tampered or replaced.

[22] For example, when a user taps their payment card or smartphone digital wallet at a point-of-sale (PoS) terminal, only an authorization message originates at the user device and is routed to the user's bank through the PoS device.

3.4.2 Integrity of the supply

A CBDC system requires a mechanism to ascertain the total amount of funds in circulation. Ideally, such a mechanism could be triggered arbitrarily by the issuing authority to conduct an on-demand audit of the currency. In the baseline OpenCBDC 2PC implementation, it was not feasible to aggregate the supply, so MIT DCI subsequently implemented two variants to enable it. We describe the variants below before summarizing the feasibility analysis.

Variants

Micro-partitioned systems can be based on a funds model of short-lived (e.g., UTXOs) or long-lived (e.g., bills) objects. OpenCBDC 2PC relies on the UTXO model. Each transaction consists of inputs and outputs. Inputs cease to exist once the transaction is settled. In this way, the model is very similar to UTXOs in a blockchain. However, unlike a blockchain, the core tracks only the set of valid outputs. With every transaction, new outputs are inserted and spent inputs are deleted. Validity of a UTXO is defined by its existence in the UHS. Previous transaction history or any other linkages between inputs and outputs are not stored in the core. In the baseline variant of the implementation, each unspent output is minimally stored in the core as a hash, with no other ancillary data such as its value.

Transactions in OpenCBDC 2PC are constructed by combining inputs and outputs. A full transaction utilizes the following distinct primitives:

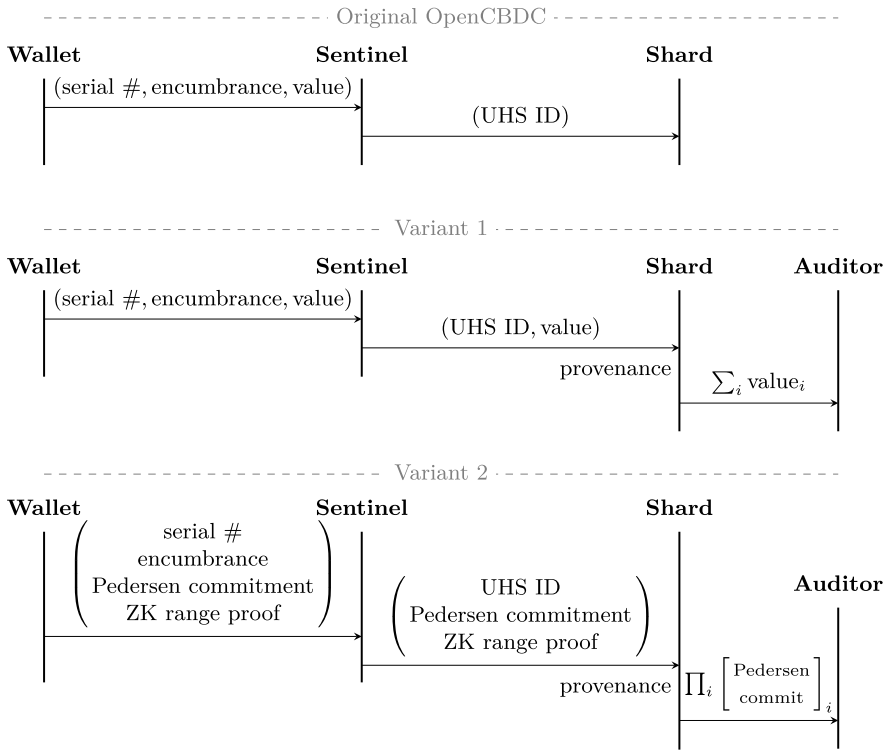
- **Digital signatures** prove ownership of funds and authorize the transfer.
- **Asymmetric cryptography** binds the recipient's public key to their portion of the funds.
- **Hashing** has a dual role: It acts as the proof of a valid output that can be compared with the inputs of a transaction, and it irretrievably masks the output values. The mechanism of constructing the transaction injects randomness into the outputs, guaranteeing that all hashes stored in the UHS are unique so there are no collisions, and so duplication of money becomes impossible through cryptographic means.

Two variants improve system auditability and correctness, either of which would suffice for a retail micro-partitioned CBDC (**Figure 6**). For every output in the UHS, in addition to the hash, the following is also stored, depending on the variant selected:

- **Variant-values**—A hash representing a UTXO in the UHS is replaced with a <hash, value> tuple. The first field is the UHS ID. The value (integer) is stored "in the clear."
- **Variant-PPA**—PPA stands for privacy-preserving audit. A hash representing a UTXO in the UHS is replaced with a <hash, Pederson commitment, range proof> tuple. The first field is the UHS ID. The second is the Pedersen commitment (PC), constructed on the pre-image to hide the value, while the third is the range proof (RP) computed against the commitment. The purpose of the RP is to prevent a user from committing negative values, which would lead to an unauthorized creation of money.[23]

[23] The initial implementation used the BulletProof scheme to construct proofs. A later implementation used a lightweight version called BulletProofs++ to reduce the proof size for efficiency.

Figure 6: Comparison of the baseline OpenCBDC with Variant 1 (values) and Variant 2 (privacy-preserving audit)



Note: This figure combines three figures from a working paper from the Massachusetts Institute of Technology Digital Currency Initiative. The authors of that paper refer to the baseline OpenCBDC as Original. UHS ID is the identification number for an unspent transaction output in the hash set.

Source: Stuewe et al. (2024)

Variant-values is lightweight and efficient but is less cash-like because amounts are known to the system. Variant-PPA uses additional cryptographic operations to preserve privacy at the expense of increased computation cost and storage space. Both approaches can be applied to micro-partitioned solutions in general, as PCs are efficient cryptographic constructs and can be implemented on lightweight wallets (such as smartcards) using existing technology.

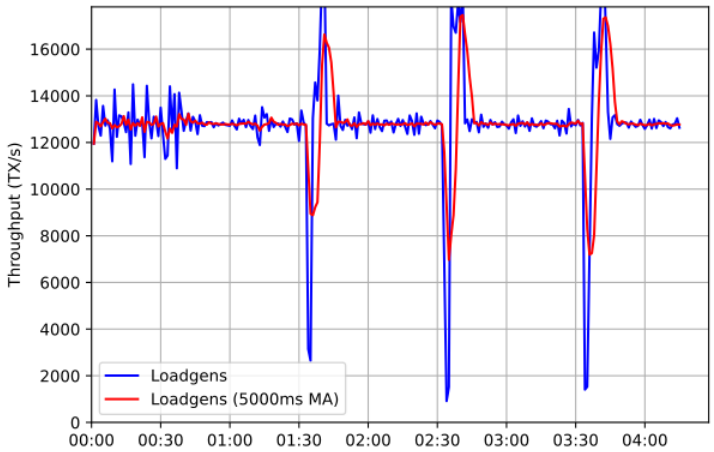
In both variants, an **audit** logging process runs at periodic epochs system-wide to construct the artifacts needed to calculate the total amount of money in circulation. The logging process runs automatically per epoch (a fixed time interval) and generates a log independently on each node. An out-of-band process would consume all logs generated in a specific audit to calculate the total supply in circulation and match it to the sum of all minting and destruction requests.

Crucially, for an audit to play a critical role in system integrity, it must be interleaved with the **delete** step. Hashes of spent outputs in a given epoch are only deleted from the UHS once the audit for that epoch is successfully completed. A successful audit implies that the sum of all valid outputs in circulation is equal to the predicted total, based on the list of issuance and redemption requests. While a variant-values audit reveals the total amount in circulation, the variant-PPA audit can only attest to a binary condition: whether the total matches the predicted, or not. In both cases, an additional process is needed to identify the precise transaction

resulting in an audit failure, in case a discrepancy is found, for forensic investigation. The introduction of a TS-archive (see Figure 3) could aid in tracing the faulty outputs to a unique transaction and associated source (wallet or MSB), although this analysis does not work out the details.

The frequency of audits represents a trade-off between security and performance, as spent hashes are not deleted from the UHS unless they are verified by an audit. In a high-throughput system such as OpenCBDC, spent hashes can quickly accumulate to saturate available memory. Therefore, audits occur frequently at a default interval of 60 seconds. However, each audit logging event materially impacts the system performance for a period of 2–5 seconds, during which transaction throughput is reduced to a fraction of its original performance. As an example, **Chart 1** depicts the system throughput from a test run on the values-based UTXO. The periodic troughs and spikes of throughput align with the frequency of the audit logging process. The system experiences a brief loss of throughput when the process first starts and then surges to catch up once logging is completed.

Chart 1: System throughput during audit logging with the variant-values implementation



Note: Throughput is given in transactions (TX) per second. Loadgens refers to the throughput as measured at load generators, which submit transactions to the system and collect responses from the system. 5,000ms MA is the moving average of the throughput over five seconds.

Source: Bank of Canada calculations

Although not shown explicitly, the behaviour of a variant-PPA is observed to be similar. We note that the logging process is memory-intensive and needs to be studied further to confirm long-term operational stability.

3.4.3 Sentinel attestations

Since the core components and services of a micro-partitioned system would be operated by trusted entities (as shown in Figure 3), standard security controls available to other centralized infrastructure systems can be used without modification, including access control, firewalls and threat-monitoring tools, among others.

OpenCBDC 2PC assumes a cash-like model, where minimal information is stored in the UHS. The sentinel is the first point of interaction for a wallet and the only point in the system where inputs and outputs are summed to generate an equal values proof. This approach imposes a heavy burden on the integrity of the sentinel. If a sentinel were to be compromised, the system would lack a secondary safety measure to detect the compromise, and it would become possible to double-spend or mint money without permission. This violates a core security principle around defence in depth, where two or more controls must safeguard a sensitive operation (in this case, a transaction approval). The following are possible attack scenarios:

- A malicious sender modifies outputs, so that they do not match the outputs the recipient is expecting, but the output total is still equal to the input total. We can consider this attack a theft of funds. Note that this is not exactly an attack since the sender is choosing to send the incorrect amount of funds to the recipient. However, the system has zero knowledge of what the sender and recipient agreed to.
- A malicious user submits the same UTXO as input in two separate transactions.
- A sentinel modifies the output to send funds to an alternate address, one that the attacker controls.
- A sentinel (either alone or in collusion with a malicious user) modifies the output address and increases the output amounts to mint money, sending a portion to the amount that the attacker controls.

The first risk scenario does not compromise the integrity of the system and, moreover, is precluded if transaction legs are ordered, as per Figure 5. The second scenario is mitigated by the fact that the shard checks the validity of the hash before updating the set. Since shards are replicated, for this attack to succeed an entire shard cluster would have to be compromised, which, if defence in depth is achieved properly, is difficult if not impossible. Therefore, our focus is on the third and fourth scenarios. In both cases, the underlying attack is to compromise a sentinel and make it behave maliciously.

The root cause stems from the fact that provenance is not established as a property of the system. Specifically, this means that, given a set of UTXOs (the UHS), it is impossible to inspect a UTXO and ascertain if it was created by, or is linked to, a valid minting operation at the CB. This gap is by design, as sets of links between a valid UTXO and a minting UTXO can lead to the creation of a transaction graph, eroding privacy. While provenance can provide strong guarantees, the common way of achieving this property is to create a link to previous transactions leading back to a minting operation, essentially a property that blockchains possess. However, blockchains incur a severe performance penalty in achieving provenance. What is desired here is a lightweight solution that achieves similar guarantees to consensus while still performing well.

Sentinel attestation (SA) is a proposed control to solve the problem in a scalable fashion. SA is a parameterized control, where $n>0$ represents the number of sentinels required to validate the transaction. Each sentinel in the chain will sign the transaction attesting to its veracity. (The final sentinel has the task of stripping out pre-images prior to forwarding the transaction to a coordinator.) When $n=1$, only the sentinel receiving the transaction will validate and sign. For $n>1$, additional sentinels will attest. The feature is disabled when $n=0$. If any one sentinel detects a discrepancy, the transaction is aborted. The SA approach is horizontally scalable (one need only add more sentinels). Note that the number of signatures that need to be checked is

tied to the attestation parameter, which may require an additional compute per coordinator (or more coordinators).

For a retail CBDC, an attestation value of at least n=2 is needed to appropriately manage risks, despite the impact of extra computing and communications on performance (more on this later). To mitigate some of the performance impact, the attestations can be consumed by the coordinators so that their validation in the shards can be turned off without incurring additional risk.

3.5 Non-repudiation

Non-repudiation is commonly understood as the guarantee that a sender of a message cannot later deny their involvement. In the context of a payment, we consider how a system such as OpenCBDC can enable a receiver to prove that they received funds from a particular sender and a sender to prove that they sent funds to a particular receiver.

We note that a system like OpenCBDC 2PC can prove these claims if it can satisfy four guarantees, G1 to G4, at the time of settlement.

G1: The system should be able to detect an attempt by an MITM (man in the middle) to modify the transaction that the sender constructed en route to the system.

G2: The system should be able to confirm that the receiver or another entity possesses funds (i.e., pre-images of outputs).

G3: The receiver or their delegate should be able to confirm that the funds are encumbered to the receiver (i.e., to keys belonging to the receiver).[24]

G4: The system should be able to confirm that the party who received the funds is authorized to receive them, ideally on the receiver's behalf.

If the two legs of a transaction are unordered (Figure 4, panel a), G1 is achievable but G2, G3 and G4 are not.[25] This is because when a transaction is submitted by the sender, the system can make no inference about what, if anything, a receiver or their delegate knows about it at the time of submission.

Instead, if the two legs are ordered (Figure 4, panel b), then G1 to G4 are all achievable for both RP (receiver present, type-U and type-M wallets) and RNP (receiver not present, type-K wallets) cases (see **Appendix A** for details). Wallets may use ephemeral or long-lived keys to receive payments. The former offer higher privacy but lead to higher demands being placed on some system components such as FR and PA services. We show that the four guarantees are feasible using ephemeral keys, so it follows that they are feasible using long-lived keys as well.[26]

---

[24] This is easily proven if the receiver themself receives the funds. But if a delegate receives them, this should be provable without divulging the receiver's spending credentials to the delegate.

[25] G1 can be achieved by the sender signing an artifact that covers the entire transaction. In OpenCBDC 2PC, the sender signs the transaction identification (TxID) with the spending key of each input. Hence, any change to the transaction inputs or outputs would invalidate the sender's signatures and be discoverable.

[26] How wallets manage their keys involves many choices. Keys can be ephemeral or long-lived, generated randomly or deterministically, etc. We don't assume one choice or the other (wallets being outside the core system, designed by the private sector) except to note that to enable non-repudiation, wallet custodians would need to store sufficient data to associate their users with the public keys used in payments, regardless of how keys are generated.

We briefly outline how the two non-repudiation claims can be proven.

A sender can prove that they sent funds to a particular receiver. Suppose that Alice claims she paid Bob, but Bob disputes it. Alice would present the full transaction of the payment from her wallet, compute its transaction identification number (TxID) and query the TS-archive for its status. Assuming the status says "settled," it would prove that *both legs* of the transaction had been completed and that the receiver or a delegate confirmed receipt of funds for the transaction amount. If a delegate received the funds, it would also prove that the receiver had authorized that delegate to act on their behalf.[27] To confirm that the receiver was Bob, Alice would examine the transaction to find the public keys to which outputs were encumbered and provide them to Bob's MSB, who could check if those keys belong to a wallet associated with Bob.[28]

A receiver can prove that they received funds from a particular sender. Suppose that Bob claims that he was paid by Alice, but Alice disputes it.[29] As above, Bob would present the full transaction, compute its TxID and query the TS-archive for its status. Assuming the status says "settled," it would prove that both legs of the transaction had been completed and, importantly, that the transaction as constructed by the sender wasn't modified en route to the system. To confirm that the sender was Alice, Bob would examine the transaction to find the public keys to which the spent inputs had been encumbered and provide them to Alice's MSB, who could check that those keys belong to a wallet associated with Alice.

In summary, non-repudiation is feasible in systems such as OpenCBDC 2PC. As in traditional account-based systems, it would require MSBs to maintain detailed transaction records that are not known to the CB's settlement system. However, unlike in traditional systems, the CBDC system would need to prove claims about not only the core system update, but also the wallet-to-wallet transfer, which adds complexity. These ideas would be useful to investigate in collaboration with private-sector parties in a future phase.

3.6 Scalability

The user base of a CBDC system would likely be in the tens of millions, covering the national population, but could go into billions if future payments trends such as internet of things and micropayments materialize, and devices act as autonomous transaction endpoints. The **throughput** requirement (in transactions per second, or TPS) would likely be a few thousand TPS at the lower end, matching existing electronic payments networks.[30] However, use cases for emerging payments and high adoption could push this into the hundreds of thousands or millions of TPS. The **latency** needed (i.e., the time from submission to settlement for one

[27] By receiver, we don't mean a particular identity or person, but rather the entity that holds the private key corresponding to the public key to which funds are encumbered.

[28] A claim is provable only if the party being investigated (e.g., the receiver Bob in this claim or the sender Alice in the second claim) operates a type-K or type-M wallet, since an MSB has visibility only to those wallet records. It is not possible to prove claims about parties who operate type-U wallets because their records are not visible to any MSB.

[29] For example, a party could deny being the sender of a payment related to illegal activity.

[30] Several electronic payment networks in use today clear in real time, with settlement being deferred. In contrast, CBDC designs are in most cases envisioned as supporting real-time settlement.

transaction) would be in the order of seconds to be suitable for retail PoS payments.[31] Hence, the ability to scale the system up to handle users and throughput at these ranges while preserving latency suitable for retail use—at a reasonable cost point—is foundational to a CBDC technical design.

Scalability testing assesses the ability of a UTXO-based, micro-partitioned architecture to scale linearly using OpenCBDC 2PC as a representative example.[32] We note that not all micro-partitioned systems are alike and, even in similar systems, particular design choices can lead to marked differences in performance. Nevertheless, observations on OpenCBDC 2PC are useful to gauge the scalability achievable in systems where sound design choices are combined with the key idea of micro-partitioned systems, i.e., to minimize the state in the system core by pushing it to entities at the system periphery and optimize how that core state is updated to maximize efficiency.

We experiment with three variants of OpenCBDC 2PC, the baseline and two auditability variants (Stuewe et al. 2024):

- **Baseline**—Records only hashes in the UHS. It is expected to be the best performer.
- **Variant-values**—Provides auditability with values in the clear. It is less performant than the baseline because it increases the workload on shards to run the periodic audit procedure to aggregate the supply.
- **Variant-PPA**—Implements privacy-preserving audits, so that values are hidden from the settlement system. It is the least performant of the three, due to the cryptographic work during validation and aggregation and to a larger transaction size.

We set the UHS size (i.e., size of monetary supply) to 50 million UTXOs for the baseline, 10 million for variant-values and 0.5 million for variant-PPA, all tested with one to eight logical shards. Note that a production system would almost certainly have to support a larger CBDC supply.[33] The baseline and variant-values implementations could scale up to a larger supply by using more logical shards (say, 256) or by vertically scaling shards (from 8 to 32 or 64 vCPUs). It is unclear how far up the variant-PPA implementation can scale.

We do not implement the TS-archive as posited in the two-tier model for testing. This would, when implemented, add overhead to the latency, although it could be minimized by using a distributed key-value store that supports very fast write speeds.

3.6.1 Comparison of three variants

We experiment with three variants of the system design by measuring throughput (in TPS) and latency (in milliseconds). Throughput is the number of transactions settled, measuring how fast core system updates can be completed. Latency measures the time from the submission to the confirmation of a transaction.

---

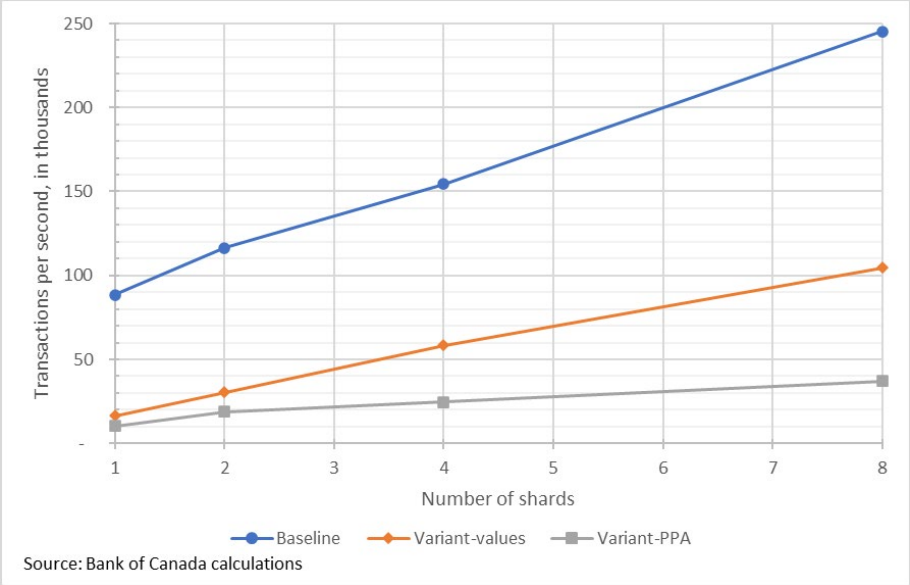
[31] For our experiments, we measure only the latency of a core system update, i.e., the UHS update. Since a wallet-to-wallet transfer is not implemented, we do not measure its latency.

[32] *Linear* scalability means that when the system capacity (e.g., compute) is increased by a factor "c," its throughput will increase by a constant multiple of c. Within limits, such a system can be scaled up to any desired throughput by adding capacity.

[33] For comparison, there are approximately 3 billion physical currency bills in circulation in Canada (Bank of Canada 2023).

Chart 2 shows the throughput for the three variants, scaling from one to eight logical shards. The baseline variant scales linearly to approximately 250,000 TPS.[34] The two auditability variants, owing to additional computation work, achieve lower absolute throughputs and also lower factors of scaling. In particular, variant-PPA is observed to struggle to scale, despite using a smaller UHS size.

Chart 2: Throughput of the three variants



Source: Bank of Canada calculations

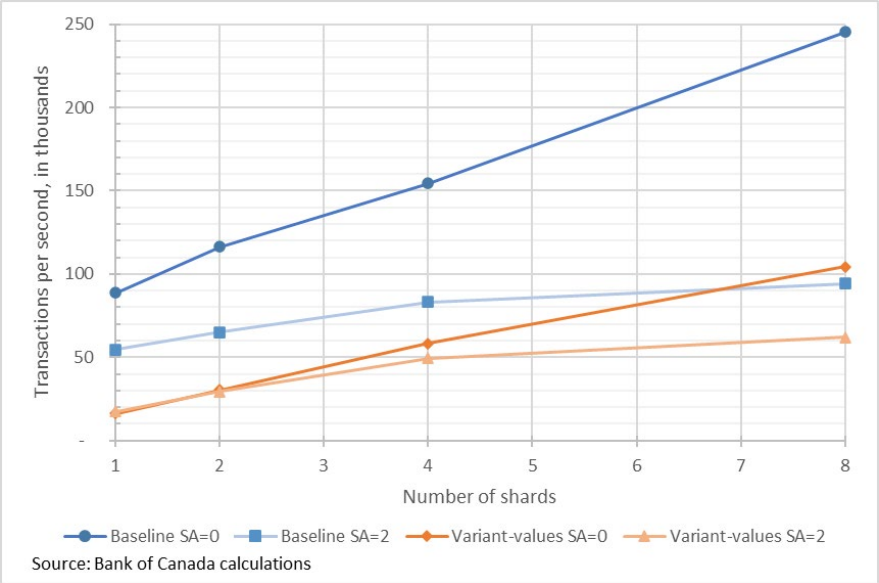
3.6.2 Assessment of sentinel attestations

As described earlier, SA are a way to counter threats that arise from a compromise of sentinels. The SA feature is a good example of the additional computational burden that could be placed on the core system as it evolves to meet future requirements.

We test this feature by setting SA=2 for the baseline and variant-values implementations, summarized in **Chart 3**. We observe that throughput is markedly lower for SA=2 than for SA=0, and the degree of scalability is also affected. It is not clear if linearity is affected; this requires investigation.

[34] This variant scales much higher, beyond 1 million TPS, for larger configurations, as described in the original OpenCBDC white paper by Lovejoy et al. (2023).

Chart 3: Throughput with sentinel attestations



Source: Bank of Canada calculations

We summarize our observations about system scalability:

- **The baseline variant demonstrates linear scalability**. The baseline throughput scales linearly as capacity is increased from one to eight shards, without significant degradation of latency.
- **The degree of scalability is impacted by supply audits.** Auditing the supply entails increased storage and computational overhead in the core compared with the baseline, leading to a predictable performance impact. Variant-values scales linearly or close to it, although at a lower rate than the baseline, if SAs are disabled.
- **Scalability of auditing with privacy-preserving commitments remains unproven**. Auditability with commitments is a compelling idea, as it would allow the system to detect discrepancies in the supply *without knowing the amounts in circulation*. However, it struggles to scale linearly, evidently due to the burden of cryptography. These are preliminary results, so optimizations would likely improve performance. However, they point to the trade-off that packing more functionality into the core system would likely render it less scalable.
- **The size of the monetary supply impacts auditing performance**. The auditability variants periodically aggregate the entire supply. During this process (called audit logging), we observe a latency degradation in our testing.[35] By varying UHS size in the range of 1 million to 50 million UTXOs for the variant-values implementation, we confirm that a smaller UHS results in less severe latency spikes. In a production system with a larger UHS, audit logging would have to be tuned carefully, so latency is kept within required limits.[36] We are not

[35] With an audit interval of 60 seconds, we observe that latency spikes last around 10 seconds or less, with peaks of 3–4 seconds. This means that, in those 10 seconds, latency could be as high as 3–4 seconds, although on average it is 10% of that (around 400 milliseconds).

[36] In a production system, other ways to optimize the CBDC supply could be explored. For example, wallets could autonomously consolidate small-value UTXOs during off-peak hours.

able to test variant-PPA with a UHS larger than 500 thousand UTXOs, so it remains to be shown that it can support a production-size UHS.

- **Determining the optimal deployment footprint requires continual tuning**. In a distributed system composed of instances of several components (such as shard, coordinator and sentinel), the number of instances and their relative ratios need to be established through careful monitoring and analysis. With every change in the system, behaviour needs to be assessed to understand if bottlenecks in the system have shifted and relative ratios have changed, to adjust capacity.

3.7 Privacy

A retail CBDC system will involve the collection of users' PII to comply with legal requirements, such as anti–money laundering (AML) and anti-terrorist finance (ATF) laws. PII related to a CBDC falls into three key categories: identity data for KYC purposes, transaction data for settlements and metadata to detect suspicious activities. Collection and management of PII data will play a key role in the level of privacy offered to users. In general, CBDC archetypes that allow end users rather than institutions to hold information (e.g., the direct and micro-partitioned) can more easily achieve higher privacy than the other archetypes. However, it is crucial to adopt a privacy-by-design approach to safeguard the privacy of end users throughout the design and development of an end-to-end retail CBDC system.

3.7.1 Privacy design

In designing a micro-partitioned and UTXO-based, end-to-end retail CBDC system, we introduce several key privacy-focused design features of OpenCBDC 2PC and additional components. These features are:

Identity decoupled from transactions—User identities are separated from transactional data using unique cryptographic identifiers. This prevents linkage to individuals, enhancing anonymity and protecting personal information. In our two-tier design, PII is managed exclusively by the MSBs (acting as wallet custodians) or the end user in the case of a self-custody wallet.

Minimal data retention for core transaction processors—The core transaction processor components (sentinels, coordinators and shards) have access to only the non-PII data (e.g., transaction hashes), enabling the CB to settle transactions without accessing end users' PII. This ensures a high level of privacy for end users from the CB.

Ephemeral keys for transaction signatures—Transaction signing keys are generated dynamically for each specific transaction, offering enhanced privacy compared with long-term keys. This approach ensures k-anonymity, where k represents the set of wallets actively participating in the system at any given time.

Self-custody wallets (type-U)—The micro-partitioned architecture inherently supports self-custody wallets, allowing users to maintain direct control over their funds and keys. This setup ensures a high level of privacy, particularly for NR users.

Privacy-preserving transactions—MIT DCI proposes and implements variant-PPA, based on Pedersen commitments and zero-knowledge range proofs, to hide transaction amounts from core settlement processor, while maintaining auditability. This solution improves privacy beyond the level offered by the baseline OpenCBDC 2PC design.

Privacy-by-design in supporting components—The design of additional components such as the TS-archive, gatekeepers and registry adheres to privacy-by-design principles by minimizing the collection and retention of PII. This is achieved through a careful analysis of data flows and models, ensuring that only the data essential for core functionality are collected and stored. In our design and analysis, we design these components to minimize data exposure while maintaining the operational integrity of the end-to-end CBDC system.

3.7.2 Privacy analysis

We also conduct a thorough privacy analysis of various design options, using rating-based techniques (Arora and Darbha 2020) from the following two perspectives:

Visibility of holdings and transaction data—This perspective encompasses information such as the sender address, receiver address and transaction amount. Examining who can access this data during transaction processing is crucial for ensuring privacy. **Table 5** presents the privacy rating for this solution. A rating of 3 indicates the entity has no visibility and stores no data, rating 2 indicates the entity has visibility but has no storage requirement and rating 1 indicates both visibility and storage are required.

Table 5: OpenCBDC 2PC visibility ratings for the three variants and cash

Solution	Wallet type	Central bank					Payer MSB					Payee MSB				
		H		T			H		T			H		T		
		O	B	S	R	A	O	B	S	R	A	O	B	S	R	A
OpenCBDC 2PC baseline variant	U	3	3	2	2	2	3	3	3	3	3	3	3	3	3	3
	K/M	3	3	2	2	2	1	1	1	1	1	1	1	1	1	1
OpenCBDC Variant-values	U	3	3	2	2	1	3	3	3	3	3	3	3	3	3	3
	K/M	3	3	2	2	1	1	1	1	1	1	1	1	1	1	1
OpenCBDC Variant-PPA	U	3	3	2	2	3	3	3	3	3	3	3	3	3	3	3
	K/M	3	3	2	2	3	1	1	1	1	1	1	1	1	1	1
Cash		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

Note: MSB is money services business. H means users' holdings, and T means transactions. O is the owner, B is the balance, S is the sender, R is the receiver and A is the data amount. A rating of 3 indicates the entity has no visibility and stores no data, rating 2 indicates the entity has visibility but has no storage requirement and rating 1 indicates both visibility and storage are required.

We make the following key observations based on these ratings:

- The CB has no access to information about users' holdings, such as their identities and total balances. This approach ensures a high degree of privacy from the CB, though not necessarily from intermediaries.
- The CB does not have access to transaction or holdings data for any type of wallet. The choice of wallets (type-U, type-K or type-M) has a big impact on users' privacy from intermediaries, but it has no impact on privacy from the CB.
- The CB, running sentinels, has access to the sender and receiver addresses and the value of the transaction. Sender and receiver addresses are only needed by the sentinels to verify the correctness of the transaction at the time of processing, and they don't need to store this information.

- OpenCBDC variant-values stores the transaction amount to support an operation to audit the supply, i.e., aggregating all values in each shard to determine the total currency in circulation. However, this variant reveals the values of individual UTXOs to the core, which weakens privacy. Note that this variant does not affect user privacy with respect to intermediaries; it impacts privacy only in relation to the CB.
- OpenCBDC variant-PPA is intended to strengthen privacy while allowing the supply to be audited. Instead of sending values in the clear to sentinels, cryptographic commitments to the value are sent instead (a Pederson commitment and a range proof). These commitments, instead of plain values, are validated and stored in the core. The commitments can be added to arrive at a cryptographically hidden value that represents the aggregate currency in circulation and can be compared with an expected aggregate value, while keeping values hidden from the core. Importantly, values are also hidden from sentinels, so they can no longer see amounts. This provides a very high level of privacy to the users from the CB.
- Transaction and holding data for NR users with type-U wallets are inaccessible to any entity, ensuring an exceptionally high level of privacy for these users.

Ability to generate spending profiles—This perspective focuses on the potential to link users to individual transactions and establish connections between multiple transactions. By understanding this capability, we can assess the risk of creating detailed spending profiles, which may compromise user privacy. **Table 6** present the privacy ratings for this solution, based on the criteria in **Figure 7**.

Figure 7: Criteria for the spending profile ratings

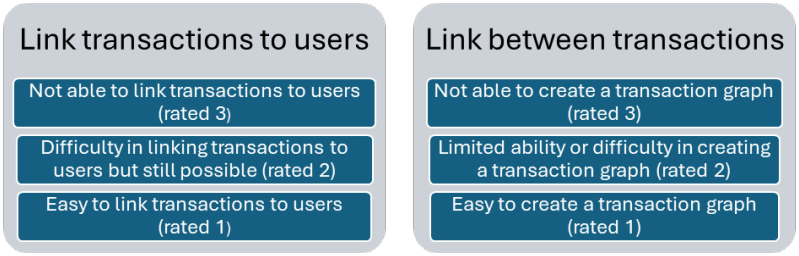


Table 6: Spending profile rating for variant-PPA

Solution		Central bank			MSB			
		Link tx to NR and R users	Tx graph	User profiling – R and NR users	Link tx to R users	Link tx to NR users	Tx graph	User profiling – NR users
OpenCBDC 2PC baseline variant	U	2	2	2	NA	3	3	3
	K	2	2	2	1	1	2	1
	M	2	2	2	1	1	2	1
OpenCBDC 2PC Variant-PPA with ephemeral keys	U	3	3	3	NA	3	3	3
	K	3	3	3	1	1	2	1
	M	3	3	3	1	1	2	1

Note: PPA means privacy-preserving audit, and MSB means money services business. Tx is transaction, R users are registered and NR users are unregistered. U, K and M are type-U, type-K and type-M wallets, respectively.

We make the following key observations, based on the above ratings:

- Sentinels operated by the CB have access to transaction data and the ability to store them. However, they are not required to retain these data after processing. The use of ephemeral keys ensures that the CB cannot easily link transactions to specific users and doesn't have the ability to build user spending profiles.
- MSBs have access to the transaction history of R users with type-K and type-M wallets, enabling those MSBs to link transactions to users and build spending profiles. The CB, however, does not have this capability for any user.
- From the user profiling perspective, the choice of wallet type (U, K or M) has a big impact on users' privacy from MSBs but has no impact on privacy from the CB. This observation is similar to the visibility analysis of transactions and holdings, above.

3.8 Compliance

The micro-partitioned CBDC based on the OpenCBDC two-tier architecture solution involves various datasets managed by multiple entities. These include intermediaries responsible for user onboarding and managing customer transaction history. Each intermediary maintains data for their own users and is responsible for ensuring compliance with AML and ATF laws.

3.8.1 User onboarding

The CBDC system would encompass several onboarding processes tailored to meet diverse user needs and business objectives. These processes range from onboarding procedures done entirely online to in-person interactions. KYC procedures during onboarding are a fundamental component of AML and ATF laws. Under these laws, financial institutions and other regulated entities must verify the identities of their customers and keep records of the relevant information.

For the two-tier CBDC systems, it would be up to the individual intermediaries to design the KYC solution for their R users. These intermediaries would have several options, ranging from developing a custom solution to utilizing KYC as a service. Intermediaries can leverage this opportunity to attract new customers by offering convenient onboarding techniques. In this setup, intermediaries are fully responsible for ensuring compliance, with the CB not involved in this process. The architecture supports this model by providing intermediaries access to the necessary data, enabling them to fulfill their compliance obligations.

For NR users with type-U wallets, onboarding would involve obtaining the wallet without providing any identity information or needing to interact with an intermediary.

3.8.2 Rule-based compliance

Rule-based compliance, also known as prescriptive or specific compliance, relies on a set of explicit, detailed rules, guidelines and regulations that dictate specific behaviours and actions to follow. For example, any transaction above $10,000 must be reported to the regulatory authority.

In this solution, for R users with type-M and type-K wallets (both senders and receivers), intermediaries have full visibility into the transactions and identities of their users. The CB does not have access to this data. Therefore, intermediaries are responsible for applying the necessary rules and reporting any violations to the appropriate authorities.

For NR users, potential rules to prevent illicit usage (e.g., transaction limits) are enforced by the gatekeeper component before the transaction is submitted to the transaction processor (i.e., the sentinel for OpenCBDC) for settlement. The gatekeeper would reject any transaction that violates these rules.

3.8.3 Principle-based compliance

Principle-based compliance, also known as outcome-based compliance, emphasizes broad objectives that guide behaviour and decision-making without dictating specific processes or procedures.

In this solution, intermediaries bear the responsibility for implementing compliance measures for their users. Intermediaries can deploy the latest techniques, such as machine learning models, to detect suspicious transactions based on patterns and anomalies. These technologies can enhance the effectiveness of compliance operations by identifying potential risks and fraudulent activities in real time, allowing for prompt and appropriate action. Historical transaction data managed by intermediaries can also be used for in-depth analysis to detect suspicious activities over time. The data can help uncover trends and patterns that might indicate illicit behaviour.

For NR users with type-U wallets, principle-based compliance is not applied. These wallets are designed to function similarly to cash, where transactions can occur without the need to verify identity or adhere to compliance principles.

In summary, two-tier systems such as OpenCBDC 2PC allow intermediaries to handle compliance activities for R users by utilizing customer data (KYC, transaction and meta data), much like in traditional account-based systems. They can adopt custom-built compliance frameworks or leverage third-party compliance-as-a-service solutions to fulfill regulatory requirements effectively. This flexibility allows intermediaries to enhance user experiences and attract new customers through convenient onboarding processes. For NR users, the system allows the implementation of appropriate controls to minimize illicit activities.

3.9 Resilience

The total system state in OpenCBDC 2PC exists in two parts: in the core system, as contents of the UHS, and in the system periphery, as funds information in wallets. The core system state is in the CB's or operator's control, while the wallet state is in the end user's control, typical of how the state is organized in micro-partitioned systems. We discuss how the system state can be made resilient, i.e., protected against loss in the event of a system failure.

3.9.1 Resilience of funds and wallets

The part of the system state pertaining to the supply is stored in the form of UTXO pre-images in end-user wallets.[37] This state is in the control of wallet owners and, if lost, results in a loss of funds for the owners since the funds do not exist elsewhere in the system. However, the loss of one user's funds does not impact other users' funds or the integrity of the supply.

[37] End-user wallets would contain other information that is not system state, e.g., credentials such as spending keys, records of transaction history and so on.

If a wallet is type-M or type-K, an MSB would have custody over its funds. The MSB would presumably use institutional backup and recovery mechanisms that enterprises already employ for critical systems they operate, to safeguard the contents of their customers' wallets.

If a wallet is type-U, then its funds would be in the custody of the user, on whom the onus would rest to prevent a loss. Vendors of smartphone wallet software could feasibly build backup and recovery mechanisms into their apps to allow regular backups of funds to the user's cloud account. These mechanisms could be designed to minimize the risk of funds being stolen by encrypting backups at rest and ensuring that spending keys are not backed up with funds.

In an open and competitive ecosystem, users would have a choice of many wallet types from vendors. A privacy-conscious user might choose a type-U wallet that provides backup and recovery capability that they deem as adequate for their needs. Another user might opt for a type-M or type-K wallet, giving up some privacy in return for enterprise-grade institutional backup and recovery of their funds.

Hence, resilience of funds in user wallets is certainly feasible from a technical standpoint, but the exact form in which it is realized would depend on the ecosystem and market factors.

3.9.2 Resilience of the core UHS state

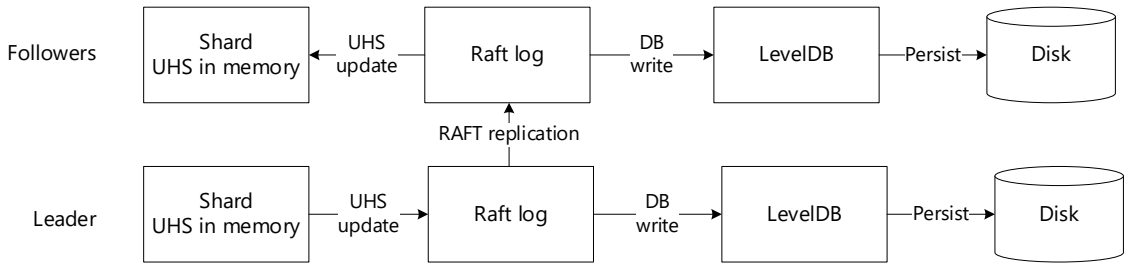
We consider how the core system state can be made resilient, so that it is preserved and recoverable without loss of data in the event of an outage. The core system state comprises information about unspent funds in the supply and is the crown jewel of the system. A data loss would be catastrophic since it would manifest as users being unable to spend their funds, which would have been incorrectly deemed by the system to be invalid.

In the OpenCBDC 2PC design, the UHS is maintained in memory on each shard instance, rather than in a persistent store. This makes UHS updates fast. The UHS is partitioned between several logical shards, each of which stores a part of the UHS. A logical shard cluster contains multiple physical shards that replicate its UHS state among themselves. The physical shard designated as the leader produces and distributes state update messages to follower shards who consume and apply them, serving as up-to-date replicas of the shard state. A state replication protocol (Raft) propagates state updates across a cluster.

In our experiments, we deployed three physical shards per logical shard, i.e., the UHS of each logical shard was replicated 3x across the three data centres (DCs).

Figure 8 shows the sequence of steps for a UHS state update to be applied by the leader and propagated to followers in a logical shard cluster.

Figure 8: Steps to record a UHS state update in a logical shard cluster



Note: UHS is the hash set of all unspent transaction outputs. Raft is a replication protocol, and LevelDB is a database.

First, the leader writes the UHS update to both its in-memory UHS and its Raft log. This update is Raft-replicated to all followers in the cluster (in the figure, only one follower is shown, for clarity). Each shard writes the update to its local LevelDB (Google Inc. n.d.) database as a new Raft log entry.[38] The database instance saves entries periodically to a disk at a cadence specified in its configuration.

To settle a transaction, the leader shard writes the update to its in-memory UHS and its local Raft log, which is replicated to the Raft logs of followers. It then marks the transaction as complete via a reply to the coordinators. At this point, the update is available in the in-memory UHS of both the leader and followers, and the transaction is considered settled.

We consider two failure scenarios—a partial outage and a total outage—and discuss the recovery time objective (RTO) and recovery point objective (RPO) achievable in each case. **RTO** is the time after an outage needed to restore full service. Ideally, this should be in the order of a few hours at most, e.g., two hours, since users will see the system as unavailable during this time. **RPO** is the amount of time leading up to the outage for which data has been lost. For a feasible design, this should be zero. In other words, once a transaction has been confirmed as settled, its data should always be recoverable from persistent storage. If the RPO is non-zero, say 30 seconds, then transactions already declared as settled in that 30-second interval would appear after recovery to not have been settled.

Recovery from a partial system outage

We consider a partial outage as an outage of one or more DCs, so that *at least one replica of each logical shard remains operational*.

First, we consider the RPO. A transaction's state is replicated to all followers before it is confirmed as settled. So, as long as at least one replica in a cluster remains operational, the state of settled transactions would be available in the in-memory UHS of that replica. The leader, even if newly elected, would possess the UHS state of all settled transactions. Hence, the system could recover from a partial system outage with an RPO of zero, i.e., without loss of state.

Next, we consider the RTO. On a failure event, the Raft protocol discovers the outage and, if needed, elects a new leader for affected clusters. We observed these steps occurring in 10–20 seconds, during which time some transactions that are in flight may fail to settle. They can be resubmitted to the system a short time later for settlement, after the leader election. Hence, an RTO in the order of seconds or minutes is feasible with a similar state replication mechanism.

Recovery from a total system outage

We consider a total system outage as *an operational loss of all replicas of a logical shard*, e.g., due to a widespread power outage.

Recovery from a total outage would require reconstituting the UHS state from the persistent store. The only persistent store related to the UHS state is the set of Raft logs containing the events recorded since the start of the log. On a restart, a shard reads and replays its Raft logs to recreate the state as recorded in those logs.

First, we consider the RPO. To achieve a zero RPO, the system would need to guarantee not only that a transaction's state has been replicated across a cluster, but also that it has been

[38] The OpenCBDC design uses a local LevelDB instance on each physical shard, so that the LevelDB instances in a cluster are independent and disconnected.

written to the persistent store in all shards of the cluster, before it is confirmed as settled. We are unable to confirm this in our testing. Depending on the database technology and configuration, it may be theoretically possible in a system design that a small-time window exists during which this guarantee isn't met. If so, data loss and a non-zero RPO could result, especially if the transaction volume is large. It would be vital in a production system to confirm this. If design changes are needed to achieve this guarantee, their performance impact would depend heavily on the replication protocol and the database technology.

Next, we consider the RTO. In the current design, Raft logs grow unbounded, so the time to replay them, and hence the RTO, is theoretically unbounded. This would be unacceptable in a production system. To make state recovery from the persistent store feasible and practical, one possible idea is to snapshot the UHS state periodically, say, every few hours, and keep only Raft logs subsequent to the latest snapshot. It may be possible to incorporate snapshotting into the audit log process since both tasks entail walking the contents of the UHS periodically, although the performance impact of such a change is unknown. Recovery would involve first reading the latest snapshot to create an instance of a UHS corresponding to it, then applying updates from subsequent Raft logs to the UHS.[39] The frequency of snapshots would need to be tuned, while considering the time to replay events from Raft logs, to achieve an RTO of no more than a few hours.

Section 4: Lessons Learned

We summarize the key lessons learned regarding feasibility of micro-partitioned architectures such as OpenCBDC 2PC for an online retail CBDC system.

A design like OpenCBDC 2PC aligns well with a two-tier system. The design allows a clean separation of CB functions (minting, validation and settlement) from MSB functions (KYC, wallet management and compliance). We posit three types of wallets and show that they can coexist, supporting both user custody and institutional custody of funds in one system. We describe other necessary components such as a peer alias service for peer discovery, a funds receiver service to enable receiver-not-present payments, etc., and show how they could operate within a two-tier system.

Achieving strong non-repudiation guarantees requires the system to be able to confirm both legs of a transaction. A key attribute of micro-partitioned designs such as OpenCBDC 2PC is that a transaction involves two legs, a core update and a wallet-to-wallet transfer. We describe a modified payment flow that enables the system to establish completion of the inter-wallet transfer along with the core update. We demonstrate its feasibility for transactions from any wallet type to any wallet type. We show that this modified flow, supported by the CB's transaction status archive and MSBs' PII databases, can enable the system to achieve strong non-repudiation guarantees to resolve payment disputes between senders and receivers. In other words, a sender can prove that they paid a specific receiver, and a receiver can prove they were paid by a specific sender.

The use of a system such as OpenCBDC 2PC for retail payments would require enhancements in the payments ecosystem to accommodate complex payment flows. The current retail

[39] Raft logs would also contain other entries, e.g., protocol messages of leader and follower designations. All entries would need to be replayed in sequence to update a node to the latest state as known in Raft logs.

payments ecosystem is based on funds being represented in account balances at institutions and a transaction resulting in balance updates. To use a micro-partitioned system like OpenCBDC 2PC for retail payments, the ecosystem would have to adapt to a more complex model of funds residing in end-user wallets and a transaction involving an inter-wallet transfer in addition to a system update. This would entail technical enhancements across the ecosystem to support the new funds model and integrate with a more complex payment flow while handling novel boundary conditions, e.g., aborting a transaction when one of two legs has been completed.

It is functionally feasible to audit the monetary supply, although its performance on a production-size supply needs further investigation. Mechanisms to fix discrepancies also require investigation. Our assessment of two variants of OpenCBDC 2PC for auditability indicates that mechanisms similar to either of them would be feasible functionally to audit the supply and detect a discrepancy, i.e., furnish a binary yes/no answer to the question "Did tampering occur?" The auditing process has a latency impact, and ways to mitigate it would need to be investigated for a production-size supply via optimizations and operational controls such as audit frequency, size of the monetary supply and core system capacity.

Further, a production system must not only be capable of detecting a discrepancy, but also of reversing it. This would require the system to maintain and compare historical audit logging and transaction information. The functional complexity and performance impact of reversal mechanisms are unknown, and we leave their investigation to future work.

Although linear scalability of throughput is achievable with systems such as OpenCBDC 2PC, the addition of enhancements like auditability would require careful design to preserve it to achieve throughput and latency goals. Our experiments with the baseline and two auditability extensions suggest that the baseline variant scales throughput linearly, while preserving latency in a range suitable for retail payments.[40] The two auditability variants, being more compute-intensive, do not scale as efficiently, the variant-PPA implementation in particular facing challenges. These early-stage observations can likely be improved with design optimizations. Further investigation is needed to assess feasibility for production UHS sizes. Our findings point to an important trade-off: the more information that is packed into the core state and the more intensive the process to update that state, the less scalable the resulting system is. Nevertheless, they speak to the efficiency of such systems that even the least performing configurations could achieve throughput in excess of 10,000 TPS.

Micro-partitioned systems such as OpenCBDC 2PC that support user custody of funds naturally support high levels of privacy from the CB and, optionally, from MSBs too. This system achieves high privacy from the CB by hiding from it the owner's identity, holdings and transactions. By choosing to be non-registered and with a type-U wallet, if permitted, a user can increase their privacy from MSBs as well. These ideas could be combined with the use of privacy-enhancing technologies to hide amounts even from the settlement system. This achieves a very high level of privacy that would exceed the privacy available in electronic payment systems today. We also observe that system enhancements must be carefully designed to avoid storing PII data, ensuring that the privacy provided by the system remains intact.

[40] Our experiments do not include wallet-to-wallet transfers, which when implemented could add to the latency experienced by users in some payments use cases.

Compliance in systems such as OpenCBDC 2PC relies on MSBs to maintain the necessary data. The CB and core system have no personal data about user identities, funds and transactions. Hence, rule-based and principle-based compliance are the sole responsibility of MSBs. They are feasible for registered users that operate type-M and type-K wallets, assuming that MSBs maintain the necessary data about those wallets and their owners, such as full transactions and history. This type of compliance is unfeasible for non-registered users, especially those with type-U wallets, since MSBs would not have data about them. However, controls (e.g., transaction limits) can be implemented in the system to prevent illicit usage. Policy-makers would need to balance compliance and privacy when determining which wallet and registration types to allow in a production system.

Guaranteeing the resilience of the core system state is feasible, but it requires engineering effort. Achieving resilience of the wallet state is also feasible. The core state (the UHS in OpenCBDC 2PC) constitutes the crown jewel of the system and hence must never be lost, even in the event of a total system outage. That is, the state of the UHS up to the last settled transaction must always be recoverable with zero data loss. This is theoretically feasible for systems like OpenCBDC 2PC, but it would require engineering effort to realize in a practical and efficient manner. We outline the enhancements needed—periodic persistent snapshots of the core state, persistent logs of subsequent updates and a recovery process to reconstitute the core state from the persistent store. Further, recovery must be timely, regardless of factors such as the time of last failure and the transaction volume. We leave to future work the investigation of how to achieve these goals, which would depend heavily on the state replication mechanisms and database technologies used.

The resilience of wallet contents would be the responsibility of wallet custodians such as MSBs and end users. We note that institutional backup and recovery mechanisms are already widely used by enterprises, and end users have access to a range of solutions to back up the contents of their devices, which could be employed for CBDC wallets.

Appendix A: Proposed transaction flow

We consider a payment from Alice to Bob in OpenCBDC 2PC. **Table A-1** shows the original transaction flow on the left and the proposed flow on the right. Our proposed flow achieves all four guarantees (G1 to G4) needed for non-repudiation (see Section 3.5), while the original flow achieves only G1.

Table A-1: OpenCBDC transaction flows to achieve non-repudiation

	Original flow	Proposed flow
a		Bob generates many (say, in the 1,000s) ephemeral public and private keys. Bob generates an equal number of certificates for an FR service by signing the latter's public key with his ephemeral private keys. Bob publishes the public keys to a PA service.[41] Bob also publishes public keys and corresponding certificates to his FR service.
b		Alice queries the PA service for Bob's public key. The PA serves a public key and, if ephemeral, deletes it from its store.
1	Alice creates a full-tx by signing witnesses for inputs being spent and generating outputs encumbered to Bob's public key.[42]	(Same as in Original flow)
2	Alice submits the full-tx to the system.	Alice sends the full-tx (with the ephemeral public key) to Bob or his FR service.
3a		(If the receiver is present) Bob verifies that each output is encumbered to a public key that corresponds to one of his private keys (G3). OR
3b		(If the receiver is not present) The FR computes a witness commitment from the input and Bob's public key and matches it to the witness commitment in the full-tx, hence verifying that output is encumbered to a public key that Bob published (G3).
4		Bob or the FR retrieves and stores outputs from the full-tx. Bob or the FR may store the full-tx for non-repudiation.

[41] Alternately, Bob could publish only a long-lived key to the peer alias (PA) service, which Alice could query. Alice would then compute an ephemeral key by choosing a random nonce and including that nonce in the full transaction she sends to Bob. There is no risk of another party intercepting the full transaction and stealing the outputs because, even with the nonce, they cannot constitute Bob's ephemeral private key without knowing his long-lived private key. Bob can, from the nonce, compute the ephemeral private key.

[42] This analysis can easily be extended to multiple public keys belonging to Bob. It could also be extended to multiple receivers in one transaction, who would need to communicate out of band to construct the transaction before submission. We exclude such transactions without loss of generality. However, an important case is when *two* receivers are involved, one of whom is Alice receiving change. That is a straightforward extension of the single receiver case already discussed, as Alice can append a transaction identification number (TxID) with her signature to the full transaction before sending it to Bob.

	Original flow	Proposed flow
5a		(If the receiver is present) Bob adds a TxID to the full-tx, one per output, signed by the private key that corresponds to his public key that the output is encumbered to (G1). OR
5b		(If the receiver is not present) TheFR adds a TxID to the full-tx, one per output, signed by its certificate that is signed using Bob's private key that corresponds to the public key the output is encumbered to.[43]
6		The receiver or FR submits the enhanced full-tx to a sentinel.
7	The system (sentinel) verifies the witness for each input. This proves inputs have been signed by a legitimate owner and that outputs have not been tampered with (G1).[44]	(Same as in Original flow)
8a		(If the receiver is present) The system (sentinel) verifies TxIDs are signed by a private key that corresponds to the public key to which outputs are encumbered, hence confirming that output pre-images are known to the entity to whom they are encumbered (G2; G4 is trivially true). OR
8b		(If the receiver is not present) The system (sentinel) verifies that the FR certificate is signed by the private key that corresponds to the public key to which outputs are encumbered. [45] This confirms that output pre-images are known to the FR (G2) that is authorized to act on behalf of the entity to whom outputs are encumbered (G4).[46]
9	System progresses transaction to core components for settlement.	(Same as in Original flow)

Note: FR is a funds receiver service, and PA is a peer alias service. Tx is a transaction, and TxID is a transaction identification number.

[43] Note that this does not require the funds receiver (FR) service to know Bob's private key. A simpler alternative could be for the FR to use a certificate signed by the central bank (CB). This would be more efficient but a weaker guarantee since, while it proves that the FR is authorized by the CB, it does not prove it is authorized to act on Bob's behalf.

[44] Each witness is a signature on the TxID computed over all inputs *and outputs*. Thus, if a man in the middle (MITM) attempts to tamper with outputs, it would change the TxID, invalidating witness signatures over the original TxID. Hence, validation of witnesses also confirms that an MITM did not tamper with outputs.

[45] Though these extra validation checks will impact performance, they are expected to preserve linear scalability, i.e., adding more validation capacity (at sentinels) for the same traffic would meet the throughput and latency targets.

[46] Note that nowhere in this construction is it necessary to communicate long-lived keys to the system. The system only knows Alice's and Bob's ephemeral keys and cannot create transaction graphs linking them over time.

Appendix B: Details of scalability test results

The configurations of tests listed below are as follows:

- **#shards, #coordinators, #sentinels:** Given as "x3" to reflect the number of instances of these components in each data centre (DC) and the total number across three DCs.
- **Pre-seeds:** Size of the hash set of unspent transaction outputs (UHS) at start of test. All tests use 2in:2out transactions, so the UHS size does not change for the duration of the test.
- **Block size:** Size of a block, i.e., a distributed transaction that coordinators send to shards, in #transactions. It is larger for the baseline and variant-values and smaller for the variant-PPA since transaction size is larger in a latter variant.
- **Throughput:** Transactions per second (TPS) settled and recorded in the UHS. This does not include the time for a wallet-to-wallet transfer, which is not implemented.
- **Latency:** Measured from the traffic generator back to itself, the average and 99-percentile (99p) latency figures in milliseconds. The 99p latency means that 99% of the transactions are completed in this time. For auditability variants, there is a latency spike during the audit logging process. Its duration and peak (in seconds) are also recorded. We consider only those tests whose average latency is under 1 second and the 99p is under 5 seconds, the audit spike time is under 10 seconds out of 60 seconds, and the peak spike is under 5 seconds.
- **Server size:** All server types are eight virtual central processing units (vCPUs), except load generators at 2 vCPUs.
- **SA (sentinel attestations):** The #attestations that each transaction must have to be accepted by the system and the #attestations that are validated by sentinels. If SA=0, it means attestations are disabled.
- **Audit interval:** In seconds, the period of audit logging process in the two auditability variants. The default is 60 seconds, so the total money in circulation is aggregated every minute.

Comparison of three architecture variants

Baseline

Baseline tests use 50 million pre-seeds and a block size of 100,000 (**Table B-1**).

Table B-1: Average throughput and latency (baseline, sentinel attestations=0)

Test ID	Shards	Coords	Sentinels	Avg. TPS	Avg. (99p) latency (ms)
4b6c0a8453e4	1x3	2x3	4x3	88,576	466 (661)
13cd512f643b	2x3	4x3	8x3	116,426	477 (689)
f98ad6ae6bdc	4x3	8x3	16x3	154,520	528 (839)
9e73b87d46b7	8x3	16x3	32x3	245,337	547 (844)

Source: Bank of Canada calculations

Variant-values

Variant-values tests use 10 million pre-seeds and a block size of 100,000. The audit interval is 60 seconds (**Table B-2**).

Table B-2: Average throughput and latency (variant-values, sentinel attestations=0)

Test ID	Shards	Coords	Sentinels	Avg. TPS	Avg. (99p) latency (ms)	Audit spike peak (s)	Audit spike time (s)
2cdec15b51a1	1x3	2x3	4x3	16,223	406 (2,116)	2.4	6
3d0990d2e9e7	2x3	4x3	8x3	30,444	526 (3,162)	3.3	8
e11d933ddda5	4x3	8x3	16x3	58,344	569 (3,445)	4.5	10
81c389e1d589	8x3	16x3	32x3	104,511	586 (3,762)	4.5	10

Source: Bank of Canada calculations

Variant-PPA

Variant-PPA tests use 0.5 million pre-seeds and a block size of 4,000.

In the implementation of the default OpenCBDC 2PC variant-PPA, sentinels perform only Pederson commitment (PC) checks on incoming transactions, deferring range proof (RP) checks to the audit logging process in shards. That poses the risk that a transaction could be declared settled, but its RP is later found to be invalid. To prevent this, we use a modified implementation in which sentinels perform both the RP and PC checks at the time of transaction validation so that RP checks are completed *before* a transaction is settled. This increases the burden on sentinels (and hence more of them are needed, as in **Table B-3**, compared with the other variants). Since RP checks are computationally expensive, one idea is to offload them to specialized hardware alongside sentinels to improve performance in a production system (not tested).

Table B-3: Average throughput and latency (variant-PPA, sentinel attestations=0)

Test ID	Shards	Coords	Sentinels	Avg. TPS	Avg. (99p) latency (ms)	Audit spike peak (s)	Audit spike time (s)
b0a36baa7f80	1x3	2x3	8x3	10,318	450 (1,810)	2.0	8
4ae962c4f597	2x3	4x3	16x3	18,657	517 (1,186)	1.0	6
267cead07d0d	4x3	8x3	32x3	24,669	537 (1,472)	1.2	7
49d19be32780	8x3	16x3	64x3	37,006	506 (1,211)	1.0	12

Source: Bank of Canada calculations

Sentinel attestations feature

Sentinel attestation (SA) tests are executed for only two variants, baseline and variant-values. Variant-PPA is not tested since, even with the SA feature disabled, it scales quite slowly, indicating that optimizations would be needed to execute it with the feature enabled.

Baseline

Baseline tests with SA=2 use 10 million pre-seeds and a block size of 100,000 (**Table B-4**).

Table B-4: Average throughput and latency (baseline, sentinel attestations=2)

Test ID	Shards	Coords	Sentinels	Avg. TPS	Avg. (99p) latency (ms)
4434d00fa46e	1x3	2x3	4x3	54,689	443 (636)
1c9773c7f3c2	2x3	4x3	8x3	64,995	674 (2,009)
1f52045544a4	4x3	8x3	16x3	83,166	565 (1,464)
49a7301a5d3e	8x3	16x3	32x3	94,319	623 (2,369)

Source: Bank of Canada calculations

Variant-values

Baseline tests with SA=2 use 10 million pre-seeds and a block size of 100,000 (**Table B-5**).

Table B-5: Average throughput and latency (variant-values, sentinel attestations=2)

Test ID	Shards	Coords	Sentinels	Avg. TPS	Avg. (99p) latency (ms)	Audit spike peak (s)	Audit spike time (s)
2a1e7db314e8	1x3	2x3	4x3	17,483	658 (3,600)	4.0	8
160239f8f6f0	2x3	4x3	8x3	29,463	630 (3,299)	4.5	10
d98c500fda0e	4x3	8x3	16x3	49,310	740 (4,308)	4.0	15
3f0aca3b169a	8x3	16x3	32x3	62,069	599 (2,364)	3.5	7

Source: Bank of Canada calculations

Limitations

- **Transient degradation:** In variant-PPA tests spanning 5 or 10 minutes, we notice an unexplained degraded performance in the first 200 seconds or so, which stabilizes thereafter. We are unable to ascertain the root cause and surmise there may be initial start-up issues to be resolved. We also notice that memory leaks would accumulate steadily in all variants. Although the tests don't run long enough to trigger out-of-memory conditions, they indicate the presence of potential issues of unknown impact.
- **Optimal inter-component associations:** A sentinel communicates with a coordinator to submit transactions for settlement. If their numbers are identical, e.g., eight sentinels and eight coordinators, they get allocated 1-to-1. If not, e.g., if there are eight sentinels and six coordinators, it is unlikely that the traffic from sentinels is uniformly distributed to the coordinators in the current codebase. Another factor is their deployment across three DCs. (For example, is a sentinel in DC-1 always guaranteed to find a coordinator in the same DC?) Some of the tests have unequal numbers of instances, and we are unable to ascertain if our configurations are sub-optimal.

- **Optimal relative ratios of components:** We establish the relative ratios of shards, coordinators and sentinels for one shard. Then, as the shard count is doubled to 2, 4 and 8, the other components are doubled as well. A more in-depth approach would ascertain optimal ratios at each of those shard numbers separately.
- **Third-party changes:** We observe early in our testing that throughput scales at a rate close to 2.0, i.e., adding twice the capacity would also increase throughput by two times. However, in later testing, after the compiler had been upgraded along with some libraries, the rate of scaling dropped to lower than two times, even though these upgrades were unrelated to the core logic. We are unable to find the cause of this degradation. It shows how seemingly innocuous changes in large codebases that include third-party and open-source contributions can have unforeseen impacts. This highlights the need for continual monitoring and measurement of the system as it evolves over its life cycle.

References

Arora, R. and R. Darbha. 2020. "Privacy in CBDC Technology." Bank of Canada Staff Analytical Note No. 2020-9. DOI: https://doi.org/10.34989/san-2020-9.

Bank of Canada. 2023. "Currency." *Annual Report 2023*. https://www.bankofcanada.ca/publications/annual-reports-quarterly-financial-reports/annual-report-2023/currency/.

Baudet, M., G. Danezis and A. Sonnino. 2020. "FastPay: High-Performance Byzantine Fault Tolerant Settlement." *ACM Conference on Advances in Financial Technologies (AFT):* 163–177. ACM.

Buldas, A., M. Saarepera, J. Steiner, L. Ilves, R. Olt and T. Meidla. 2021. "A Formal Model of Money Schemes and Their Implications for Central Bank Digital Currencies." Joint research paper by Eesti Pank and Guardtime. https://haldus.eestipank.ee/sites/default/files/2021-12/EP-A_Formal_Model_of_Money_2021_eng.pdf.

Buterin, V. 2016. "Thoughts on UTXO." *Medium*. https://medium.com/@ConsenSys/thoughts-on-utxo-by-vitalik-buterin-2bb782c67e53.

Darbha, R. 2022. "Archetypes for a Retail CBDC." Bank of Canada Staff Analytical Note No. 2022-14. DOI: https://doi.org/10.34989/san-2022-14.

George, N., T. Dryja and N. Narula 2023. "A Framework for Programmability in Digital Currency." Massachusetts Institute of Technology Digital Currency Initiative. https://dci.mit.edu/s/Public-Copy-A-Framework-for-Programmability-in-Digital-Currency-August-1st-2023-MIT-DCI.pdf.

Google Inc. n.d. "Level DB." https://github.com/google/leveldb.

Gravitis, A., N. Goh and D. Toliver. 2019. "TODA Primer." Report from TODAQ Holdings Inc. https://engineering.todaq.net/TODA_Tech_Primer_v1.1.pdf

Guerraoui, R., P. Kuznetsov, M. Monti, M. Pavlovic and D.-A. Seredinschi. 2019. "The Consensus Number of a Cryptocurrency." *ACM Symposium on Principles of Distributed Computing (PODC)*: 307–316. ACM.

Lovejoy, J., M. Virza, C. Fields, K. Karwaski, A. Brownworth and N. Narula. 2023. "Hamilton: A High-Performance Transaction Processor for Central Bank Digital Currencies." *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI '23)*: 901–915. Boston, MA: USENIX Association.

Massachusetts Institute of Technology Digital Currency Initiative. 2022. "OpenCBDC." https://dci.mit.edu/opencbdc.

Stuewe, S. "Payment Routing for CBDCs." Personal note shared May 30, 2024.

Stuewe, S., M. Virza, M. Maurer, J. Lovejoy, R. Bohme and N. Narula. 2024. "Beware the Weak Sentinel: Making OpenCBDC Auditable Without Compromising Privacy." Massachusetts Institute of Technology Digital Currency Initiative working paper. https://dci.mit.edu/beware-the-weak-sentinel-making-opencbdc-auditable-with-out-compromising-privacy.