Solving Data Availability Limitations in Client-Side Validation with UTxO Binding

Yunwen Liu[1], Bo Wang[2], and Ren Zhang*[1,3]

[1] Cryptape
yunwen@cryptape.com, ren@cryptape.com
[2] UTxO Stack
cipher@nervos.org
[3] Nervos

Abstract. Issuing tokens on Bitcoin remains a highly sought-after goal, driven by its market dominance and robust security. However, Bitcoin's limited on-chain storage and functionality pose significant challenges. Among the various approaches to token issuance on Bitcoin, client-side validation (CSV) has emerged as a prominent solution. CSV delegates data storage and functionalities beyond Bitcoin's native capabilities to off-chain clients, while leveraging the blockchain to validate tokens and prevent double-spending. Nevertheless, these protocols require participants to maintain token ownership and transactional data, rendering them vulnerable to data loss and malicious data withholding. In this paper, we propose UTxO binding, a novel framework that achieves both robust data availability and enhanced functionality compared to existing CSV designs. This approach securely binds a Bitcoin UTxO, which prevents double-spending, to a UTxO on an auxiliary blockchain, providing data storage and programmability. We formally prove its security and implement our design using Nervos CKB as the auxiliary blockchain.

1 Introduction

Bitcoin's dominant market share among cryptocurrencies—61% as of March 2025 [6]—and robust security, forged through a decade of battle-tested proof-of-work consensus [10, 26, 17, 31], have made it an attractive platform for token issuance since 2011 [11]. Early efforts, including Colored Coins [24], Counterparty [8], and Omni Layer [23], aimed to leverage Bitcoin's security by embedding token ownership information directly on the blockchain. However, Bitcoin's limited on-chain storage space and restricted programmability hinder its suitability for tokens carrying data and requiring complex transactional logic. These constraints led to the emergence of alternative platforms like Ethereum [4], co-founded by Vitalik Buterin, a key contributor to Colored Coins, and specifically designed to address programmability limitations.

Recognizing the limitations inherent to Bitcoin in on-chain token issuance, recent years have witnessed a surge in interest in *client-side validation (CSV)*

* Corresponding author.

designs. Inspired by Todd [28] and exemplified by the RGB project [9], these designs store token ownership and transaction data off-chain while embedding only short commitments for issuance and validation on the Bitcoin blockchain. Compared to on-chain tokens, CSV designs offer reduced on-chain space consumption and freedom from Bitcoin's programmability constraints. However, they also introduce two new challenges. Firstly, the *data availability issue* arises from storing ownership and transaction data off-chain. Reliable external storage mechanisms are crucial to ensure this information remains accessible to the users. Secondly, the *client coherence issue* stems from the removal of on-chain enforcement of transactional logic. All users must adhere to the same set of rules to maintain consensus, which can be challenging to achieve and enforce.

This paper introduces *UTxO binding*, a novel technique to address the data availability and client coherence issues inherent in CSV designs. Our approach leverages an *auxiliary UTxO-based blockchain*, auxChain for short, to store token ownership and transaction data and to enforce transactional logic after token issuance on Bitcoin. In a UTxO-based blockchain, as in Bitcoin, a user's assets are managed as a set of *unspent transaction outputs*—hence the name UTxO, rather than a unified account. Each transaction output specifies an amount of cryptocurrency and the conditions under which that amount can be spent. By delegating data availability and client coherence to the auxChain, our approach achieves enhanced programmability compared to Bitcoin and stronger robustness than prior CSV designs. The core challenge is establishing a secure and verifiable *binding* mechanism: a one-to-one correspondence between a Bitcoin UTxO, representing token ownership and validity, and a corresponding *shadow UTxO* on the auxChain, which stores the associated data. We address this challenge by proposing a generic framework and demonstrating its feasibility through an instantiation with Nervos CKB [22]—CKB for short—as the auxiliary blockchain. Our work and contributions are as follows:

Generic UTxO Binding Designs. We address the core challenge through the following workflow. The Bitcoin token-binding UTxO is generated first, whose generation transaction commits to its future shadow UTxO. The committed shadow UTxO can only be generated on auxChain after the Bitcoin transaction is confirmed. This shadow UTxO must embed information that uniquely identifies its Bitcoin counterpart. We develop this workflow into two generic variants of UTxO binding: *basic UTxO binding*, implementable in any UTxO-based blockchain that supports arbitrary data embedding, which requires users to verify transactions on both chains, and *autonomous UTxO binding*, which leverages an on-chain Bitcoin light client on the auxChain to automate verification. We formally prove that in our designs, an accepted binding relation is *exclusive*, meaning it must be a bijection, *unforgeable*, meaning it cannot be disrupted or invalidated by an attacker, and *verifiable* either by any party with access to both blockchains, for basic UTxO binding, or by the auxChain's smart contract, for autonomous UTxO binding. Consequently, token ownership and transactional logic are reliably stored and enforced. Our designs thus resolve the critical challenges of data availability and client coherence in CSV.

Implementation on Nervos CKB. A practical implementation of autonomous UTxO binding is deployed on CKB, a UTxO-based blockchain with Turing-complete smart contracts. Key innovations include structuring transactions to prioritize token-carrying inputs/outputs and integrating CKB's native Bitcoin light client for on-chain verification. Our implementation also strengthens the protection of benign users' tokens during Bitcoin reorganization. Performance metrics highlight efficiency: token transfers incur minimal computational overhead and collateral costs under 0.75 USD per UTxO. By March 2025, the protocol supported 581 tokens, with one token, Seal, achieving over 42 000 holders.

Comparative Analysis. We compare UTxO binding against nine other influential Bitcoin layer-2 protocols based on four key metrics: data storage, programmability, privacy, and deployment status. The comparison highlights UTxO binding's unique combination of security, data availability, and programmability, distinguishing it from existing solutions.

2 Token Issuance on Bitcoin

Issuing tokens—fungible or non-fungible—on existing blockchains has become a common practice, rather than creating a new blockchain for each token. This approach offers several advantages to issuers, including leveraging the established security, readily available tools, and large user bases of existing platforms. By building upon existing infrastructure, issuers can minimize development and marketing costs, enabling them to concentrate on their core business objectives.

Bitcoin, renowned as the first, most influential, and arguably most secure blockchain, has consistently attracted token issuance projects. While numerous endeavors have aimed to achieve this on the Bitcoin blockchain [13], none have achieved widespread success. Many of these Bitcoin-based token issuance projects offer limited functionalities, while others remain in development with uncertain prospects for a near-term launch. As a result, CoinMarketCap [7], a prominent platform for tracking cryptocurrencies, lists only 116 tokens within the Bitcoin ecosystem as of March 2025. In stark contrast, the Ethereum, Solana, and BNB Smart Chain ecosystems involve 4201, 2365, and 4284 tokens, respectively. Next, we briefly overview Bitcoin's transaction structure and examine these Bitcoin-based token issuance protocols, emphasizing both their strengths and weaknesses.

2.1 Bitcoin's Transaction Structure

Bitcoin transactions record value transfers between users. Each transaction comprises *inputs* and *outputs*. Inputs reference previous outputs, which represent the sender's received funds, while new outputs specify the destinations of the funds. A previous output is identified by its *transaction ID* (txid)—the SHA256 hash of the transaction data that generated the output, and an index, the index number of the output within that transaction, starting with zero. A new output consists of a value—its worth in satoshis, the smallest unit of Bitcoin, and a *script* defining

the spending conditions. The most common script is "`P2PKH addr`", where `P2PKH` signifies "pay to public key hash" and `addr` specifies the receiver's *address*, i.e., the hash of the receiver's public key. When the output is spent, the receiver's full public key P and their signature on the complete transaction, generated with the corresponding private key, must be provided as *parameters*. The set of unspent transaction outputs (UTxOs) defines the ownership of all bitcoins.

Beyond simple value transfers, Bitcoin's scripting system enables arbitrary data embedding. Unless specified otherwise, we define "data embedding" and "embedded in the data field" as information provided during UTxO creation, and "parameters" as information provided during UTxO spending. The opcode `OP_RETURN` marks an output as unspendable, allowing up to 80 bytes of arbitrary data within the script after it.

Bitcoin's Taproot upgrade [29] allows a selection of spending conditions, where satisfying any one authorizes UTxO spending. This upgrade introduces a new address format, the *Taproot address*, encoded from a *tweaked public key* $\mathsf{P}^{\mathsf{t}} = \mathsf{P} + \mathrm{H}(\mathsf{P} \,||\, \mathsf{root}) \cdot \mathrm{G}$. Here, G is a public generator point of an elliptic curve, P is a public key of the Schnorr signature with corresponding private key p, satisfying $\mathsf{P} = \mathsf{p}\mathrm{G}$, H is a hash function, and root is a Merkle root. The Merkle tree associated with root presents each alternative spending script as a leaf. P^{t}'s corresponding private key is thus $\mathsf{p}^{\mathsf{t}} = \mathsf{p} + \mathrm{H}(\mathsf{P} \,||\, \mathsf{root})$. Revealing and fulfilling any single leaf authorizes the transaction. During UTxO spending, the owner provides, in addition to P^{t}, either a corresponding signature signed with p^{t}, or P and a Merkle authentication path with respect to a leaf, via a *witness* field attached to the transaction. All alternative conditions remain secret.

Designing a Token Issuance Protocol. A token issuance protocol design comprises two distinct components: the *token carrier*, which defines how to encode the type, value, and ownership of tokens, and the *token contract*, which defines the transaction logic, including initial issuance, transaction rules, and, optionally, decentralized applications that utilize the token. These components roughly correspond to Bitcoin's UTxO and *Bitcoin Script*—Bitcoin's native programming language used to specify spending conditions, respectively. The key challenges in designing the token carrier and token contract are determining where to store the extra data and how to enforce the logic, respectively.

2.2 Non-CSV Designs

On-Chain Tokens. These protocols store all relevant data on the Bitcoin blockchain. They differ in how they repurpose existing Bitcoin fields to implement the token carrier, given Bitcoin's lack of a native mechanism for on-chain data storage. For example, Colored Coins [24] embeds the token carrier in the `nSequence` field of UTxOs, which accommodates 4 bytes. Counterparty [8] and Omni Layer [23] store token data after the `OP_RETURN` field, which allows up to 80 bytes.

Ordinals. Ordinals leverages Taproot by chunking data within the Merkle tree leaves corresponding to a Taproot address, effectively expanding token carrier capacity to 4 MB. Although these data are verifiable using the Taproot address,

most of them reside off the Bitcoin blockchain. Consequently, Ordinals is not an on-chain token design.

Limitations. On-chain token protocols implement token carriers within Bitcoin's UTxO and transaction structures. These structures are intentionally space-limited to prevent excessive bandwidth and storage costs on Bitcoin network participants. Conversely, Ordinals leverages external storage, introducing data availability concerns.

Colored Coins and Ordinals enforce their token contracts through Bitcoin's consensus, which offers limited programmability. Mitigating this limitation hinges on the development of advanced smart contract frameworks like BitVM [3, 12], which are currently in their conceptual stages. Counterparty and Omnilayer enforce token contracts on the client side, introducing client coherence concerns. These concerns are further discussed in Sec. 2.3.

2.3 Client-Side Validation

Recognizing the limitations of early attempts, recent years have seen a surge in CSV protocols. Inspired by Todd [28], these designs offload token carrier to the client side, while embedding concise commitments on the Bitcoin blockchain for transaction ordering and verification. Consequently, users must store their transaction history locally. As on-chain enforcement of the token contract is no longer feasible, enforcement shifts to the client side, removing constraints imposed by Bitcoin's programmability. RGB [9], the earliest and most influential CSV protocol, was initially proposed in 2018 and has undergone continuous revisions. Following the approach of Ordinals, Taproot Assets [21] leverages the Taproot upgrade for its token carrier. Intmax2 [25] and Shielded CSV [15] further enhance these designs by incorporating zero-knowledge proofs and advanced cryptographic signature schemes to compress on-chain data and enhance users' privacy. Next, we detail four challenges inherent to CSV protocols.

Data Availability. CSV protocols require users to maintain their full transaction history; data loss results in token loss. This places a heavier burden on ordinary users than private key management. While private keys typically require a one-time backup as mnemonic phrases, CSV users must back up their latest transaction history each time they send or receive tokens.

This situation could be addressed via a globally accessible backup mechanism. However, maintaining such a backup is challenging, as evidenced by the various attempts within the Ethereum ecosystem to address their own data availability issues. Delegating backup responsibilities to third parties inherently grants them the power to censor information and exert control over the system. For instance, following a security breach in June 2024, the Ethereum project Linea temporarily suspended operations to prevent further financial loss by halting its *sequencer*—a trusted entity responsible for collecting Linea's transaction data [20]. This operation, though benign, revealed that such data centers constitute a single point of failure for the system. Conversely, solutions that eliminate this single point of failure typically involve a committee of nodes [27, 1, 18] and additional, often expensive, mechanisms to guarantee their honesty [2].

Peer Discovery. Transmitting transaction history data to the recipient poses a further challenge. Protocol designers face undesirable choices. Directly sending data to the recipient necessitates disclosing their IP address, which raises privacy concerns and increases vulnerability to DoS attacks. Alternatively, broadcasting data over a dedicated P2P network consumes significant resources, as the history expands linearly with time.

Client Coherence. Existing CSV protocols require all users to execute identical validation rules, typically achieved by running the same client version. This presents challenges for rule updates and impacts client diversity, which is generally considered crucial for blockchain protocol security [19]. Furthermore, due to the difficulty of maintaining coherent validation rules, existing protocols employ simple rules to minimize vulnerabilities. As of March 2025, all four aforementioned CSV designs are restricted to simple token transfers without programmability support. Specifically, Taproot Assets, Intmax2, and Shielded CSV do not include programmability in their roadmaps. This limitation is likely not coincidental. Taproot Assets, aiming for native support by most Bitcoin clients, is constrained by Bitcoin's programmability. Intmax2 and Shielded CSV, already employing complex cryptographic tools, could experience further performance degradation with added programmability. Only RGB's architecture envisions expressive programmability via AluVM. However, like BitVM, AluVM remains in early development with no imminent release.

State Integrity. A token maintains *state integrity* when its complete ownership is committed to the Bitcoin blockchain. On-chain tokens inherently satisfy state integrity, as all ownership information resides on-chain. CSV users, however, lack access to the full token ledger. Therefore, a proof of state integrity is essential. Without such a proof, users cannot be certain that malicious actors are prevented from generating tokens through undefined methods or presenting multiple valid versions of the transaction history. Existing designs lack this proof.

3 Preliminaries

We now present the preliminaries necessary for understanding UTxO binding. We begin with a description of the RGB protocol, the foundation of our design. Originally proposed by Orlovsky and Zucco in 2016, the RGB protocol is the earliest CSV design and, as of March 2025, remains highly influential due to its relative simplicity and flexibility. Following this, we provide an overview of Nervos CKB's Cell model, the platform underlying our implementation.

3.1 RGB Protocol

In RGB, each token unit must be bound to a Bitcoin UTxO, known as the *token-binding UTxO*. A token-binding UTxO is indistinguishable from an ordinary Bitcoin UTxO until it is spent; it contains no token-specific information. The token is considered spent when its binding UTxO is spent.

Issuance. Issuing an RGB token involves designating an existing Bitcoin UTxO, owned by the issuer, as the *genesis UTxO*. This genesis UTxO—a special token-binding UTxO—represents the token's initial supply. The token's complete issuance policy and its *schema*, i.e., the token contract, are defined off-chain, e.g., on its website, and agreed upon by all users of the token.

Transaction. Transferring a token is more complex. Let Alice be the sender and Bob the receiver. Assume Alice has demonstrated to Bob that $\sigma_{\mathcal{A}}$, Alice's UTxO, is bound to a sufficient quantity of tokens. We will later explain how Alice accomplishes this. Alice must also demonstrate two further points to Bob: First, Alice has not previously transferred the tokens associated with $\sigma_{\mathcal{A}}$. This is evident as $\sigma_{\mathcal{A}}$ remains unspent. Second, Bob is the legitimate recipient. Alice demonstrates this through the following interactive *token transfer protocol*. First, Bob (1) selects $\sigma_{\mathcal{B}} = (\mathsf{txid}_{\mathcal{B}}, \mathsf{index}_{\mathcal{B}})$, one of his existing Bitcoin UTxOs, as the receiving token-binding UTxO, (2) computes a *seal* $S_{\mathcal{B}} = \mathrm{H}(\mathsf{txid}_{\mathcal{B}}, \mathsf{index}_{\mathcal{B}}, salt_{\mathcal{B}})$ where H is a hash function and $salt_{\mathcal{B}}$ is a random number, and (3) sends $S_{\mathcal{B}}$ to Alice. The random number prevents Alice from learning $\sigma_{\mathcal{B}}$ from $S_{\mathcal{B}}$ by enumerating all Bitcoin UTxOs. Upon receiving $S_{\mathcal{B}}$, Alice (1) generates an RGB transaction $\mathsf{Tx}_{\mathcal{A}\to S_{\mathcal{B}}}$, a cryptographic proof transferring tokens to the UTxO sealed in $S_{\mathcal{B}}$, (2) computes $C_{\mathsf{Tx}} = \mathrm{H}(\mathsf{Tx}_{\mathcal{A}\to S_{\mathcal{B}}})$, a commitment to the RGB transaction, (3) spends $\sigma_{\mathcal{A}}$ in a Bitcoin transaction whose first output (index 0) embeds C_{Tx} after `OP_RETURN` , and (4) sends $\mathsf{Tx}_{\mathcal{A}\to S_{\mathcal{B}}}$ to Bob. Bob verifies the correct construction and embedding of C_{Tx} in the Bitcoin transaction spending $\sigma_{\mathcal{A}}$. Successful verification convinces Bob that he has received the tokens. Note that only Bob knows that the tokens are now bound to $\sigma_{\mathcal{B}}$. Alice retains her bitcoins associated with $\sigma_{\mathcal{A}}$, as they are transferred to other outputs of the Bitcoin transaction. She can no longer transfer the tokens to anyone else because $\sigma_{\mathcal{A}}$ is spent.

To demonstrate to Bob that $\sigma_{\mathcal{A}}$ is a token-binding UTxO with the required balance, Alice provides Bob with (1) the sequence of token-binding UTxOs from the token's genesis UTxO to the UTxO(s) that sent tokens to $\sigma_{\mathcal{A}}$, (2) the corresponding salts used to compute their seals, and (3) the complete history of RGB transactions among these UTxOs. This allows Bob to verify the full transaction history related to $\sigma_{\mathcal{A}}$. If any seal, transaction, or commitment in its history is invalid, $\sigma_{\mathcal{A}}$ is invalid. When Bob transfers the tokens to Carol, he adds $\sigma_{\mathcal{B}} = (\mathsf{txid}_{\mathcal{B}}, \mathsf{index}_{\mathcal{B}})$ to the list of token-binding UTxOs, $salt_{\mathcal{B}}$ to the salt list, and $\mathsf{Tx}_{\mathcal{A}\to S_{\mathcal{B}}}$ to the transaction history, and repeats the token transfer protocol.

Limitations. RGB suffers from the previously mentioned data availability, peer discovery, and client coherence issues. Regarding state integrity, while the default setting is generally considered correct, no formal proof or argument is provided to demonstrate why all acceptable schemas and future updates maintain it.

Some advantages claimed by RGB proponents are overstated or unrealized. The *recipient privacy*—the receiver's token-binding UTxO remains unknown to others—is temporary, as it is eventually revealed to subsequent recipients. The *programmability* of AluVM remains conceptual, with no development progress in years. While RGB reduces on-chain storage costs by using constant-size com-

mitments for RGB transactions, the limited number of inputs and outputs in most Bitcoin transactions suggests that the *throughput gain* is modest.

Furthermore, RGB incurs greater costs than an ordinary blockchain transaction in two ways. First, the protocol's interactive nature—requiring the receiver to compute and send the seal—increases communication overhead and limits its applicability. Second, the history of transactions associated with a token grows over time and can eventually comprise a substantial portion of RGB's full ledger.

3.2 Nervos CKB Cells and Scripts

We chose CKB for our implementation for two reasons. First, its UTxO-based model allows a direct correspondence with Bitcoin UTxOs. Second, its Turing-complete virtual machine supports complex cryptographic operations. Specifically, an off-the-shelf Bitcoin light client is available on CKB, enabling on-chain verification of Bitcoin transaction embedding. While not strictly necessary for UTxO binding, this simplifies our threat model and implementation.

Central to CKB's architecture is the Cell model, an enhanced UTxO framework designed to address the limitations of Bitcoin's UTxO model. We provide a brief overview here and refer readers to the official documentation for a detailed description [30]. Each UTxO, termed a *cell*, is identified by its transaction ID and output index, as in Bitcoin. Each cell comprises four fields: capacity, data, lock script, and type script. *Capacity* specifies the cell's allocated storage space, while *data* stores arbitrary key-value pairs up to that capacity. Cell behavior is governed by programmable scripts, categorized as *lock scripts* and *type scripts*. The lock script defines the spending conditions, which must be satisfied when the cell is used as input, thus enforcing ownership, similar to Bitcoin's script. The type script offers greater flexibility, defining conditions that must be satisfied when the cell is used as input and/or output. For example, a type script might specify that "the transaction generating this cell is valid only if one of its input cells has a capacity exceeding 1 MB." In contrast to Bitcoin, where scripts lack access to transaction outputs, CKB scripts possess full transaction visibility. Consequently, *covenants*, which are constraints on transaction outputs [14, 16], can be enforced within either script.

CKB scripts employ code abstraction: each script comprises (1) a `code_hash` field, referencing the actual smart contract code stored in a previous cell's data field, and (2) an `args` field providing transaction-specific parameters. The invocation of code stored within a previous cell's data field does not destroy that cell. For instance, a lock script implementing the secp256k1 signature scheme references the algorithm's code binaries via its `code_hash`, while the owner's public key is included in `args` when the cell is spent. To unlock the lock script and spend the UTxO, the owner must provide a valid signature generated using the corresponding private key. This signature is appended to the block containing the transaction in its witness field, as in Bitcoin. Unlike Bitcoin, where parameters are only provided to transaction inputs, both lock and type scripts in CKB also accept parameters when the cell acts as an output. As output lock scripts are not executed, their parameters are accessible when the cell is spent.

CKB natively supports diverse token types, each uniquely identified by a token ID. Typically, both the token type and its associated contract are defined as smart contract code, which is then referenced by all cells holding that token as the type script. Our implementation adheres to this convention: the type script identifies the token type and verifies that it is issued through UTxO binding.

4 UTxO Binding

Our key idea is to employ an auxiliary blockchain, called auxChain, to store the token carrier and enforce the token contract. A token transfer is valid only if confirmed on both Bitcoin and auxChain. Leveraging a blockchain network, which exhibits greater robustness than a collection of clients managing a token, effectively resolves all four challenges inherent in CSV designs. Data availability is ensured, as it is persistently stored by the auxChain's full nodes. The token contract is enforced by the auxChain's consensus, guaranteeing client coherence. The peer discovery dilemma is resolved: only the most recent transaction, instead of the entire transaction history, requires broadcasting within the auxChain's P2P network, and the recipient's IP address remains concealed. Finally, the public transaction ledger eliminates the need for state integrity proof.

The key challenge is maintaining a one-to-one correspondence between Bitcoin's and auxChain's token-binding UTxOs, preventing the forgery or invalidation of such a UTxO on either chain. This section presents *UTxO binding*, a generic design addressing this challenge. We begin with the threat model and desired properties, then present two variants of our design.

Remark. Adopting an auxChain differs fundamentally from issuing tokens on a *sidechain*, a blockchain designed for bidirectional bitcoin transfers. CSV designs aim to issue tokens directly on the Bitcoin blockchain and commit every transaction to it. These objectives are analogous to those of *rollups*, an approach infeasible on Bitcoin. A sidechain-based token, however, is issued only on the sidechain and secured by it, failing to achieve these core objectives.

4.1 Threat Model

Given that Bitcoin is selected as a token issuance platform for its security, we assume that every Bitcoin transaction becomes irreversible after a sufficient number of blocks are mined following the block containing the transaction. This assumption allows us to broadcast the Bitcoin transaction first, and broadcast the corresponding auxChain transaction only after the former is confirmed. Like other CSV protocols, our protocol relies on the principle that each Bitcoin UTxO can be spent only once by its owner. Furthermore, we do not utilize Bitcoin to enforce token-specific transactional logic, such as verifying transaction confirmation on the auxChain.

The auxChain natively supports multiple assets, identified with unique *token IDs*, and can embed all token-related data as key-value pairs within the corresponding output. Its smart contract is sufficiently expressive to enforce the

token contract. We do not rely on the irreversibility of auxChain blocks. Even if an adversary conducts a consensus attack and forks the auxChain, token ownership remains unaffected, because the adversary cannot forge a valid shadow transaction. Several UTxO-based blockchains, including Cardano [5], Ergo, and Nervos CKB, meet these requirements.

The *issuer* honestly constructs token issuance transactions on both chains. The *sender* can initiate transactions on both chains and operates consistently across both blockchains to prevent token loss. Sending conflicting transactions in UTxO binding is equivalent to sending assets to unspendable addresses. The *receiver* securely provides receiving addresses on both chains to the sender. Unlike RGB, UTxO binding can be non-interactive: after providing the addresses, the receiver does not need to be online to receive the tokens. Moreover, UTxO binding does not require the receiver to possess an existing Bitcoin UTxO.

If the auxChain offers an on-chain Bitcoin light client, only auxChain access is required to verify the token transaction. Otherwise, the receiver must access both chains to verify the transaction.

The *attacker* aims to break the binding between a Bitcoin transaction and its corresponding shadow transaction, double-spend tokens, or invalidate token ownership. The attacker can observe all public blockchain data and transactions propagated across the P2P networks. Upon receiving an honest transaction, the attacker may construct and confirm transactions on both blockchains before the honest transaction is confirmed. However, the attacker cannot break the ownership locks on either blockchain.

We stress that this threat model introduces no additional assumptions beyond prior CSV designs. The auxChain merely provides an extra layer of protection for data availability, client coherence, and state integrity, facilitating the receiver's verification. Crucially, Bitcoin and auxChain do not need to be synchronized: token transfer remains valid as long as the shadow transaction is eventually confirmed, even after the Bitcoin token-binding UTXO is spent, if the receiver is convinced of the transaction's validity.

4.2 Desired Properties

As argued, with UTxO binding, establishing a secure one-to-one correspondence between Bitcoin and shadow transactions naturally addresses data availability, peer discovery, client coherence, and state integrity issues. We now define the desired properties for this secure one-to-one correspondence. When the auxChain does not offer an on-chain Bitcoin light client, UTxO binding should satisfy the following properties:

Exclusive Binding. Each Bitcoin token-binding UTxO σ corresponds to exactly one valid shadow UTxO σ' in auxChain. Conversely, σ' corresponds exclusively to σ.

Unforgeability. An attacker cannot invalidate token-binding UTxOs or disrupt token transactions.

Public Verifiability. In the event of a token contract violation, such as double-spending or over-issuing, any party with access to both blockchains can identify the offending transactions.

Exclusive binding ensures that neither the sender nor an attacker can create multiple valid Bitcoin or shadow UTxOs to confuse the receiver. Unforgeability prevents attackers from double-spending or invalidating tokens. Public verifiability is stronger than the *local verifiability* of other CSV designs, where receivers can only verify the absence of over-issuing or double-spending concerning their own tokens. Both of our variants achieve these properties.

Exclusive binding alone is insufficient for security. Consider the following RGB variant. Instead of using the hash of the RGB transaction $\mathsf{Tx}_{\mathcal{A}\to S_{\mathcal{B}}}$ as the commitment C_{Tx}, we define $C_{\mathsf{Tx}} = \mathrm{H}(\mathsf{Tx}_{\mathcal{A}\to S_{\mathcal{B}}}||r)$, where r is a random number. We prescribe that the first auxChain UTxO embedding r becomes the receiver's shadow UTxO. The total ordering of auxChain ensures exclusive binding: each random number corresponds to only one earliest UTxO on each chain. However, this design fails to achieve unforgeability, as it enables a *frontrunning attack*. Upon observing the shadow transaction containing r, an attacker can submit a competing transaction that also embeds r and confirm it before the honest transaction. This effectively locks the receiver's tokens on the auxChain.

When auxChain provides an on-chain Bitcoin light client, we can replace public verifiability with a stronger property:

Contractual Integrity. If a shadow UTxO σ' is valid, then the corresponding Bitcoin token-binding UTxO σ adheres to the token contract.

This implies that only auxChain access is required for transaction verification. Our second variant, *Autonomous UTxO Binding*, achieves this property.

4.3 Basic UTxO Binding

When the auxChain does not offer an on-chain Bitcoin light client, the receiver provides addresses on both blockchains to the sender and requires access to both blockchains for token transaction verification.

Issuance. Similar to RGB, the token issuer designates an existing Bitcoin UTxO as the genesis UTxO. No information embedding is necessary within this genesis UTxO. Subsequently, the issuer creates a transaction on auxChain with a *shadow genesis UTxO*. This shadow genesis UTxO specifies: (1) the token contract, including its issuance policy and transactional logic within its smart contract, and (2) its token ID, along with the Bitcoin genesis UTxO's transaction ID and output index in its data field. This action is equivalent to issuing the token on the auxChain, a function natively supported, with the added information of the Bitcoin genesis UTxO. From this point forward, the token contract is enforced as a covenant by the auxChain's consensus. The issuer then announces both UTxOs to the community via a trustworthy channel.

Transaction. Let Alice be the sender and Bob the receiver. Alice possesses a Bitcoin token-binding UTxO $\sigma_{\mathcal{A}}$ and its corresponding shadow UTxO $\sigma'_{\mathcal{A}}$.

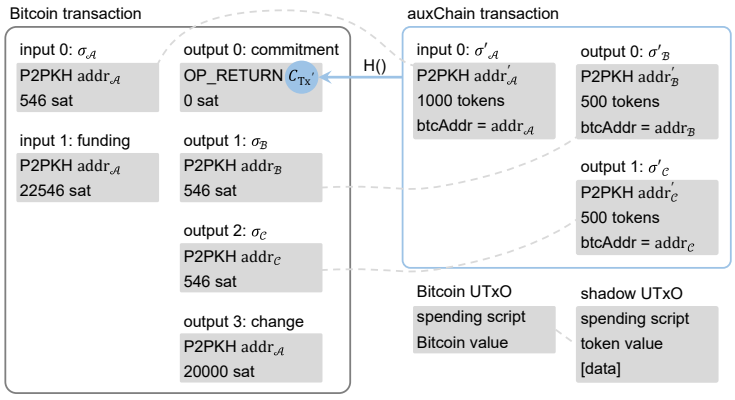

Fig. 1. A pair of transactions illustrating basic UTXO binding, where Alice transfers 500 tokens to Bob and another 500 to Carol. In the Bitcoin transaction, input 1 provides the 2000-satoshi transaction fee and the additional values of the token-binding UTxOs, and output 3 receives the remaining bitcoins. We omit the auxChain transaction's additional inputs and outputs, listing only those used to compute the commitment. In autonomous UTXO binding, we replace the auxChain's spending script with a script validating the three conditions detailed in Sec. 4.4, and the data field with the txid and index of the corresponding Bitcoin UTxO.

Verifying Alice's sufficient token balance is straightforward due to the public transaction history. Bob has provided Alice with his Bitcoin address $\mathsf{addr}_{\mathcal{B}}$ and auxChain address $\mathsf{addr}'_{\mathcal{B}}$. In our design, the token-binding UTxO is generated within the transaction, eliminating Bob's need to pre-assign an existing Bitcoin UTxO.

Alice begins by constructing, without broadcasting, an auxChain transaction $\mathsf{Tx}'_{\mathcal{A}\to\mathcal{B}}$ that sends tokens to $\mathsf{addr}'_{\mathcal{B}}$ and including $\mathsf{addr}_{\mathcal{B}}$ in its data field. She then computes a commitment $C_{\mathsf{Tx}'} = H(\mathsf{Tx}'_{\mathcal{A}\to\mathcal{B}})$. If certain transaction fields remain unfixed before confirmation, they can be left blank during the commitment computation. Next, Alice constructs a Bitcoin transaction that spents $\sigma_{\mathcal{A}}$ as the first input. The first output of the transaction has zero value and embeds $C_{\mathsf{Tx}'}$ after `OP_RETURN`. We can also commit to $C_{\mathsf{Tx}'}$ via Taproot, as described in Appendix A. The second output transfers 546 satoshis——the minimum transferable Bitcoin amount——to $\mathsf{addr}_{\mathcal{B}}$. This second output becomes Bob's token-binding UTxO $\sigma_{\mathcal{B}}$. Additional *funding inputs* might be required to cover transaction fees, and a *change output* might be necessary to collect the remaining bitcoins. Once the Bitcoin transaction is confirmed, Alice broadcasts $\mathsf{Tx}'_{\mathcal{A}\to\mathcal{B}}$ on the auxChain network. The output of this transaction becomes Bob's shadow UTxO $\sigma'_{\mathcal{B}}$. Bob then verifies that the spending transactions of $\sigma_{\mathcal{A}}$ and $\sigma'_{\mathcal{A}}$ commit to both of his addresses and that the histories of $\sigma_{\mathcal{A}}$ and $\sigma'_{\mathcal{A}}$ are consistent back to the genesis UTxO and its shadow.

Generalizing these conditions to multiple-input-multiple-output transactions is straightforward. Replace $\mathsf{Tx}'_{\mathcal{A}\to\mathcal{B}}$ with a multiple-input-multiple-output trans-

action. The transaction inputs consist of an array of n_s input shadow UTxOs, denoted as $\sigma'_{\mathrm{sender}}[]$, where each element represents a sender. The transaction outputs consist of an array of n_r newly generated shadow UTxOs, denoted as $\sigma'_{\mathrm{receiver}}[]$, where each element targets a receiver by embedding the corresponding Bitcoin address in its data field. The corresponding Bitcoin transaction's inputs must include all Bitcoin token-binding UTxOs, $\sigma_{\mathrm{sender}}[]$, that correspond to the input shadow UTxOs as the first n_s inputs. Additionally, it must include all newly generated token-binding UTxOs as the second to $(n_\mathrm{r}+1)$-th outputs, maintaining the same order as in the shadow transaction.

As the token contract is enforced by auxChain's consensus, the clients' only remaining task is a consistency check between the Bitcoin and auxChain transactions. This check is indispensable because without an on-chain Bitcoin light client, a shadow transaction might be confirmed despite the absence of its corresponding Bitcoin transaction, or even if it corresponds to a different receiver. In such cases, as in any other CSV designs, the receiver must not deem the transaction settled. Our second variant relieves the receiver of this burden.

4.4 Autonomous UTxO Binding

When the auxChain offers an on-chain Bitcoin light client, the receiver provides a Bitcoin address to the sender and requires only auxChain access for token transaction verification.

Issuance. Token issuance is nearly identical to the previous design, with some additional spending conditions in shadow genesis UTxOs. These conditions, which apply to all shadow UTxOs, will be specified later.

Transaction. Let Alice be the sender and Bob the receiver. Alice possesses a Bitcoin token-binding UTxO $\sigma_{\mathcal{A}}$ with address $\mathsf{addr}_{\mathcal{A}}$. The corresponding shadow UTxO is $\sigma'_{\mathcal{A}}$. She intends to transfer $\mathsf{value}_{\mathcal{B}}$ tokens to Bob, whose Bitcoin address is $\mathsf{addr}_{\mathcal{B}}$.

The main workflow mirrors the previous design. Alice constructs a commitment $C_{\mathsf{Tx}'}$ and embeds it after `OP_RETURN` of the first output of $\sigma_{\mathcal{A}}$'s spending transaction. The second output $\sigma_{\mathcal{B}}$ transfers 546 satoshis to $\mathsf{addr}_{\mathcal{B}}$. Upon confirmation of the spending transaction, Alice reveals the commitment in a corresponding auxChain transaction, creating the shadow UTxO $\sigma'_{\mathcal{B}}$. This design differs from the previous one in the commitment's content and the shadow UTxOs' spending conditions.

The commitment $C_{\mathsf{Tx}'}$ is defined as $H(\sigma'_{\mathcal{A}}, 1, \mathsf{value}_{\mathcal{B}})$. The number "1" indicates that a single output, i.e., the second one, is a token-binding UTxO. We specify the commitment's content to highlight the exclusion of both auxChain and Bitcoin addresses. The Bitcoin address is omitted because it will be committed within the second output of the Bitcoin transaction. The auxChain address is unnecessary due to the specialized spending conditions outlined below.

Beyond the token-specific contract, the shadow genesis UTxO defines spending conditions applicable to all shadow UTxOs, ensuring the proper expenditure of their associated Bitcoin UTxOs. These conditions are:

Condition 1. The corresponding Bitcoin token-binding output is spent with commitment $C_{\mathsf{Tx}'}$, and the Bitcoin transaction is confirmed.

Condition 2. The full content of commitment $C_{\mathsf{Tx}'}$ is reconstructible from the shadow UTxO's spending transaction and is computed correctly. This verification process includes checking the correct specification of the spent shadow UTxO(s) and the token value(s) of the newly generated shadow UTxO(s).

Condition 3. The newly generated shadow UTxO(s) must replicate the spending conditions of the current one.

Condition 1 is validated through the on-chain Bitcoin light client. This requires the following: (1) the full Bitcoin transaction is stored on auxChain, and (2) the shadow UTxO spending script can access the txid and index of its Bitcoin correspondence. The method by which the Bitcoin transaction is supplied to the auxChain varies depending on the specific auxChain implementation. The Bitcoin token-binding output's txid and index can be conveyed either as a data entry or output parameter during the creation of the shadow UTxO, or as an input parameter during its spending. With access to the Bitcoin transaction, the spending script can extract the list of newly generated Bitcoin token-binding UTxOs and their corresponding addresses, including $\mathsf{addr}_{\mathcal{B}}$, and bind them with their shadow UTxOs for further reference. Consequently, there is no need to explicitly transfer $\mathsf{addr}_{\mathcal{B}}$ to the auxChain. Since the entire transactional logic resides on the auxChain, Bob only needs to verify the existence of a valid shadow UTxO on the auxChain. This shadow UTxO must contain amount $\mathsf{value}_{\mathcal{B}}$ of the desired token, and reference a Bitcoin output that targets $\mathsf{addr}_{\mathcal{B}}$ at the correct index of the Bitcoin transaction, which is also recorded on the auxChain. Notably, $\sigma'_{\mathcal{A}}$ is not secured by Alice's public key; any entity possessing the commitment content can construct this transaction.

To generalize these conditions to multiple-input-multiple-output transactions, we replace $\sigma'_{\mathcal{A}}$ with an array of n_{s} senders $\sigma'_{\mathrm{sender}}[]$, the number 1 with the number of receivers n_{r}, and $\mathsf{value}_{\mathcal{B}}$ with an array of output values $\mathsf{value}_{\mathrm{receiver}}[]$. All newly generated shadow UTxOs must specify the same spending conditions. The corresponding Bitcoin transaction's inputs must include all Bitcoin token-binding UTxOs, $\sigma_{\mathrm{sender}}[]$, as the first n_{s} inputs. Additionally, it must include all newly generated token-binding UTxOs as the second to $(n_{\mathrm{r}}+1)$-th outputs, maintaining the same order as in the shadow transaction.

5 Security Evaluation

Theorem 1. *Both UTxO binding designs achieve exclusive binding.*

Proof. Since the issuer is honest, the Bitcoin genesis UTxO and the shadow genesis UTxO are exclusively bound. Assume all existing UTxO pairs are exclusively bound, and we will prove that any newly generated pairs, if they pass the validity checks, are also exclusively bound.

Valid transactions come in pairs. Each Bitcoin token-binding UTxO's spending transaction corresponds to exactly one auxChain transaction, because the

involved shadow UTxOs can be spent only once. This also holds in the reverse direction.

If a transaction pair is accepted by the public (basic UTxO binding) or the auxChain (autonomous UTxO binding), then the following arguments hold: (1) the first output of the Bitcoin transaction commits to the auxChain transaction; (2) each shadow output maps to exactly one Bitcoin output. Our design ensures the second condition by mandating the i-th shadow output maps to the $(i+1)$-th output of the Bitcoin transaction. In basic UTxO binding, further assurance is provided by the embedding of a Bitcoin address within the output's data field.

Consider any shadow output $\sigma'_{\mathcal{B}}$ of the auxChain transaction. By argument (2) above, it maps to a unique Bitcoin output, denoted as $\sigma_{\mathcal{B}}$, with address $\mathsf{addr}_{\mathcal{B}}$. Conversely, starting from this Bitcoin output $\sigma_{\mathcal{B}}$, argument (1) dictates that its generation transaction and the embedded commitment uniquely identify an auxChain output. Since the Bitcoin transaction, the auxChain transaction, and its commitment must align for the transaction pair to be accepted, this identified auxChain output must be $\sigma'_{\mathcal{B}}$. Therefore, we have proven the exclusive binding between any pair of new UTxOs.

By induction, all accepted pairs exhibit exclusive binding. □

Theorem 2. *Both UTxO binding designs achieve unforgeability.*

Proof. By assumption, the token issuance is successful. For a newly accepted pair of transactions, assuming its entire transaction history is unforgeable, we will prove that the new pair is also unforgeable.

In basic UTxO binding, the attacker cannot forge either transaction, as both are signed by the same sender, who maintains consistency to prevent token loss. Similarly, in autonomous UTxO binding, the attacker cannot forge the Bitcoin transaction, as the sender signs it. This transaction commits to the number of receivers, their Bitcoin addresses and values, preventing the attacker from double-spending, over-issuing, or invalidating any tokens.

By induction, all accepted pairs are unforgeable. □

Theorem 3. *Both UTxO binding designs achieve public verifiability.*

Proof. The genesis transaction pair is accessible and verifiable through reliable channels. By examining all descendants of the shadow genesis transaction, we can reconstruct the complete list of shadow transactions and shadow UTxOs. For each shadow transaction, its corresponding Bitcoin transaction can be located by computing the commitment and searching the Bitcoin blockchain. As the full transaction details and the entire token contract reside on the auxChain, we can verify the correct construction of all transactions. □

Theorem 4. *Autonomous UTxO binding achieves contractual integrity.*

Proof. Since all evaluation rules for exclusive binding and unforgeability are enforced by the auxChain's smart contract, autonomous UTxO binding achieves contractual integrity. □

6 Implementation on CKB

We implemented autonomous UTxO binding on CKB and launched the protocol in April 2024. Our implementation[1] leverages CKB's on-chain Bitcoin light client and its robust feature set. This section details our implementation and presents its performance metrics.

6.1 Implementing Autonomous UTxO Binding

We detail our implementation by first specifying the storage of key data to ensure accessibility by validating scripts and unforgeability, and then describing our workflow and validation logic.

Key Data Structures. For every CKB transaction with n_s inputs and n_r outputs carrying UTxO binding tokens, we mandate that these token-carrying inputs and outputs must occupy the first positions in the transaction. Other inputs and outputs, such as funding inputs providing storage capacity and change outputs collecting remaining storage, must be listed after the token-carrying inputs and outputs. Additionally, we require that each CKB transaction and each Bitcoin transaction involves at most one token type. The commitment of this transaction is computed as

$$C_{\mathsf{Tx}'} = H(n_\mathrm{s}||n_\mathrm{r}||\mathsf{ckb\_tx.inputs}[0:n_\mathrm{s}]||\mathsf{ckb\_tx.outputs\_raw}[0:n_\mathrm{r}]) \ ,$$

where $\mathsf{ckb\_tx.inputs}[0:n_\mathrm{s}]$ are the token-carrying inputs, and $\mathsf{ckb\_tx.outputs\_raw}[0:n_\mathrm{r}]$ are the token-carrying outputs without their lock script `args` fields. The Bitcoin transaction committing $C_{\mathsf{Tx}'}$ is constructed as described in Sec. 4.4.

The confirmation of this Bitcoin transaction finalizes its txid. An entry $\mathsf{btc\_utxo} = (\mathsf{txid}, \mathsf{index})$ is then added to each shadow output's `args` as a lock script parameter. We prescribe that a shadow output with index i ($0 \leq i \leq n_\mathrm{r} - 1$) must point to the Bitcoin output with index $i + 1$. This $\mathsf{btc\_utxo}$ entry is accessed by the shadow transaction's input lock script and when the shadow UTxO is spent. The shadow transaction's input lock script retrieves the Bitcoin transaction and its corresponding inclusion proof from the CKB transaction's witness. This design choice is motivated by the principle that, for benign senders, the $\mathsf{btc\_utxo}$ exhibits greater persistence than the corresponding Bitcoin block. As a result, our implementation offers enhanced protection against transient Bitcoin reorganization, as detailed in Appendix B.

Main Workflow and Validation Scripts. The main workflow follows Sec. 4.4, with two additional steps between broadcasting the Bitcoin and CKB transactions. Specifically, after the Bitcoin transaction is deemed irreversible, the sender (1) adds the $\mathsf{btc\_utxo}$ entry to each shadow output's `args`, and (2) uploads the Bitcoin transaction to the CKB Bitcoin light client. For the light client to accept

[1] The source code for RGB++, a multi-component project, is available at `https://github.com/utxostack`. It encompasses features beyond those outlined here.

the transaction as authentic, the sender might provide supplementary information, such as a series of block headers to validate the proof of work and a Merkle proof confirming the transaction's inclusion in a block. Although we refer to "the sender" for simplicity, these steps can be performed by any party with access to the Bitcoin blockchain. Once completed, the sender broadcasts the shadow transaction.

All UTxOs carrying a token must use the same type script, which identifies the token. This ensures token ID uniqueness and enforces the token contract. It also ensures that all token-carrying UTxOs reference the same `code_hash` in their lock script, thereby satisfying Condition 3 in Sec. 4.4. The logic of this lock script is detailed below.

The lock script contains the main complexity. A lock script is executed only when its UTxO is used as the transaction's input, that is, when the shadow UTxO is spent. To spend a shadow UTxO $\sigma'_{\mathcal{A}}$, its lock script fetches (1) its Bitcoin UTxO $\sigma_{\mathcal{A}}$ from its `btc_utxo` entry in `args`, which is supplied when $\sigma'_{\mathcal{A}}$ is generated, and (2) `Tx`, the spending transaction of $\sigma_{\mathcal{A}}$ from the `btc_utxo` entry of any newly generated shadow UTxO. The script then verifies the following conditions. First, $\sigma_{\mathcal{A}}$ is among the inputs of `Tx` unless `Tx` is an issuance transaction, validating Condition 1. Second, it recomputes $C_{\mathsf{Tx}'}$ from the current transaction and checks whether it matches the commitment in `Tx`, validating Condition 2. Third, all newly generated shadow UTxOs embed `Tx` in their `btc_utxo` entry, and their `index` values are correctly numbered. The shadow UTxO is spendable if all three conditions are met.

6.2 Performance Metrics

By March 2015, less than a year after our mainnet launch, 581 tokens has been issued. Of these, eight tokens has over a thousand holders. The most widely adopted token, `Seal`, has 42 560 holders and a peak of 1 883 transactions within 24 hours.

Computational Costs. The execution times of our type and lock scripts are negligible, as they involve only string comparisons and hash operations that scale linearly with the length of the Bitcoin and CKB transactions. The primary complexity lies in proving the authenticity of the Bitcoin transaction to the on-chain Bitcoin light client, which requires accessing the Bitcoin blockchain. However, fetching a Bitcoin transaction and its associated data consumes far less resource than running a Bitcoin full node, a requirement often imposed by other CSV protocols.

Collaterals and Transaction Fees. To be spendable, each Bitcoin token-binding UTxO must carry 546 satoshis, equivalent to less than 0.5 USD as of March 2025. Each shadow UTxO occupies 158 bytes of storage on CKB, costing less than 0.74 USD. These collaterals are locked with the tokens and are not spent. The transaction fee for a Bitcoin transaction, which varies with its size, typically remains within 2 USD. The CKB transaction fee is less than 0.01 USD. The Bitcoin transaction and its corresponding inclusion proof are conveyed to

	Data storage	Programmability	Privacy	Deployment
Colored Coins, Omni Layer, Counterparty	Bitcoin	✗	○ public	✓
Ordinals	off-chain	✗	○ public	✓
RGB, Taproot Assets	off-chain	✗	◐ selective	✓
Intmax2, Shielded CSV	off-chain	✗	● high	✗
Lightning Network	off-chain	✗	◐ off-chain	✓
UTxO Binding	Nervos CKB	✓	○ public	✓

Table 1. Comparison with other Bitcoin layer 2 protocols.

the lock script as witnesses and, therefore, do not occupy any cell's capacity or incur extra fees.

Transaction Confirmation Latency. A transaction is considered final only after both the Bitcoin and CKB transactions are confirmed. At a minimum, this process takes approximately 10 minutes, given Bitcoin's average block time of 10 minutes and CKB's block time of 8 seconds. However, to mitigate the risk of blockchain forks, we recommend users wait for six Bitcoin block confirmations, which extends the confirmation time to around one hour.

7 Comparison with Other Bitcoin Layer 2 Protocols

This section compares UTxO binding with several influential payment protocols, commonly categorized as *Bitcoin layer 2 protocols*. To maintain a focused scope, the comparison is limited to protocols that issue or operate tokens directly on the Bitcoin blockchain, including those detailed in Sec. 2, alongside the Lightning Network and UTxO binding. Table 1 provides a detailed comparison of these protocols using four key metrics.

Data Storage. This metric assesses how transaction data are stored and accessed. Colored Coins, Omni Layer, and Counterparty store data directly on the Bitcoin blockchain, which offers limited storage. Ordinals, Taproot Assets, Intmax2, Shielded CSV, and the Lightning Network store data off-chain, placing the responsibility on individual users. Consequently, users risk losing tokens if specific state data is lost. UTxO binding, in contrast, stores data on CKB, which provides greater storage capacity and enhanced reliability.

Programmability. This metric evaluates the protocol's support for complex smart contract logic. All listed protocols, except for RGB which plans to incorporate programmability, are limited to basic token transfers. UTxO binding inherits CKB's expressive programmability.

Privacy. This metric assesses the level of privacy offered by each protocol. Most protocols, including UTxO binding, provide weak privacy guarantees due to the

public nature of transaction data. RGB offers limited recipient privacy, as the receiver's token-binding UTxO remains hidden until it is spent. However, spending the UTxO reveals it, along with its entire transaction history, to the subsequent receiver. Furthermore, the `OP_RETURN` output in the spending transaction publicly indicates the presence of a token-binding UTxO among the inputs. Taproot Assets offers slightly stronger anonymity, as token-binding UTxOs are indistinguishable from other Taproot outputs, thus we categorize them as offering "selective" disclosure. The Lightning Network provides "off-chain" privacy, as some transactions occur outside the Bitcoin blockchain, complicating the reconstruction of the complete transaction history for attackers. Intmax2 and Shielded CSV offer stronger privacy through encrypting transaction histories. We aim to enhance privacy in future updates.

Deployment. Most protocols are deployed, with Intmax2 and Shielded CSV being the exceptions.

8 Conclusion

CSV is the most promising approach to achieving the long-standing goal of issuing tokens on Bitcoin. However, existing designs have failed to gain wide adoption due to issues with data availability, peer discovery, client coherence, and state integrity. In this paper, we address these limitations by introducing UTxO binding, which securely links a Bitcoin UTxO to a UTxO on an auxiliary blockchain. We defined the desired properties and formally proved that our designs meet these properties. Moreover, to demonstrate its feasibility and efficiency, we implement our design on CKB. We hope that this design, along with our implementation, can finally unleash the high demand for Bitcoin-based tokens. Furthermore, UTxO binding showcases a new direction to enrich Bitcoin's functionalities and leverage its battle-tested security without modifying the Bitcoin consensus, which could enable new possibilities for the Bitcoin ecosystem.

Acknowledgements. The authors extend their gratitude to Chaotic[1], jjy[2], EthanYuan[3], and Flouse[4] for their contributions to the implementation of CKB scripts. Additionally, we acknowledge ahonn[5], Dylan[6], Dawn[7], Shook[8], and Flouse for their work on the RGB++ SDK and assets API. Finally, we express our appreciation to Ajian, cyberorange, Jan Xie, Shawn, and DaPangDun for their valuable contributions and insightful discussions.

[1] https://github.com/chaoticlonghair
[2] https://github.com/jjyr
[3] https://github.com/EthanYuan
[4] https://github.com/Flouse
[5] https://github.com/ahonn
[6] https://github.com/duanyytop
[7] https://github.com/Dawn-githup
[8] https://github.com/ShookLyngs

References

1. Al-Bassam, M.: Lazyledger: A distributed data availability ledger with client-side smart contracts. arXiv preprint arXiv:1905.09274 (2019), https://arxiv.org/pdf/1905.09274
2. Al-Bassam, M., Sonnino, A., Buterin, V., Khoffi, I.: Fraud and data availability proofs: Detecting invalid blocks in light clients. In: Financial Cryptography and Data Security. pp. 279–298. Springer (2021)
3. Aumayr, L., Avarikioti, Z., Linus, R., Maffei, M., Pelosi, A., Stefo, C., Zamyatin, A.: BitVM: Quasi-Turing Complete Computation on Bitcoin. Cryptology ePrint Archive (2024), https://eprint.iacr.org/2024/1995.pdf
4. Buterin, V.: Ethereum: A next-generation smart contract and decentralized application platform (2014), https://github.com/ethereum/wiki/wiki/White-Paper
5. Chakravarty, M.M., Chapman, J., MacKenzie, K., Melkonian, O., Peyton Jones, M., Wadler, P.: The extended UTXO model. In: Financial Cryptography and Data Security: FC 2020 International Workshops, AsiaUSEC, CoDeFi, VOTING, and WTSC. pp. 525–539. Springer (2020)
6. CoinMarketCap: Bitcoin dominance (2025), https://coinmarketcap.com/charts/bitcoin-dominance/
7. Coinmarketcap: Cryptocurrency sectors (2025), https://coinmarketcap.com/cryptocurrency-category/
8. Counterparty: Counterparty (2025), https://www.counterparty.io/
9. Docs, R.: Commitment Schemes within Bitcoin and RGB (Accessed 2024-Nov-22), https://docs.rgb.info/commitment-layer/commitment-schemes
10. Garay, J., Kiayias, A., Leonardos, N.: The Bitcoin backbone protocol: analysis and applications. Journal of the ACM **71**(4), 1–49 (2024)
11. Hearn, M.: Smart property (2011), https://bitcointalk.org/index.php?topic=41550.0
12. Linus, R., Aumayr, L., Zamyatin, A., Pelosi, A., Avarikioti, Z., Maffei, M.: BitVM2: Bridging Bitcoin to Second Layers (2024), https://bitvm.org/bitvm_bridge.pdf
13. Liu, Y.: Issuing assets on bitcoin: A simple guide to current projects and approaches (2024), https://blog.cryptape.com/issuing-assets-on-bitcoin-a-simple-guide-to-current-projects-and-approaches
14. Möser, M., Eyal, I., Sirer, E.G.: Bitcoin covenants. In: Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC. vol. 9604, pp. 126–141. Springer (2016)
15. Nick, J., Eagen, L., Linus, R.: Shielded CSV: Private and Efficient Client-Side Validation (2024), https://github.com/ShieldedCSV/ShieldedCSV?tab=readme-ov-file
16. O'Connor, R., Piekarska, M.: Enhancing Bitcoin transactions with covenants. In: Financial Cryptography and Data Security - FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA. vol. 10323, pp. 191–198. Springer (2017)
17. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: EUROCRYPT 2017. vol. 10211, pp. 643–673 (2017)
18. Polygon: Avail - the data availability blockchain (2021), https://github.com/maticnetwork/data-availability
19. Ether Alpha: Client diversity | Ethereum (2025), https://clientdiversity.org/
20. Hope C: Consensys' L2 Linea Halts Block Production Following Velocore DEX Hack (2024), https://finance.yahoo.com/news/consensys-l2-linea-halts-block-051657899.html

21. Lightning Labs: Taproot Assets (2025), https://docs.lightning.engineering/the-lightning-network/taproot-assets/taproot-assets-protocol
22. Nervos Foundation: Nervos network (2025), https://www.nervos.org/
23. Omni Team: Omni Layer (2025), https://www.omnilayer.org/
24. Rosenfeld, M.: Overview of Colored Coins (2012), https://allquantor.at/blockchainbib/pdf/rosenfeld2012overview.pdf
25. Rybakken, E., Hioki, L., Yaksetig, M.: Intmax2: A ZK-rollup with Minimal Onchain Data and Computation Costs Featuring Decentralized Aggregators. Cryptology ePrint Archive (2023), https://eprint.iacr.org/2023/1082.pdf
26. Sompolinsky, Y., Zohar, A.: Secure high-rate transaction processing in Bitcoin. In: Financial Cryptography and Data Security. vol. 8975, pp. 507–527. Springer (2015)
27. Tas, E.N., Boneh, D.: Cryptoeconomic security for data availability committees. In: Financial Cryptography and Data Security. pp. 310–326. Springer (2023)
28. Todd, P.: Scalable semi-trustless asset transfer via single-use-seals and proof-of-publication (2016), https://petertodd.org/2017/scalable-single-use-seal-asset-transfer
29. Wuille, P., Nick, J., Towns, A.: Bip341-taproot: Segwit version 1 spending rules (2019), https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki
30. Xiao, X.: Data structures of Nervos CKB (2019), https://github.com/nervosnetwork/rfcs/blob/master/rfcs/0019-data-structures/0019-data-structures.md
31. Zhang, R., Preneel, B.: Lay down the common metrics: Evaluating proof-of-work consensus protocols' security. In: 40th IEEE Symposium on Security and Privacy (S&P). pp. 1190–1207. IEEE (May 2019)

A Committing via Taproot

Our designs explicitly embed the commitment on the Bitcoin blockchain via `OP_RETURN` in a dummy output. Alternatively, by leveraging the Taproot upgrade, we can remove this dummy output and embed the commitment within the Bitcoin token-binding UTxO. Specifically, we can replace the Bitcoin transaction's receiving address $\mathsf{addr}_{\mathcal{B}}$ with a Taproot address $\mathsf{addr}^{\mathrm{t}}_{\mathcal{B}}$, encoded from the tweaked public key $\mathsf{P}^{\mathrm{t}}_{\mathcal{B}} = \mathsf{P}_{\mathcal{B}} + C_{\mathsf{Tx}'} \cdot \mathrm{G}$, where $\mathsf{P}_{\mathcal{B}}$ is the public key corresponding to $\mathsf{addr}_{\mathcal{B}}$. The tweaked private key is thus $\mathsf{p}^{\mathrm{t}}_{\mathcal{B}} = \mathsf{p}_{\mathcal{B}} + C_{\mathsf{Tx}'}$, where $\mathsf{p}_{\mathcal{B}}$ is the private key corresponding to $\mathsf{P}_{\mathcal{B}}$. The shadow transaction remains unchanged. Note that in this case, the Bitcoin token-binding outputs becomes the transaction's first to n_{r}-th outputs, rather than the second to $(n_{\mathrm{r}}+1)$-th.

This commitment approach, compared to using `OP_RETURN`, presents two advantages and one disadvantage. First, it conserves Bitcoin on-chain space. Second, aligning with Taproot Assets, the token-binding UTxO appears indistinguishable from a standard Taproot output. Conversely, it introduces additional computation in autonomous UTxO binding. Specifically, the auxChain must verify the proper construction of the Taproot addresses when deriving $\sigma_{\mathcal{B}}$ from $\sigma'_{\mathcal{B}}$.

B Resistance against Bitcoin Reorganization

Our implementation safeguards user assets against transient Bitcoin reorganization, also known as *forks*, because btc_utxo entry used to identify the Bitcoin token-binding UTxO is persistent after a Bitcoin reorganization. This safeguard stems from the fact that a Bitcoin reorg affects only the blocks and their corresponding inclusion proofs, leaving benign txids unaltered. Since the affected fields—the block inclusion proof—reside within the CKB transaction's witness, which is inaccessible by subsequent transactions, the CKB blockchain remains unaware of and unaffected by the Bitcoin fork.

Specifically, should the Bitcoin block be invalidated after the CKB transaction's confirmation, the shadow UTxOs become temporarily unspendable due to the lack of its txid on the Bitcoin blockchain. The sender can rebroadcast the Bitcoin transaction within the Bitcoin network. Upon successful confirmation, the txid will match its prior value. Consequently, the btc_utxo entries on the CKB blockchain remain unaffected, and the shadow UTxOs become spendable once more. Conversely, if the sender attempts to double-spend tokens on Bitcoin, the Bitcoin txid will alter, rendering the shadow UTxOs that commit to the original btc_utxo unspendable.