CONFIDENTIAL

UNITED KINGDOM INTELLECTUAL PROPERTY OFFICE

PATENT APPLICATION

Applicant

The Bitcoin Corporation Ltd

Inventor

Richard Boase

Date of Preparation

8 March 2026

Title of Invention

Content Distribution Network Membership and Access Control Using Limited-Supply Blockchain Ticket Tokens with Integrated Staking, Trading, and Revenue-Sharing Mechanics

Field of the Invention

The present invention relates to decentralised content distribution networks built upon blockchain infrastructure, and more particularly to a system in which access to digital content is controlled by limited-supply UTXO-based tokens (“tickets”) that simultaneously function as (a) access credentials, (b) tradeable membership instruments, (c) staking instruments conferring dividend eligibility, and (d) demand signals driving content distribution economics. The invention addresses the absence of a unified token primitive that connects content access, content economics, and network participation in a single on-chain instrument.

Background of the Invention

Problem Statement

1. **Content Access and Content Economics Are Disconnected**

   In existing content distribution systems, the mechanism by which a user accesses content (a subscription, a one-time purchase, a paywall cookie) is entirely separate from the economic instrument representing the content's value. A Netflix subscription grants access but cannot be resold, staked, or used to participate in the content's economic upside. A Patreon pledge pays the creator but confers no tradeable asset. There is no instrument that unifies "I can access this content" with "I hold an economic position in this content."

2. **CDN Membership Has No Market Price**

   Traditional content delivery networks (CDNs) — both centralised (Cloudflare, Akamai) and decentralised (BitTorrent, IPFS) — treat participation as binary: a user either has access or does not. There is no price discovery mechanism for content access rights. A piece of content that is in high demand costs the same to access as one that nobody wants. The CDN cannot signal scarcity, and creators cannot benefit from secondary market activity around their content.

3. **BitTorrent's Missing Payment Layer**

   BitTorrent solved content distribution at scale but has no native payment or ownership layer. Seeders contribute bandwidth without compensation. Leechers consume without paying. There is no mechanism for a content creator to monetise distribution, for a seeder to earn from hosting popular content, or for supply and demand to set prices. Attempts to add payment layers (e.g., BitTorrent Token) bolted a token onto a system that was not designed for economic coordination, resulting in misaligned incentives.

4. **NFTs Without Utility**

   Non-fungible tokens (NFTs) on various blockchains represent ownership of digital assets but typically confer no functional access rights. Owning an NFT of an artwork does not grant access to a high-resolution version, a creator's private channel, or any ongoing economic benefit. The token is a certificate of ownership disconnected from the content delivery infrastructure. Conversely, access-granting tokens (API keys, subscription tokens) have no ownership semantics and cannot be traded.

5. **No Mechanism for Demand-Driven Content Economics**

In existing systems, a creator sets a price and users pay it. There is no mechanism by which the act of accessing content feeds back into the content's economic value for stakeholders. A viral piece of content generates no additional economic benefit for early supporters or for nodes hosting it. The network cannot self-organise around high-demand content because there is no economic signal connecting demand to distribution incentives.

Prior Art Limitations

Subscription platforms (Netflix, Spotify, Patreon) provide access but not ownership. Subscriptions cannot be resold, subdivided, or staked. They create recurring revenue for creators but no secondary market and no way for fans to participate in economic upside.

NFT marketplaces (OpenSea, Magic Eden) provide ownership and trading but not content access. Buying an NFT does not grant functional access to a content delivery network. NFTs are collectibles, not access credentials.

BitTorrent and IPFS provide decentralised content distribution but no payment layer, no access control, and no economic coordination between creators, distributors, and consumers.

Existing token-gated access systems (e.g., holding a specific NFT to access a Discord channel) provide binary access control but the token itself has no connection to the content delivery infrastructure, no staking mechanics, no dividend eligibility, and no lifecycle (the token is never "consumed" by accessing content).

NFT ticketing platforms (Ticketmaster/Live Nation, GET Protocol, YellowHeart, Aventus) — Multiple platforms have implemented event tickets as blockchain tokens. Ticketmaster mints NFT ticket stubs on the Flow blockchain for commemorative purposes. GET Protocol mints every ticket as an ERC-721 NFT on Polygon with smart contract controls on secondary resale (price caps, transfer restrictions). YellowHeart issues NFT tickets with coded resale royalties flowing to the artist/issuer. Aventus provides Ethereum-based ticketing where AVT tokens are staked for network security and governance. All of these platforms differ from the present invention in several respects: (i) they use account-based blockchain NFTs (Flow, Ethereum, Polygon), not UTXO-based tokens encoded with `OP_FALSE OP_RETURN` scripts; (ii) none implement a three-state lifecycle (held/staked/redeemed) where redemption removes the ticket from circulation as a deflationary mechanism; (iii) secondary sale royalties flow to the issuer/artist in existing systems, not to stakers as proportional dividends from a distribution pool; (iv) none tie CDN participation incentives to the economic activity of the tickets being tracked; and (v) existing NFT tickets are event access tokens, not content delivery network membership instruments that simultaneously function as access credentials, tradeable assets, and staking instruments with dividend rights.

Theta Network (US Patent 11,075,891, "NFT-based digital rights management in a decentralized data delivery network", 2021) — Theta uses NFT ownership to gate access to encrypted data streams in a decentralised content delivery network. Source nodes broadcast content, and viewers holding the relevant NFT receive decryption keys. This is the closest prior art for combining token-gated content access with decentralised content delivery. However, Theta's NFTs are persistent access credentials that are never consumed or redeemed — there is no three-state lifecycle. Theta does not implement staking-for-dividends from secondary market activity, does not use fixed-supply deflationary dynamics (tickets leaving circulation upon redemption), and does not encode tokens as UTXO outputs with structured `OP_RETURN` metadata. Theta nodes earn TFUEL for bandwidth relay, not for economic activity of specific content tokens. Theta provides decentralised CDN with token gating; the present invention provides a unified token primitive that simultaneously functions as access credential, economic instrument, and CDN incentive mechanism with lifecycle-driven supply dynamics.

LooksRare LOOKS Token Staking (2022) — The LooksRare NFT marketplace distributed 100% of platform trading fees to LOOKS token stakers as WETH, creating a direct staking-to-dividends-from-secondary-market mechanism. This is the closest prior art for the staking dividend concept. However, LOOKS is a platform-wide governance token — stakers earn from all trading on the platform, not from trading of a specific content item's tickets. LOOKS does not function as an access credential, has no three-state lifecycle, no deflationary redemption mechanism, no fixed supply per content item, and no CDN integration. LooksRare distributes platform-wide trading fees; the present invention distributes per-content secondary sale revenue to stakers of that specific content's tickets.

nChain UTXO Tokenization (CA3013180A1, US20190303887A1, 2017–2019) — nChain Holdings disclosed methods for representing tokens as UTXOs on the Bitcoin blockchain with metadata encoded in redeem scripts, covering general tokenization of assets including tickets, property deeds, and currency tokens. These patents cover the infrastructure layer — how to represent tokens as UTXOs with embedded metadata. However, nChain's disclosures describe a general tokenization framework and do not define the specific application of a three-state ticket lifecycle (held/staked/redeemed), do not implement staking-for-dividends from secondary market trading of same-content tokens, do not tie CDN node compensation to per-content ticket economic activity, and do not combine the access-credential, tradeable-asset, and staking-instrument functions into a single unified token primitive.

The Applicant's prior HTTP Status Code Tokenization Suite patent defines a regulatory taxonomy ($401 identity, $402 content/commodity, $403 securities) but does not specify the ticket token lifecycle, staking mechanics, dividend accrual from secondary sales, or the relationship between ticket supply, content access, and CDN economics.

No existing system combines: limited-supply UTXO ticket tokens that simultaneously function as access credentials, tradeable assets, and staking instruments; a three-state lifecycle (held/staked/redeemed) where redemption consumes the token from circulation; a secondary market revenue loop distributing per-content trading fees to stakers of that specific content's tickets; fixed-supply scarcity driving market-based price discovery for content access rights; and CDN participation incentives tied to the economic activity of content-specific tickets rather than to general bandwidth or storage contribution.

Summary of the Invention

The present invention provides a unified token primitive for decentralised content distribution networks, comprising:

a. A **Ticket Token** — a limited-supply UTXO on a blockchain, minted by a content creator for a specific piece of content, where each ticket simultaneously represents an access credential, a tradeable asset, and a potential staking instrument, and where the total supply of tickets for a given content item is fixed at creation time by the creator;

b. A **Ticket Lifecycle** comprising three terminal states: (i) **held** — the ticket is owned by a user and may be traded, staked, or redeemed; (ii) **staked** — the ticket is temporarily locked, removing it from circulation and conferring dividend eligibility on the staker; (iii) **redeemed** — the ticket is returned to the creator's address, granting the redeemer access to the content and consuming the ticket from active circulation;

c. A **Secondary Market Revenue Loop** whereby every transfer of a ticket between parties for payment generates a sale event that is recorded on-chain, and a configurable portion of each sale accrues as dividends to current stakers of tickets for the same content item — creating a direct economic link between content demand (trading activity) and stakeholder returns;

d. A **Supply-Demand Pricing Signal** whereby the fixed supply of tickets per content item, combined with the removal of redeemed and staked tickets from active circulation, creates natural scarcity that allows market price discovery — high-demand content drives ticket prices up, directly benefiting creators (who can re-mint redeemed tickets) and stakers (who earn dividends from higher-value sales);

e. A **CDN Participation Incentive** whereby nodes in the content distribution network that index, host, and serve ticket state and content data earn compensation tied to the economic activity of the tickets they track — aligning content distribution work with content economics.

Detailed Description of the Invention

1. System Architecture

1.1 Ticket Token Structure

Each ticket token is a UTXO (unspent transaction output) on a Bitcoin-protocol blockchain, conforming to a token standard (e.g., BSV-21) with the following properties:

- **Content Identifier**: A unique identifier linking the ticket to a specific piece of digital content (e.g., a hash of the content, a content URI, or a creator-defined identifier).
- **Ticket Number**: A sequential identifier within the content's ticket supply (e.g., ticket #7 of 100).
- **Creator Address**: The blockchain address of the content creator who minted the ticket.
- **Current Holder**: The address currently controlling the UTXO (the ticket owner).
- **State Flag**: One of ACTIVE, STAKED, or REDEEMED (encoded in the UTXO's script or metadata).

1.1.1 Ticket Token UTXO Script Template

Each ticket UTXO contains an output script with the following structure. The locking script is a standard Pay-to-Public-Key-Hash (P2PKH) script for the current holder, followed by an `OP_RETURN` data carrier output encoding the ticket metadata. The `OP_RETURN` output has the following byte layout:

```
OP_FALSE OP_RETURN
  <protocol_prefix>      // 4 bytes: 0x5449434B ("TICK" in ASCII)
  <content_id>           // 32 bytes: SHA-256 hash of the content
payload
  <ticket_number>        // 4 bytes: uint32, big-endian (0x00000000 to
0xFFFFFFFF)
  <total_supply>         // 4 bytes: uint32, big-endian — fixed at mint
time
  <creator_address>      // 25 bytes: Base58Check-decoded Bitcoin
address
                         //   (1 byte version + 20 byte pubkey hash + 4
byte checksum)
  <state_flag>           // 1 byte: 0x01 = ACTIVE, 0x02 = STAKED, 0x03 =
REDEEMED
  <dividend_bps>         // 2 bytes: uint16, big-endian — dividend
percentage in
                         //   basis points (e.g., 0x03E8 = 1000 bps =
10%)
  <mint_block_height>    // 4 bytes: uint32, big-endian — block at which
ticket was minted
```

The **Content Identifier** is computed as follows: the creator concatenates the raw content bytes with a 32-byte creator-chosen nonce, then computes `content_id = SHA-256(content_bytes || nonce)`. The nonce prevents content identifier collisions when different creators publish identical content and is disclosed alongside the content for verification.

The **Ticket Number** is a zero-indexed sequential integer: the first ticket minted for a content item has ticket number `0x00000000`, the second `0x00000001`, and so on up to `total_supply - 1`. The TopicManager rejects any minting transaction where the ticket number is greater than or equal to `total_supply`.

The **State Flag** encodes the lifecycle state as a single byte: - `0x01` (ACTIVE): the ticket is held and transferable. - `0x02` (STAKED): the ticket is locked in a staking contract. - `0x03` (REDEEMED): the ticket has been returned to the creator.

The state flag in the `OP_RETURN` output is set at transaction creation time, but the canonical state is determined by the TopicManager's validation of the spending conditions (see sections 1.3 and 1.4). If the `OP_RETURN` state flag contradicts the TopicManager's assessment of the transaction structure, the TopicManager rejects the transaction as invalid.

The ticket is a standard spendable UTXO. Transferring it requires a valid transaction signed by the current holder. This means tickets inherit all properties of UTXOs: they can be

transferred peer-to-peer without intermediaries, their provenance is traceable via the transaction graph, and their existence is provable via Merkle proofs to block headers.

1.2 Content Creator Minting

A content creator mints tickets by constructing a minting transaction that produces N ticket UTXOs for a given content identifier, where N is chosen by the creator. The minting transaction is admitted to the overlay network's ticket topic (TopicManager), which validates:

- The creator is the registered owner of the content identifier.
- The total supply does not exceed the creator's declared limit.
- Each ticket has a unique ticket number within the content's supply.

Once minted, tickets enter circulation and can be acquired by users via direct sale from the creator, secondary market purchase, or any other UTXO transfer mechanism.

1.3 Ticket Lifecycle States

1.3.1 ACTIVE (Held)

An ACTIVE ticket is a standard spendable UTXO held by a user. The holder may:

- **Transfer** it to another user (creating a sale event if payment is involved).
- **Stake** it by sending it to a staking contract address (transitioning to STAKED state).
- **Redeem** it by returning it to the creator's address (transitioning to REDEEMED state and granting content access).

1.3.2 STAKED

A STAKED ticket is locked in a time-locked or condition-locked UTXO script. The staker retains economic ownership but cannot transfer or redeem the ticket until unstaking. While staked:

- The ticket is removed from active circulation, reducing tradeable supply.
- The staker becomes eligible for dividends from future sale events for the same content item.
- Dividend eligibility begins at the block height of the staking transaction.
- The staker may unstake at any time (subject to any minimum lock period set by the creator), returning the ticket to ACTIVE state.

1.3.2.1 Staking Contract Script

To stake a ticket, the holder constructs a transaction that spends the ticket UTXO and produces a new UTXO with the following locking script:

```
<min_lock_blocks> OP_CHECKSEQUENCEVERIFY OP_DROP
OP_DUP OP_HASH160 <staker_pubkey_hash> OP_EQUALVERIFY OP_CHECKSIG
```

Where: - `<min_lock_blocks>` is a 3-byte little-endian integer encoding the minimum number of blocks before the ticket can be unstaked. This value is set by the content creator at mint time and embedded in the minting transaction's `OP_RETURN` metadata. If the creator specifies no minimum lock period, this value is `0x000000` (zero blocks), and the `OP_CHECKSEQUENCEVERIFY OP_DROP` prefix is omitted entirely, allowing immediate unstaking. - `<staker_pubkey_hash>` is the HASH160 (RIPEMD-160 of SHA-256) of the staker's public key, ensuring only the staker can unstake.

The `OP_CHECKSEQUENCEVERIFY` (CSV) opcode enforces a relative time lock: the staking UTXO cannot be spent until `min_lock_blocks` blocks have been mined after the block containing the staking transaction. The transaction's `nSequence` field must be set to a value greater than or equal to `min_lock_blocks` with the relative lock-time type flag (bit 22) set to 0 (block-height-based).

Alternatively, for absolute time locking (e.g., "staked until block 900,000"), the creator may specify `OP_CHECKLOCKTIMEVERIFY` (CLTV):

```
<unlock_block_height> OP_CHECKLOCKTIMEVERIFY OP_DROP
OP_DUP OP_HASH160 <staker_pubkey_hash> OP_EQUALVERIFY OP_CHECKSIG
```

Where `<unlock_block_height>` is a 4-byte little-endian integer specifying the absolute block height after which the UTXO becomes spendable. The transaction's `nLockTime` must be set to a value greater than or equal to `<unlock_block_height>`.

The staking transaction must also produce an `OP_RETURN` output carrying the ticket metadata with `state_flag` set to `0x02` (STAKED). The TopicManager validates staking transactions by confirming: (a) the input spends a valid ACTIVE ticket UTXO, (b) the output locking script matches one of the two permitted staking script templates above, and (c) the `OP_RETURN` metadata preserves the `content_id`, `ticket_number`, `total_supply`, `creator_address`, and `dividend_bps` fields unchanged from the input.

1.3.3 REDEEMED

A REDEEMED ticket has been returned to the creator's address. This is the content access event:

- The overlay network records the redemption as a spend event on the ticket topic.
- The content distribution network grants the redeemer access to the content (via a signed access token, a decryption key, or a direct content delivery).
- The redeemed ticket is removed from circulation. The creator may choose to re-mint it (creating a new UTXO with the same ticket number) or permanently retire it.

1.3.3.1 Redemption Verification

The TopicManager validates a redemption transaction by performing the following checks in order:

1. **Input validity**: The transaction input spends a valid ticket UTXO that is currently in ACTIVE state (`state_flag == 0x01`) according to the overlay's indexed state. If the ticket is in STAKED state, the redemption is rejected — the holder must first unstake.
2. **Destination address check**: The transaction output carrying the ticket UTXO value (excluding the `OP_RETURN` data output) must pay to the `creator_address` recorded in the ticket's `OP_RETURN` metadata. The TopicManager extracts the `pubkey_hash` from the output's P2PKH script and compares it byte-for-byte with the `creator_address` field (bytes 44–68) of the input ticket's `OP_RETURN` . If they do not match, the redemption is rejected.
3. **State flag update**: The `OP_RETURN` output of the redemption transaction must set `state_flag` to `0x03` (REDEEMED) and preserve all other metadata fields unchanged.
4. **Content delivery authorisation token derivation**: Upon successful validation, the TopicManager computes an authorisation token that the redeemer will use to access the content:

```
auth_token = HMAC-SHA-256(
    key   = creator_content_secret,
    data  = redemption_txid || redeemer_pubkey
)
```

Where: - `redemption_txid` is the 32-byte transaction ID of the redemption transaction. - `redeemer_pubkey` is the public key used to sign the redemption transaction input (extracted from the scriptSig). - `creator_content_secret` is a per-content secret held by the creator's

node (or by the overlay's content delivery service on the creator's behalf), never published on-chain.

The `auth_token` is returned to the redeemer via the LookupService query interface and is valid indefinitely (or until the creator revokes the `creator_content_secret` and re-keys).

This consumption model means that accessing content has a real cost denominated in a market-priced asset. Unlike a flat subscription, the cost of access is determined by the market value of the ticket, which reflects genuine demand.

1.4 Secondary Market and Revenue Loop

When an ACTIVE ticket is transferred between parties for payment (a sale), the sale transaction is admitted to the overlay's sale topic. The sale event contains:

- The ticket identifier (content ID + ticket number).
- The sale price in satoshis.
- The buyer and seller addresses.

Upon admission of a sale event, the overlay's dividend LookupService:

1. Identifies all tickets for the same content item currently in STAKED state.
2. Computes each staker's proportional share (e.g., if there are 100 tickets total, 20 are staked, and Staker A holds 5 staked tickets, Staker A's share is 5/20 = 25%).
3. Computes the dividend amount as a configurable percentage of the sale price (set by the creator at minting time, e.g., 10%).
4. Increments each staker's `dividends_owed` balance by their proportional share of the dividend amount.

1.4.1 Dividend Computation Algorithm

The dividend LookupService executes the following algorithm upon each admitted sale event. All arithmetic uses integer satoshi values to avoid floating-point rounding errors.

```
FUNCTION process_sale_event(sale_event):
    content_id       ← sale_event.content_id
    sale_price_sats  ← sale_event.sale_price    // uint64, in satoshis
    dividend_bps     ← lookup_dividend_bps(content_id)  // uint16, basis
points

    // Step 1: Compute total dividend pool for this sale (integer
division)
    dividend_pool_sats ← (sale_price_sats * dividend_bps) / 10000

    // Step 2: Identify all staked tickets for this content_id
    staked_tickets ← query_staked_tickets(content_id)
    //   Returns list of {staker_address, ticket_count} pairs

    total_staked ← SUM(staked_tickets[i].ticket_count for all i)

    // Step 3: Handle edge case — zero stakers
    IF total_staked == 0:
        // No stakers to distribute to. The dividend pool is forfeited.
        // It remains unallocated and is NOT returned to the seller or
buyer.
        RETURN

    // Step 4: Distribute dividends proportionally using integer
arithmetic
    distributed_sats ← 0
    FOR EACH staker IN staked_tickets:
        staker_share_sats ← (dividend_pool_sats * staker.ticket_count) /
total_staked
        staker.dividends_owed ← staker.dividends_owed +
staker_share_sats
        distributed_sats ← distributed_sats + staker_share_sats

    // Step 5: Handle rounding remainder
    //   Integer division may leave undistributed satoshis.
    //   Remainder is always < total_staked (at most total_staked - 1
satoshis).
    remainder_sats ← dividend_pool_sats - distributed_sats
    IF remainder_sats > 0:
        // Award remainder to the staker with the longest continuous
stake
        // (earliest staking block height). Ties broken by lowest
ticket_number.
        longest_staker ← staker with MIN(stake_block_height),
                         then MIN(ticket_number) among tied stakers
```

```
        longest_staker.dividends_owed ← longest_staker.dividends_owed +
remainder_sats
```

The `dividends_owed` balance for each staker is maintained as derived state in the LookupService's persistent storage, keyed by `(content_id, staker_address)`. This balance is incremented by the algorithm above and decremented when the staker submits a valid claim transaction (see section 2, step 6).

1.4.2 Sale Event Detection and Classification

The overlay must distinguish a **sale** (transfer for payment) from a **gift** (transfer without payment). A transaction is classified as a sale event if and only if at least one of the following conditions holds:

1. **Explicit sale marker**: The transaction contains an `OP_RETURN` output with the protocol prefix `0x53414C45` ("SALE" in ASCII) followed by the `content_id` (32 bytes) and `ticket_number` (4 bytes) of the ticket being sold, and a `sale_price` field (8 bytes, uint64 little-endian) declaring the price in satoshis. The TopicManager validates that the declared `sale_price` is non-zero.
2. **Payment output heuristic**: In the absence of an explicit sale marker, the TopicManager applies the following heuristic: the transaction is classified as a sale if (a) the transaction has at least one non-`OP_RETURN` output paying to the seller's address (the address that held the ticket UTXO in the input), and (b) the value of that output is at least 1 satoshi. The `sale_price` is taken as the total value of all outputs paying to the seller's address, excluding the ticket UTXO output itself and any change outputs identifiable by the seller's address appearing in both inputs and outputs.
3. **No sale indicators**: If neither condition (1) nor condition (2) is met — i.e., the ticket UTXO is transferred to a new address with no payment outputs to the previous holder and no explicit sale marker — the transfer is classified as a **gift** and does not trigger dividend accrual.

Creators may mandate explicit sale markers at mint time by setting a `require_explicit_sale` flag (`0x01`) in the minting transaction's `OP_RETURN` metadata (appended after `mint_block_height`), in which case only condition (1) triggers dividend accrual.

This creates a revenue loop: content demand drives ticket trading → trading generates sale events → sale events generate dividends for stakers → dividend yields attract more staking → more staking reduces circulating supply → reduced supply increases ticket prices → higher prices generate larger dividends per sale.

1.5 Supply-Demand Price Discovery

The fixed supply of tickets per content item, combined with three removal mechanisms (staking, redemption, and natural holding), creates genuine scarcity. The market price of a ticket at any time reflects:

- **Demand for content access**: Users who want to access the content must acquire and redeem a ticket. High demand for access drives ticket prices up.
- **Demand for dividend yield**: Investors who want dividend income from popular content must acquire and stake tickets. This further reduces supply.
- **Creator re-minting decisions**: When a creator re-mints redeemed tickets, they add supply. When they retire tickets, supply permanently shrinks.

This price discovery mechanism is absent from all existing content distribution systems. It provides creators with a real-time market signal of their content's value, provides investors with a yield-bearing instrument tied to content popularity, and provides the network with an economic coordination mechanism for resource allocation.

1.6 Content Delivery Integration

When a user redeems a ticket (returns it to the creator's address), the content delivery network must deliver the content to the redeemer. The system supports multiple delivery mechanisms:

- **Direct delivery**: The creator's node serves the content directly to the redeemer, authenticated by the redemption transaction signature.
- **CDN delivery**: Nodes in the overlay network that host the content serve it to the redeemer. The redeemer presents the redemption transaction ID as proof of access. Nodes are compensated via CARS (Cloud Automated Runtime System) or equivalent hosting compensation.
- **Decryption key delivery**: The content is stored encrypted on the network. Upon redemption, the creator (or an automated service) delivers a decryption key to the redeemer, derivable from the redemption transaction.

In all cases, the redemption transaction on the blockchain serves as the unforgeable access credential. No separate authentication system, API key, or session token is required.

1.6.1 Content Delivery Authentication Protocol

When a redeemer requests content from a CDN node, the following authentication handshake is performed:

1. **Redeemer presents credentials**: The redeemer sends an HTTP request to the CDN node with the following headers:
    - `X-Redemption-TxID` : the 32-byte redemption transaction ID, hex-encoded (64 characters).
    - `X-Redeemer-Signature` : a signature over the concatenation `redemption_txid || current_unix_timestamp_seconds` (40 bytes), signed with the private key corresponding to the redeemer's public key used in the redemption transaction. The timestamp provides replay protection.
    - `X-Redeemer-PubKey` : the redeemer's compressed public key (33 bytes, hex-encoded).
    - `X-Timestamp` : the current Unix timestamp in seconds (uint64, decimal string).
2. **CDN node verification**: The CDN node performs the following checks:
    - Verifies `X-Timestamp` is within 300 seconds (5 minutes) of the node's current time. Rejects if stale.
    - Reconstructs the signed message: `redemption_txid_bytes || timestamp_bytes` (40 bytes).
    - Verifies `X-Redeemer-Signature` against `X-Redeemer-PubKey` using ECDSA on the secp256k1 curve.
    - Queries the overlay LookupService to confirm that `X-Redemption-TxID` is a valid, admitted redemption transaction for the requested `content_id` , and that `X-Redeemer-PubKey` matches the public key in the redemption transaction's scriptSig.
    - If all checks pass, the node serves the content. If any check fails, the node returns HTTP 403.
3. **Alternative: auth_token mode**: If the CDN node holds the `creator_content_secret` (or can query it from the creator's key server), the redeemer may instead present the `auth_token` (computed as described in section 1.3.3.1) in an `Authorization: Bearer <auth_token_hex>` header. The CDN node recomputes `HMAC-SHA-256(creator_content_secret, redemption_txid || redeemer_pubkey)` and compares it in constant time with the presented token. This mode avoids per-request signature computation by the redeemer.

1.7 CDN Node Economics

Nodes in the content distribution network that host overlay services (TopicManagers, LookupServices, content storage) earn compensation from multiple sources:

1. **CARS hosting compensation**: Satoshis for compute resources consumed (CPU, memory, storage, bandwidth), paid by overlay project owners.
2. **Lookup service fees**: Micropayments from users querying the LookupService for ticket state, dividend balances, or content metadata (paywalled via BRC-41).
3. **Overlay token rewards**: Proof of Indexing rewards (overlay-specific tokens) for maintaining accurate state across ticket, sale, stake, and claim topics.
4. **Content delivery fees**: Satoshis for serving content to redeemers, proportional to content size and bandwidth consumed.

This multi-revenue model means that nodes hosting popular content (which generates more trading, more queries, and more redemptions) earn more than nodes hosting unpopular content — naturally incentivising the network to allocate resources toward high-demand content, mirroring BitTorrent's organic scaling but with explicit economic compensation.

1.8 Creator Economics

Content creators benefit from the ticket system through multiple revenue streams:

1. **Primary sales**: Revenue from initial ticket sales to first buyers.
2. **Re-minting**: When redeemed tickets are re-minted and sold again, creators earn additional primary sale revenue. Popular content generates repeated redemption-and-remint cycles.
3. **Creator staking**: Creators can stake their own unsold tickets, earning dividends from secondary market activity. This aligns creator incentives with long-term content value rather than one-time sales.
4. **Supply management**: Creators can increase scarcity (and thus price) by retiring redeemed tickets rather than re-minting them, or by setting high re-mint prices.

2. Operational Flow

1. **Creator publishes content**: A creator uploads content to the decentralised network and mints N tickets for it, setting the ticket supply, the dividend percentage, and the initial sale price.
2. **Primary distribution**: Tickets are sold to initial buyers. Each sale is a standard UTXO transfer. The overlay's ticket topic admits each new ticket UTXO.
3. **Secondary trading**: Buyers resell tickets to other users. Each transfer-for-payment is admitted to the sale topic. The dividend LookupService processes each sale event, incrementing staker balances.

4. **Staking**: Holders who want dividend income stake their tickets by locking them in staking contracts. The stake topic admits staking events. Staked tickets are removed from circulation.
5. **Dividend accrual**: As sale events occur, dividends accrue to stakers proportionally. Stakers can query the dividend LookupService for their current balance at any time.
6. **Dividend claiming**: Stakers claim accumulated dividends by constructing a claim transaction. The claim topic admits the payout UTXO. The dividend LookupService decrements the staker's balance.
7. **Content access**: A user who wants to access the content acquires a ticket (from primary sale, secondary market, or by unstaking) and redeems it by returning it to the creator's address. The ticket topic records the spend event. The content delivery network serves the content.
8. **Re-minting**: The creator optionally re-mints the redeemed ticket, returning it to circulation for another cycle.
9. **Network scaling**: Nodes observe which content items generate the most ticket activity and allocate hosting resources accordingly. The network self-organises around demand.

3. Comparison with BitTorrent

The present invention can be understood as BitTorrent reimagined with a native payment and ownership layer:

BitTorrent	**Present Invention**
.torrent file identifies content	Content identifier on blockchain
Swarm of seeders/leechers	Overlay network of indexing nodes
No payment for seeding	CARS compensation + lookup fees + PoI tokens
No access control	Ticket redemption as access credential
No price discovery	Market-priced tickets with fixed supply
No creator revenue after upload	Primary sales + re-minting + creator staking dividends
No investor participation	Staking for dividend yield
Tracker maintains peer state	LookupService maintains ticket/dividend state
DHT for decentralised discovery	SHIP/SLAP for node/service discovery

The key architectural insight is that BitTorrent's tracker (which maintains derived state about who has what pieces) maps directly to the overlay LookupService (which maintains derived

state about who holds what tickets and what dividends are owed). The present invention adds economic primitives (tickets, staking, dividends) to this architecture, creating a self-sustaining content distribution economy.

Brief Description of Drawings

- **Figure 1** — Ticket token lifecycle state diagram showing transitions between ACTIVE, STAKED, and REDEEMED states, with triggering actions (transfer, stake, unstake, redeem, re-mint).
- **Figure 2** — Revenue loop diagram showing the cycle: content demand → ticket purchase → sale event → dividend accrual → staking incentive → reduced supply → price increase → larger dividends.
- **Figure 3** — System architecture showing the relationship between the blockchain layer (ticket UTXOs), the overlay layer (TopicManagers for tickets/sales/stakes/claims, LookupServices for state), and the content delivery layer (encrypted storage, access control).
- **Figure 4** — Creator economics flow showing revenue streams: primary sales, re-minting cycles, creator staking dividends, and supply management decisions.
- **Figure 5** — CDN node economics showing four revenue sources: CARS hosting compensation, lookup service fees, Proof of Indexing token rewards, and content delivery fees.
- **Figure 6** — Comparison architecture diagram showing BitTorrent components mapped to their equivalents in the present invention.
- **Figure 7** — Supply-demand dynamics showing how fixed ticket supply, staking lock-up, and redemption consumption interact to determine market price.

Initial Claims

Note: These claims are provided in sketch form for the purposes of establishing a priority date. Formal claims will be drafted and filed within 12 months in accordance with UKIPO rules.

Claim 1 — Unified Content Access and Economic Token

A blockchain-based token system for content distribution networks, comprising:

a. a limited-supply set of ticket tokens, each being a UTXO on a blockchain, minted by a content creator for a specific piece of digital content, where each ticket simultaneously functions as a content access credential, a tradeable asset, and a stakeable dividend-bearing instrument;

b. a ticket lifecycle comprising at least three states: ACTIVE (held and transferable), STAKED (locked, conferring dividend eligibility, removed from tradeable circulation), and REDEEMED (returned to creator, granting content access, consumed from circulation);

c. a secondary market revenue mechanism whereby each transfer of a ticket between parties for payment generates an on-chain sale event, and a configurable portion of each sale price accrues as dividends to holders of tickets for the same content item that are in the STAKED state; and

d. a content delivery mechanism whereby redemption of a ticket (transition to REDEEMED state) serves as the authentication credential for content access, requiring no separate authentication system.

Claim 2 — Supply-Demand Price Discovery for Content Access

A method for market-based price discovery of digital content access rights, comprising:

a. a content creator minting a fixed supply of ticket tokens for a piece of content, each token being a blockchain UTXO;

b. three mechanisms reducing the active circulating supply of tickets: staking (temporary removal), redemption (consumption for content access), and natural holding;

c. the market price of tickets being determined by supply and demand, where demand for content access (requiring redemption) and demand for dividend yield (requiring staking) both reduce supply and increase price; and

d. the content creator optionally re-minting redeemed tickets to manage supply, creating repeating redemption-and-remint cycles that generate ongoing revenue.

Claim 3 — Self-Organising Content Distribution Network

A decentralised content distribution network, comprising:

a. content identified and access-controlled by limited-supply ticket tokens on a blockchain;

b. a plurality of nodes hosting overlay services that index ticket state, process sale events, compute dividend balances, and serve content to redeemers;

c. nodes earning compensation from multiple sources tied to the economic activity of the tickets they track, said sources comprising at least hosting compensation for compute resources and micropayments for lookup queries; and

d. the network self-organising around high-demand content, as nodes hosting content with more ticket trading activity, more lookup queries, and more redemptions earn proportionally more compensation.

Claim 4 — Dividend Accrual from Content Ticket Trading

A method for distributing revenue from content trading to stakeholders, comprising:

a. recording each transfer-for-payment of a content ticket token as a sale event on a blockchain overlay network;

b. upon each sale event, identifying all ticket tokens for the same content item currently in a staked state;

c. computing each staker's proportional share of a dividend amount derived from the sale price;

d. incrementing each staker's dividend balance in a derived state maintained by a lookup service; and

e. upon a staker's claim, constructing a settlement transaction that produces a spendable UTXO on the blockchain for the claimed dividend amount, and decrementing the staker's derived balance accordingly.

Claim 5 — Ticket Redemption as Content Access Credential

A method for controlling access to digital content using blockchain tokens, comprising:

a. a user acquiring a ticket token (a blockchain UTXO) associated with a specific piece of digital content;

b. the user redeeming the ticket by constructing a transaction that transfers the ticket UTXO to the content creator's address;

c. the redemption transaction being recorded on the blockchain and admitted to a content overlay network as a spend event;

d. the content distribution network verifying the redemption transaction and delivering the content to the redeemer, using the redemption transaction signature as the sole authentication credential; and

e. the redeemed ticket being removed from active circulation, with the creator optionally re-minting it as a new UTXO for future sale.

Abstract

A system and method for decentralised content distribution using limited-supply blockchain ticket tokens that unify content access control, economic trading, staking, and dividend distribution in a single on-chain instrument. A content creator mints a fixed number of ticket tokens (blockchain UTXOs) for a piece of digital content. Each ticket can be held and traded (ACTIVE), locked for dividend eligibility (STAKED), or returned to the creator to access the content (REDEEMED). Secondary market sales of tickets generate dividend accruals for stakers, creating a revenue loop where content demand drives trading, trading generates dividends, dividends incentivise staking, and staking reduces circulating supply — producing market-based price discovery for content access rights. Ticket redemption serves as the content access credential, eliminating the need for separate authentication systems. Network nodes that index ticket state and serve content earn compensation proportional to the economic activity of the content they host, causing the network to self-organise around high-demand content. The invention combines the decentralised distribution architecture of BitTorrent with a native payment and ownership layer, addressing BitTorrent's missing economic coordination and NFTs' missing functional utility.

Document prepared for UKIPO filing. Priority date to be established upon submission. Applicant: The Bitcoin Corporation Ltd Inventor: Richard Boase Date of preparation: 8 March 2026