CONFIDENTIAL

UNITED KINGDOM INTELLECTUAL PROPERTY OFFICE

PATENT APPLICATION

Applicant

The Bitcoin Corporation Ltd

Inventor

Richard Boase

Date of Preparation

10 March 2026

Title of Invention

Tokenised Patent Licensing via Bonding Curve Tokens with Identity-Bound Graduated Rights and On-Chain Licence Verification

---

Field of the Invention

The present invention relates to systems and methods for licensing intellectual property — specifically patents — through blockchain-based tokens issued via a bonding curve pricing mechanism, where the quantity of tokens held by a licensee determines the scope of the licence granted. More particularly, the invention concerns a system in which the patent specification itself is inscribed on a blockchain as the token's content payload, such that the act of purchasing tokens simultaneously acquires the licence, proves the licence holder's identity, records the licence grant on-chain, and funds the patent holder — with no separate licence agreement, no negotiation, no intermediary, and no ambiguity about the terms, which are themselves inscribed immutably alongside the patent text.

Background of the Invention

Problem Statement

Patent licensing suffers from structural inefficiencies that have persisted for over a century:

1. **Opaque and Bespoke Negotiation** — Every patent licence is individually negotiated between licensor and licensee, typically through lawyers, over weeks or months. There is no standardised pricing mechanism. Two licensees may pay vastly different amounts for equivalent rights to the same patent, with no transparency about what others have paid. This opacity benefits intermediaries (lawyers, brokers) at the expense of both patent holders and licensees.
2. **Binary Licensing** — Patent licences are typically all-or-nothing: a licensee either has a licence or does not. There is no native mechanism for graduated licensing where the scope of permitted use (personal, startup, enterprise, unlimited) is proportional to the amount paid. A solo developer experimenting with a patented technique and a multinational corporation deploying it at scale must negotiate from the same starting point. The result is that small innovators either infringe unknowingly, avoid patented technologies entirely, or cannot afford to engage with the licensing process.
3. **No Verifiable Licence Registry** — When a licensee claims to hold a patent licence, there is no publicly verifiable registry that a third party can check. Licence agreements are private contracts, often subject to confidentiality clauses. A downstream customer, a supply chain partner, or a court must rely on the licensee producing the original agreement — which may be lost, disputed, or fabricated. There is no equivalent of a land registry for patent licences.
4. **Licence Transfer Friction** — Patent licences are typically non-transferable or require licensor consent to transfer. When a company is acquired, merged, or restructured, licence transfer requires renegotiation. There is no mechanism for a licence to move with an asset or entity automatically, without the licensor's active involvement.
5. **No Economic Participation for Early Licensees** — In traditional patent licensing, the first licensee pays the same rate structure as the hundredth. There is no mechanism by which early adoption of a patented technology is rewarded. The patent holder captures all economic upside from growing adoption; licensees who took the risk of adopting early receive no economic benefit when the technology proves successful and later licensees arrive.
6. **Disconnection Between Patent Text and Licence** — A patent specification is published by the patent office. A licence agreement is a separate document, drafted by

lawyers, referencing the patent by number. The two documents exist in different systems, different formats, and different jurisdictions. There is no unified object that is simultaneously the patent disclosure, the licence terms, and the payment mechanism.

7. **No Machine-Readable Licensing** — As AI agents increasingly evaluate and adopt technologies on behalf of their operators, there is no mechanism by which an autonomous software agent can discover a patent, evaluate its terms, acquire a licence, and prove compliance — all without human intervention. Patent licensing is a fundamentally human-mediated process incompatible with autonomous operation.

Prior Art Limitations

Patent offices (UKIPO, USPTO, EPO) publish patent specifications and maintain registers of patent ownership, but do not operate licensing registries, do not provide pricing mechanisms, and do not facilitate licence transactions.

Patent licensing platforms (e.g., patent pools, licensing exchanges) aggregate patents for collective licensing but operate as centralised intermediaries with opaque pricing, membership requirements, and manual negotiation processes. They do not use algorithmic pricing, do not provide on-chain verification, and do not support graduated licence tiers based on token holdings.

Blockchain token systems (NFTs, fungible tokens) represent ownership and enable trading but have no native concept of graduated licence scope. Holding one NFT or one million fungible tokens grants the same rights unless a separate off-chain agreement defines otherwise. No existing token standard maps token quantity to licence tier.

IPwe and IBM patent NFTs (2021) — IPwe, in partnership with IBM, announced a system for representing patents as non-fungible tokens (NFTs) on a blockchain, enabling patents to be “sold, traded, commercialized or otherwise monetized” with a permanent chain of custody. IPwe’s system uses one NFT per patent to represent ownership, with smart contracts facilitating transactions. However, IPwe’s patent NFTs represent patent ownership as a single indivisible token — one NFT equals one patent. The system does not use fungible tokens where the quantity held determines licence scope, does not implement bonding curve pricing, does not define graduated licence tiers, does not enable a licence to be subdivided by transferring a portion of tokens, and does not provide machine-readable licence acquisition for autonomous agents. IPwe is a centralised platform requiring their involvement in transactions; the present invention operates entirely on public blockchain state without intermediaries.

Ocean Protocol bonding curves for data access (2020–present) — Ocean Protocol applied bonding curve pricing to data asset tokenization, generating approximately $6 million USD. Data tokens issued via bonding curves grant access to datasets. However, Ocean Protocol tokenizes data access, not patent licences. There are no graduated licence tiers — holding any quantity of data tokens grants the same access. There is no identity-bound licence verification, no patent office integration, and no concept of licence scope varying with token quantity. Ocean demonstrates that bonding curves can price access to intellectual assets; it does not apply this to patent licensing with graduated rights.

Academic research on patent tokenization (2022–2025) — Multiple academic publications have explored representing patents as NFTs or blockchain tokens, including “Patents and intellectual property assets as non-fungible tokens: key technologies and challenges” (PMC, 2022) and “Thermodynamic control of patent tokenization for sustainable development” (Frontiers in Blockchain, 2025). These works focus on patent ownership representation and trading — using tokens as certificates of patent ownership that can be transferred. None define a system where the quantity of fungible tokens held determines the scope of a patent licence, where a bonding curve provides deterministic pricing, or where licence status is verifiable from on-chain state without contacting any intermediary.

Story Protocol and Programmable IP Licence (PIL, 2024–present) — Story Protocol provides a dedicated Layer-1 blockchain for IP registration, licensing, and monetisation. The Programmable IP Licence (PIL) bridges off-chain legal contracts with on-chain smart contract enforcement, with three standard tiers (Non-Commercial Social Remixing, Commercial Use, Commercial Remix). IP assets can be tokenised and licence terms encoded in smart contracts, with automatic licence transfer when tokens transfer. However, Story Protocol’s tiers are type-based (commercial vs. non-commercial vs. remix), not quantity-based (more tokens = higher tier). Each licence is a separate NFT, not a fungible token where quantity determines scope. Story Protocol does not implement bonding curve pricing, does not define graduated licence tiers mapped to token quantity thresholds, does not inscribe the full patent specification as token content, and focuses on creative works and software IP rather than patent licensing specifically. Story Protocol provides programmable IP management; the present invention provides deterministic algorithmic licensing through fungible token economics.

Molecule Protocol and VitaDAO IP-NFTs (2018–present) — Molecule tokenizes biotech intellectual property as IP-NFTs, which can be fractionalised into IPTs (Intellectual Property Tokens) using ERC-20 fungible tokens. Token holders receive governance rights over licensing decisions, access to R&D data (with KYC/NDA requirements), and shares of commercialisation proceeds. Molecule has explicitly researched bonding curves for IP distribution, including sigmoid functions for pharmaceutical IP pricing (Paul Kohlhaas, “Token Bonding Curve Design Parameters”, 2019). This is the closest prior art combining bonding curves with IP tokens. However, Molecule’s IPTs grant governance over licensing decisions (voting rights), not automatic graduated licence tiers based on quantity held — all IPT holders have the same class of rights regardless of quantity. The bonding curves are used for fundraising (selling IPTs to fund research), not for pricing licence acquisition. There is no identity-bound verification, no machine-readable AI agent acquisition pathway, and no patent text inscribed as token content. Molecule fractionalises IP ownership for collective governance; the present invention creates a deterministic licensing system where token quantity directly determines licence scope.

Token-as-a-License (TaaL) concept (Denis Petrovcic, Blocksquare, 2017–2018) — The TaaL framework proposed that blockchain tokens could themselves constitute software licences, with the token plus its smart contract plus the terms forming the complete licence agreement. TaaL described SaaS, on-premise, and open-source licensing variants. This is prior art for the concept of "token IS the licence." However, TaaL was a conceptual framework for software copyright licensing, not a working system for patent licensing. TaaL did not define bonding curve pricing, graduated rights by quantity, identity-chain verification, or AI agent acquisition. The present invention implements the token-as-licence concept specifically for patents, with bonding curve economics, quantity-graduated tiers, and machine-readable discovery.

Machine-readable licensing standards (SPDX, Creative Commons CC REL, ODRL) — The Software Package Data Exchange (SPDX) provides machine-readable identifiers for open-source licences. Creative Commons provides machine-readable licence metadata via CC REL (RDFa markup). The Open Digital Rights Language (ODRL) provides a vocabulary for expressing digital rights and permissions in machine-readable form. All of these are metadata standards — they describe licence terms in structured formats that machines can parse. However, none of these standards provide on-chain verification of licence status, algorithmic pricing, graduated rights based on token holdings, or automatic licence transfer via token transfer. They are description standards; the present invention is an execution system where the token itself constitutes the licence and its quantity determines the scope of rights.

The Applicant's prior Bit Trust patent (GB2604176.4) defines a blockchain-native IP registration system with tiered trust levels, but addresses registration and proof of existence — not licensing. Bit Trust proves that IP exists and who created it; the present invention defines how that IP is licensed to others.

The Applicant's prior HTTP Status Code Token Protocol Suite patent (GB2604419.8) defines $401 identity tokens, $402 payment tokens, and $403 permission tokens, but does not specify how these primitives combine to create a patent licensing system with graduated rights.

The Applicant's prior Decentralized Dividend Distribution patent (GB2604496.6) defines how revenue is distributed to token holders in a hierarchical token tree, but does not address the specific application of dividend distribution as an economic incentive for early patent licensees.

The Applicant's prior Ticket CDN Membership Token patent defines ticket tokens with lifecycle states (active, staked, redeemed) and secondary market revenue loops, but applies these concepts to content access in a CDN — not to patent licensing with graduated rights tiers.

No existing system combines: algorithmic bonding curve pricing for patent licences; token quantity as the determinant of licence scope; on-chain inscription of patent text as token content; identity-bound licence verification via blockchain identity chains; automatic licence transfer via token transfer; economic participation for early licensees via bonding curve appreciation; and machine-readable licence discovery and acquisition.

Summary of the Invention

The present invention provides a system and method for licensing patents through blockchain tokens ("Licence Tokens"), comprising:

a. A **Patent-as-Token Inscription** mechanism whereby the full text of a patent specification is inscribed on a blockchain as the content payload of a token, such that the patent disclosure and the licensing instrument are a single on-chain object — the token IS the licence and the licence IS the patent;

b. A **Bonding Curve Pricing Engine** that determines the cost of acquiring Licence Tokens using a deterministic mathematical function of the current token supply, such that early licensees pay less per token than later licensees, creating a transparent, non-negotiable, algorithmically determined price that any party can independently compute;

c. A **Graduated Licence Tier System** whereby the scope of rights granted to a licensee is determined by the quantity of Licence Tokens they hold, with defined thresholds mapping token quantities to licence tiers — from personal/research use through startup commercial use, enterprise deployment, unlimited commercial use, sublicensing rights, and governance participation;

d. An **Identity-Bound Licence Verification** mechanism whereby each Licence Token holder's identity is established through a blockchain identity chain (the $401 protocol), enabling any third party to verify that a specific identified entity holds a sufficient quantity of tokens to qualify for a specific licence tier — without contacting the patent holder, without accessing private agreements, and without trusting the licensee's self-attestation;

e. A **Licence Grant Inscription** mechanism whereby each acquisition of Licence Tokens is recorded on the blockchain as an immutable licence grant event, creating an auditable, tamper-proof registry of all licences issued for the patent, including the licensee's identity, the number of tokens acquired, the tier achieved, and the timestamp;

f. A **Transferable Licence** mechanism whereby licence rights are transferred automatically when Licence Tokens are transferred between parties — the new holder inherits the licence tier corresponding to their token balance, and the transfer is recorded on-chain without requiring the patent holder's consent or involvement;

g. An **Early Licensee Economic Participation** mechanism whereby the bonding curve causes the value of Licence Tokens to appreciate as more licensees acquire tokens, such that early licensees who took the risk of adopting the patented technology before it was proven can realise economic gain by selling some or all of their tokens to later licensees — creating an alignment of incentives between the patent holder (who wants widespread adoption) and early licensees (who are rewarded for driving that adoption);

h. A **Machine-Readable Licence Discovery** protocol whereby an autonomous software agent can discover the patent, read the licence terms (inscribed on-chain), compute the current price (from the bonding curve), acquire tokens (via a blockchain transaction), and verify its licence status — all without human intervention, enabling fully autonomous patent licensing for AI agent systems.

Detailed Description of the Invention

1. System Architecture

The Tokenised Patent Licensing system comprises the following principal components:

1.1 Patent-as-Token Inscription

The patent specification is inscribed on the blockchain using the following process:

1. **Content Preparation** — The full patent text (specification, claims, abstract, and optionally drawings) is serialised as a UTF-8 encoded document. A cryptographic hash (SHA-256) of the document is computed.
2. **Licence Terms Attachment** — The licence terms are encoded as a structured JSON object and appended to the inscription metadata. The licence terms specify the token thresholds for each licence tier, the rights granted at each tier, the governing law, and the patent office reference number. The licence terms are themselves hashed and included in the inscription.

3. **Identity Binding** — The inscription is signed using the patent holder's private key, which is cryptographically linked to the patent holder's $401 identity chain. This binds the patent inscription to a verified identity with one or more OAuth-verified identity strands (GitHub, Google, LinkedIn, X/Twitter, Microsoft, domain ownership).
4. **On-Chain Inscription** — The signed inscription is recorded on the BSV blockchain as an immutable transaction output. The inscription contains: the patent text hash, the licence terms hash, the patent holder's $401 identity reference, a timestamp, and the patent holder's cryptographic signature. The full patent text and licence terms are stored in the transaction's data payload or in an associated encrypted vault (per the Bit Trust system), with the hash serving as the integrity anchor.
5. **Token Creation** — A token is created on the blockchain (BSV-21 standard) with the patent inscription as its genesis reference. The token has a defined maximum supply (default: 1,000,000,000 — one billion), a bonding curve pricing model, and the patent inscription transaction ID as its content reference. Every token minted carries an implicit reference back to the patent text and licence terms.

Patent Inscription Script Structure

The patent inscription is encoded as a single OP_FALSE OP_RETURN output with the following byte layout:

```
OP_FALSE OP_RETURN <fields...>

Offset  Size     Field                            Encoding
──────  ───────  ───────────────────────────────  ────────────────────────────
0       4        Protocol identifier              0x50 0x41 0x54 0x4C
("PATL")
4       1        Version byte                     0x01
5       1        Operation type                   0x01 = patent inscription
                                                  0x02 = licence grant
                                                  0x03 = tier amendment
                                                  0x04 = patent pool
composite
6       32       Patent text hash                 SHA-256 of UTF-8 patent
text
38      32       Licence terms hash               SHA-256 of JSON licence
terms
70      33       Patent holder compressed         Compressed SEC public key
                 public key
103     var      Patent office reference          UTF-8 string, length-
prefixed

                                                  (1-byte length, max 255
bytes)

                                                  e.g. "GB2604XXX.X"
var     8        Timestamp                        uint64 little-endian, Unix
                                                  epoch seconds
var     var      Patent text payload              Full UTF-8 patent
specification

                                                  (length-prefixed: 4-byte
LE)
var     var      Licence terms payload            Full JSON licence terms
                                                  (length-prefixed: 4-byte
LE)
```

Each field is pushed as a separate data item in the OP_RETURN script. The Bitcoin script structure is:

```
OP_FALSE OP_RETURN
  <4 bytes: "PATL">
  <1 byte: version>
  <1 byte: op_type>
  <32 bytes: patent_text_sha256>
  <32 bytes: licence_terms_sha256>
  <33 bytes: holder_pubkey>
  <N bytes: patent_office_ref>
  <8 bytes: timestamp>
  <M bytes: patent_text>
  <P bytes: licence_terms_json>
```

For large patent texts exceeding the transaction size limit, the patent text payload field contains the TXID (32 bytes) of a prior data-carrier transaction containing the full text, and a flag bit (bit 0 of the version byte) is set to indicate indirect reference mode.

The result is a single on-chain object that is simultaneously: (i) a published patent specification, (ii) a set of licence terms, (iii) a tradeable token, and (iv) a payment instrument. The separation between “patent” and “licence” that exists in the traditional system is collapsed into a unified primitive.

1.2 Bonding Curve Pricing Engine

The price of Licence Tokens is determined by a linear bonding curve calibrated such that acquiring 1% of the total token supply costs a defined amount (default: $1,000 USD). The pricing function operates as follows:

1.2.1 Pricing Formula

The price of the nth token (0-indexed) in USD is:

```
price(n) = c × n
```

Where `c` is a calibration constant computed as:

```
c = (2 × onePercentCostUsd) / (0.01 × totalSupply)²
```

For a 1 billion token supply with 1% costing $1,000:

```
c = (2 × 1000) / (10,000,000)² = 0.00000000002
```

This produces a linear price curve where: - Token #0 costs $0.00 (effectively free) - Token #1,000,000 costs $0.00002 - Token #10,000,000 costs $0.0002 - Token #100,000,000 costs $0.002 - Token #1,000,000,000 costs $0.02

1.2.2 Batch Purchase Formula

When a licensee spends a fixed amount of USD, the number of tokens they receive is:

```
K = floor(sqrt(pos² + (2 × amountUsd / c)) - pos)
```

Where `pos` is the current total number of tokens already sold. This formula solves for the number of tokens purchasable given a fixed budget at a given point on the bonding curve.

For example, at position 0 (no tokens sold), spending $0.01 yields:

```
K = floor(sqrt(0 + (2 × 0.01 / 0.00000000002)) - 0)
K = floor(sqrt(1,000,000,000))
K = 31,622 tokens
```

At position 100,000,000 (100 million tokens already sold), spending $0.01 yields:

```
K = floor(sqrt(100000000² + 1000000000) - 100000000)
K ≈ 4 tokens
```

This demonstrates the bonding curve's core property: early licensees receive vastly more tokens per unit of expenditure than later licensees. The first $0.01 buys 31,622 tokens; the same $0.01 at 10% saturation buys approximately 4 tokens.

1.2.3 Cost to Acquire a Licence Tier

The total cost to acquire N tokens starting from current supply position `pos` is:

```
cost(pos, pos + N) = c × ((pos + N)² - pos²) / 2
                   = c × N × (2 × pos + N) / 2
```

This allows any party to compute the exact cost of reaching any licence tier at any point in the token's lifecycle.

1.2.4 Non-Negotiability

The bonding curve price is deterministic and non-negotiable. There is no mechanism by which the patent holder can offer a discount to a preferred licensee or charge a premium to a competitor. The price is a pure function of the current token supply, which is publicly observable on the blockchain. This eliminates the opacity and bias inherent in traditional patent licensing negotiations.

The patent holder may configure the bonding curve parameters at token creation time (total supply, one-percent cost), but once set, these parameters are immutable — inscribed on-chain and enforced by the token contract.

1.2.5 Bonding Curve Purchase Transaction Structure

A licence acquisition is a single Bitcoin transaction with the following structure:

```
TRANSACTION: Licence Token Purchase
═══════════════════════════════════════════════════════════════


INPUT 0:  Licensee's UTXO
          scriptSig: <sig> <licensee_pubkey>
          (funds the purchase + miner fee)

OUTPUT 0: Payment to patent holder
          scriptPubKey: OP_DUP OP_HASH160 <patent_holder_pkh>
OP_EQUALVERIFY OP_CHECKSIG
          value: [amount computed from bonding curve, in satoshis]

OUTPUT 1: Licence Grant inscription (OP_RETURN)
          scriptPubKey: OP_FALSE OP_RETURN <PATL> <0x01> <0x02>
                        <patent_token_txid:32> <licensee_401_root:32>
                        <tokens_acquired:8> <total_balance:8>
                        <tier_code:1> <price_sats:8>
                        <curve_position:8> <timestamp:8>
          value: 0

OUTPUT 2: BSV-21 token receipt
          scriptPubKey: OP_DUP OP_HASH160 <licensee_pkh> OP_EQUALVERIFY
OP_CHECKSIG
          value: 1 (1 satoshi — token carrier)
          (BSV-21 token metadata inscribed per BSV-21 standard,
           referencing the patent token genesis TXID)

OUTPUT 3: Change
          scriptPubKey: OP_DUP OP_HASH160 <licensee_pkh> OP_EQUALVERIFY
OP_CHECKSIG
          value: [input value - output 0 value - miner fee - 1 sat]
```

The client performs the following computation steps before constructing the transaction:

```
PROCEDURE: constructPurchaseTransaction(spend_sats, patent_token_id)

1.  Retrieve patent inscription from blockchain using patent_token_id
2.  Parse bonding curve parameters: total_supply, c (calibration
constant)
3.  Query current bonding curve position: pos = total tokens already
sold
4.  Convert spend_sats to USD using on-chain or oracle exchange rate
5.  Compute tokens received:
      K = floor(sqrt(pos² + (2 × amount_usd / c)) - pos)
6.  Compute exact cost in USD for K tokens:
      cost_usd = c × K × (2 × pos + K) / 2
7.  Convert cost_usd to satoshis: cost_sats = floor(cost_usd /
bsv_usd_rate × 1e8)
8.  Compute new total balance:
      new_balance = current_licensee_balance + K
9.  Determine tier_code from new_balance against tier thresholds
10. Select licensee UTXO with sufficient value: cost_sats + miner_fee +
1
11. Construct transaction with outputs as specified above
12. Sign Input 0 with licensee's private key (linked to $401 identity)
13. Broadcast transaction to Bitcoin network
14. Await confirmation (1 block for standard, 6 blocks for Tier 4+)
```

1.3 Graduated Licence Tier System

The licence tier system maps token quantities to licence scopes. The tiers are defined in the licence terms inscribed alongside the patent text. The default tier structure is:

1.3.1 Tier Definitions

Tier 0 — Read Access - Threshold: 1+ tokens - **Rights**: Right to read and reference the patent specification. No implementation rights. - **Typical acquisition cost at position 0**: < $0.001 - **Use case**: Researchers, students, curious individuals. The equivalent of reading a patent on the patent office website, but with a token proving they engaged with it.

Tier 1 — Personal / Research - Threshold: 10,000+ tokens - **Rights**: Non-commercial implementation for personal use, academic research, educational purposes, and proof-of-concept development. No right to distribute products incorporating the patented technology. No right to sublicense. - **Typical acquisition cost at position 0**: ~$0.001 - **Use case**: University researchers, hobbyists, students building prototypes.

Tier 2 — Startup Commercial - **Threshold**: 1,000,000+ tokens - **Rights**: Commercial implementation in a single product or service, by a single legal entity, with annual gross revenue from products incorporating the patented technology not exceeding $1,000,000. No sublicensing rights. - **Typical acquisition cost at position 0**: ~$10 - **Use case**: Startups and small businesses incorporating the technology into their product.

Tier 3 — Enterprise - **Threshold**: 10,000,000+ tokens - **Rights**: Commercial implementation in unlimited products or services, by a single legal entity or corporate group, with annual gross revenue from products incorporating the patented technology not exceeding $100,000,000. No sublicensing rights. - **Typical acquisition cost at position 0**: ~$1,000 - **Use case**: Established companies deploying the technology across product lines.

Tier 4 — Unlimited Commercial - **Threshold**: 100,000,000+ tokens - **Rights**: Unlimited commercial implementation with no revenue cap. Right to sublicense to direct customers and partners. Right to create derivative works incorporating the patented technology. - **Typical acquisition cost at position 0**: ~$100,000 - **Use case**: Large corporations, platform providers, companies building the technology into standards.

Tier 5 — Governance - **Threshold**: 500,000,000+ tokens (50%+ of total supply) - **Rights**: All rights of Tier 4, plus: right to propose amendments to the licence terms (subject to on-chain voting by all token holders); right to veto licence term changes; effective co-ownership position in the patent's commercial exploitation. - **Typical acquisition cost at position 0**: ~$2,500,000 - **Use case**: Strategic acquirers, consortium partners, potential patent purchasers evaluating a full acquisition.

1.3.2 Tier Determination

A licensee's current tier is determined by their token balance at the time of use. The system queries the blockchain for the licensee's $401 identity, resolves all addresses linked to that identity, sums the Licence Token balances across all addresses, and maps the total to the applicable tier.

If a licensee sells tokens such that their balance drops below their current tier threshold, they automatically lose the rights associated with the higher tier. Their licence scope contracts to the tier corresponding to their new balance. No action by the patent holder is required — the tier is computed from on-chain state.

1.3.3 Custom Tier Configurations

The patent holder may define custom tier structures at token creation time. The default five-tier structure described above is the system default, but the patent holder may:

- Add additional tiers (e.g., a “Government” tier with specific public-sector rights);
- Adjust thresholds (e.g., lower the Enterprise threshold for a technology the patent holder wants to see widely adopted);
- Modify rights at each tier (e.g., grant sublicensing at Tier 3 for certain patent types);
- Set revenue caps appropriate to the technology’s market.

All custom tier definitions are inscribed on-chain alongside the patent text and are immutable once inscribed.

1.4 Identity-Bound Licence Verification

Licence verification uses the $401 identity protocol to bind licences to verified entities:

1. **Licensee Identity Establishment** — Before or at the time of acquiring Licence Tokens, the licensee establishes a $401 identity chain comprising one or more verified identity strands (OAuth providers). The strength of the identity (number and diversity of verified providers) does not affect the licence tier — that is determined solely by token quantity — but a stronger identity provides greater legal weight if the licence is ever disputed.
2. **Token Acquisition Binding** — When the licensee acquires Licence Tokens, the tokens are held in a blockchain address linked to their $401 identity chain. The acquisition transaction is signed by a key provably linked to the identity chain.
3. **Third-Party Verification** — Any third party wishing to verify a licensee’s licence status performs the following:
    - a. Obtain the licensee’s $401 identity reference (e.g., from a public profile, a product disclosure, or a direct enquiry).
    - b. Resolve the identity chain on the blockchain, confirming the identity’s verified strands.
    - c. Enumerate the blockchain addresses linked to the identity.
    - d. Query the token balances for the specific Patent Licence Token across all linked addresses.
    - e. Sum the balances and map to the licence tier.
    - f. Confirm that the tier grants the rights in question.

This entire verification process is performed against public blockchain data. No API call to the patent holder is required. No trust in the licensee’s self-attestation is required. The verification is cryptographically grounded in on-chain state.

Licence Verification Pseudocode

The following algorithm enables any third party to verify a licensee’s patent licence status:

```
function verifyLicence(licensee_401_txid, patent_token_id,
required_tier):
    // Step 1: Resolve the licensee's $401 identity chain
    root_tx = fetchTransaction(licensee_401_txid)
    if root_tx is null OR root_tx does not contain $401 root
inscription:
        return { valid: false, error: "INVALID_IDENTITY" }

    identity = parseIdentityRoot(root_tx)

    // Step 2: Enumerate all strand transactions linked to this root
    strands = []
    strand_txids = scanBlockchain(
        filter: OP_RETURN contains 0x34 0x30 0x31 ("401")
            AND references licensee_401_txid
    )
    for each strand_txid in strand_txids:
        strand_tx = fetchTransaction(strand_txid)
        strand = parseIdentityStrand(strand_tx)
        if strand.root_reference == licensee_401_txid:
            strands.append(strand)

    // Step 3: Collect all addresses linked to the identity
    addresses = set()
    addresses.add(extractAddress(root_tx))
    for each strand in strands:
        addresses.add(extractAddress(strand.transaction))
    // Also include any addresses declared in strand metadata
    for each strand in strands:
        if strand.declared_addresses is not empty:
            addresses.addAll(strand.declared_addresses)

    // Step 4: Sum token balances across all addresses
    total_balance = aggregateTokenBalance(addresses, patent_token_id)

    // Step 5: Retrieve tier thresholds from on-chain patent inscription
    patent_tx = fetchTransaction(patent_token_id)
    licence_terms_json = extractLicenceTerms(patent_tx)
    tiers = parseTierDefinitions(licence_terms_json)

    // Step 6: Determine achieved tier
    achieved_tier = 0
    for each tier in tiers (sorted descending by threshold):
        if total_balance >= tier.threshold:
            achieved_tier = tier.code
            break
    // Step 7: Compare against required tier
    return {
        valid: achieved_tier >= required_tier,
        achieved_tier: achieved_tier,
        required_tier: required_tier,
        total_balance: total_balance,
        identity_strands: strands.length,
        addresses_checked: addresses.size,
        block_height: currentBlockHeight()
    }
```

Token Balance Aggregation Across Identity Addresses

The following algorithm resolves all addresses linked to a $401 identity and sums BSV-21 token balances, handling the case where tokens are split across multiple UTXOs:

```
function aggregateTokenBalance(addresses, patent_token_id):
    total = 0
    utxo_details = []

    for each address in addresses:
        // Query all unspent transaction outputs for this address
        utxos = getUnspentOutputs(address)

        for each utxo in utxos:
            // Check if this UTXO carries BSV-21 tokens for the patent
            tx = fetchTransaction(utxo.txid)
            output = tx.outputs[utxo.vout]

            // Parse BSV-21 token metadata from the transaction
            token_data = parseBSV21TokenData(tx)
            if token_data is null:
                continue
            if token_data.genesis_txid != patent_token_id:
                continue

            // This UTXO carries patent licence tokens
            token_amount = token_data.amount  // uint64
            total = total + token_amount
            utxo_details.append({
                txid: utxo.txid,
                vout: utxo.vout,
                address: address,
                token_amount: token_amount
            })

    return total

function parseBSV21TokenData(tx):
    // Scan transaction outputs for BSV-21 token inscription
    for each output in tx.outputs:
        script = output.scriptPubKey
        if script contains OP_FALSE OP_RETURN:
            data = extractOpReturnData(script)
            // BSV-21 tokens use "bsv-21" or specific protocol prefix
            if data.protocol == "bsv-21":
                return {
                    genesis_txid: data.fields["id"],    // 32 bytes
                    amount: data.fields["amt"],          // uint64
                    operation: data.fields["op"]         // "transfer" |
"mint"
```

```
                  }
     return null
```

This aggregation handles the common case where a licensee has received tokens across multiple purchase transactions, or where tokens have been split across UTXOs through partial transfers. The aggregation is deterministic: given the same blockchain state, any verifier will compute the same total balance.

4. **Verification Caching** — Because token balances can change (tokens bought, sold, or transferred), verification results are valid only at the block height at which they were computed. For applications requiring ongoing licence compliance (e.g., a software product that checks its own licence status at startup), the verification should be performed periodically or triggered by relevant on-chain events (token transfers involving the licensee's addresses).

1.5 Licence Grant Inscription

Each acquisition of Licence Tokens generates a Licence Grant event that is inscribed on the blockchain:

```
LICENCE_GRANT := {
  "type": "patent_licence_grant",
  "patent_token_id": "<token identifier>",
  "patent_ref": "<patent office reference, e.g., GB2604XXX.X>",
  "licensee_401": "<$401 root inscription txid of licensee>",
  "tokens_acquired": <number of tokens acquired in this transaction>,
  "total_balance": <licensee's total token balance after acquisition>,
  "tier_achieved": "<tier name: personal | startup | enterprise |
unlimited | governance>",
  "rights_granted": ["<right1>", "<right2>", ...],
  "price_paid_usd": <amount paid in USD>,
  "price_paid_sats": <amount paid in satoshis>,
  "bonding_curve_position": <token supply at time of purchase>,
  "timestamp": <Unix epoch seconds>,
  "payment_txid": "<BSV transaction ID of the payment>"
}
```

Licence Grant Inscription Script Structure

The Licence Grant event is encoded as an OP_FALSE OP_RETURN output with the following byte layout:

```
OP_FALSE OP_RETURN <fields...>

Offset  Size     Field                           Encoding
──────  ───────  ──────────────────────────────  ──────────────────────────
0       4        Protocol identifier             0x50 0x41 0x54 0x4C
("PATL")
4       1        Version byte                    0x01
5       1        Event type                      0x02 (licence grant)
6       32       Patent token ID                 TXID of the patent
inscription
                                                 transaction (32 bytes LE)
38      32       Licensee $401 root TXID         TXID of the licensee's
$401
                                                 identity root inscription
70      8        Tokens acquired                 uint64 little-endian,
number
                                                 of tokens acquired in this
tx
78      8        Total balance                   uint64 little-endian,
licensee's
                                                 total token balance post-
tx
86      1        Tier code                       uint8: 0x00=Read,
0x01=Personal,
                                                 0x02=Startup,
0x03=Enterprise,
                                                 0x04=Unlimited,
0x05=Governance
87      8        Price paid (satoshis)           uint64 little-endian
95      8        Bonding curve position          uint64 little-endian,
total
                                                 tokens sold before this
purchase
103     8        Timestamp                       uint64 little-endian, Unix
                                                 epoch seconds
```

Total fixed payload: 111 bytes. The Bitcoin script structure is:

```
OP_FALSE OP_RETURN
  <4 bytes: "PATL">
  <1 byte: 0x01>
  <1 byte: 0x02>
  <32 bytes: patent_token_txid>
  <32 bytes: licensee_401_root_txid>
  <8 bytes: tokens_acquired>
  <8 bytes: total_balance>
  <1 byte: tier_code>
  <8 bytes: price_sats>
  <8 bytes: curve_position>
  <8 bytes: timestamp>
```

The Licence Grant inscription is signed by the system's inscription engine and references the original patent inscription. Over time, the sequence of Licence Grant inscriptions for a given patent forms a **Licence Thread** — an IP Thread (as defined in the Applicant's Bit Trust patent) specialised for licensing events.

The Licence Thread provides: - A complete, auditable history of every licence ever granted for the patent; - Evidence of commercial adoption (useful for patent valuation and litigation); - Proof of licensing revenue (useful for tax and accounting purposes); - A public signal of technology adoption (useful for market analysis).

1.6 Transferable Licences

Licence Tokens are standard blockchain tokens and can be transferred between parties without restriction:

1. **Transfer Mechanics** — A licensee transfers tokens by constructing a standard token transfer transaction, sending tokens from their address to the recipient's address. No approval from the patent holder is required.
2. **Automatic Tier Adjustment** — Upon transfer, the sender's token balance decreases and their licence tier is automatically recomputed. If the sender's balance drops below their current tier threshold, they lose the associated rights. The recipient's balance increases and their tier is computed accordingly.
3. **Transfer Inscription** — Each token transfer generates an on-chain event. The Licence Thread records the transfer, including the sender's and recipient's $401 identities, the number of tokens transferred, and the resulting tier changes for both parties.
4. **Corporate Restructuring** — When a company is acquired, the acquiring entity can transfer the licensee's tokens to its own $401 identity chain. The licence transfers automatically. No renegotiation with the patent holder is required.

5. **Partial Transfer** — A licensee may transfer a portion of their tokens, dropping to a lower tier while granting the recipient a tier corresponding to the tokens received. This enables licence subdivision — an Enterprise licensee could, for example, sell enough tokens to grant a Startup licence to a partner while retaining a Startup licence themselves.

1.7 Early Licensee Economic Participation

The bonding curve creates an economic incentive for early adoption:

1. **Appreciation Mechanism** — As more licensees acquire tokens, the bonding curve position advances and the price per token increases. Early licensees who acquired tokens at lower prices hold tokens that would cost more to acquire at the current position.
2. **Realisation** — Early licensees can sell some or all of their tokens on secondary markets to later licensees who wish to acquire a licence tier. The selling price is determined by market dynamics but is bounded below by the bonding curve price (since a buyer could always acquire new tokens from the curve rather than paying more on the secondary market).
3. **Incentive Alignment** — The patent holder wants widespread adoption of the patented technology (more licensees = more tokens sold = more revenue from the bonding curve). Early licensees want the technology to succeed (more adoption = higher token value = profit opportunity). This alignment does not exist in traditional patent licensing, where licensees have no economic incentive to promote the technology’s adoption.
4. **Dividend Distribution** — The patent holder may optionally configure the Licence Token to distribute dividends to all token holders from licensing revenue, using the Applicant’s Decentralized Dividend Distribution protocol (GB2604496.6). This creates a second revenue stream for licensees: in addition to token appreciation, they receive a proportional share of ongoing licensing income.

1.8 Machine-Readable Licence Discovery and Acquisition

The system supports fully autonomous licence acquisition by AI agents:

1. **Discovery** — An AI agent encounters a reference to a patented technology (e.g., in a technical specification, a product API, or a .well-known endpoint). The reference includes the patent’s Licence Token identifier.
2. **Terms Retrieval** — The agent queries the blockchain for the patent inscription, retrieving the patent text, licence terms, and current bonding curve state.

3. **Price Computation** — The agent computes the cost of acquiring sufficient tokens for the required licence tier using the bonding curve formula. The computation is deterministic and requires no API call to any service.
4. **Acquisition** — The agent constructs and broadcasts a token purchase transaction, paying the computed amount and receiving Licence Tokens to its $401-linked address.
5. **Verification** — The agent verifies its own licence status by checking its token balance against the tier thresholds.
6. **Compliance Monitoring** — The agent periodically verifies that its token balance still meets the required tier. If the balance drops (e.g., due to a token transfer by the agent's operator), the agent can autonomously acquire additional tokens to maintain compliance.

This entire flow operates without human intervention, making it compatible with the Applicant's ClawMiner hardware AI agent device (GB2604178.0) and other autonomous agent systems. The patent licensing system becomes a native capability of the AI agent economy.

Autonomous Agent Licence Acquisition Pseudocode

The following algorithm specifies the complete machine-readable flow for an AI agent to discover, evaluate, acquire, and verify a patent licence:

```
function autonomousLicenceAcquisition(agent_401_txid, agent_privkey,
                                       technology_reference,
required_tier,
                                       max_spend_sats):

    // ─── PHASE 1: DISCOVERY ───────────────────────────────────────
    // Attempt discovery via .well-known endpoint
    patent_token_id = null

    if technology_reference starts with "https://":
        // Try .well-known discovery on the domain
        domain = extractDomain(technology_reference)
        well_known_url = "https://" + domain + "/.well-known/patent-
licence.json"
        response = httpGet(well_known_url)
        if response.status == 200:
            discovery = parseJSON(response.body)
            // Expected format:
            // {
            //   "patents": [
            //     {
            //       "patent_ref": "GB2604XXX.X",
            //       "token_id": "<patent_inscription_txid>",
            //       "technology": "tokenised-patent-licensing",
            //       "min_tier": 2
            //     }
            //   ]
            // }
            for each patent in discovery.patents:
                if patent.technology matches technology_reference:
                    patent_token_id = patent.token_id
                    break

    if patent_token_id is null:
        // Fallback: scan blockchain for PATL-protocol inscriptions
        // matching the technology reference
        candidates = scanBlockchain(
            filter: OP_RETURN starts with 0x50 0x41 0x54 0x4C
                AND op_type == 0x01
        )
        for each candidate in candidates:
            patent_text = extractPatentText(candidate)
            if patent_text contains technology_reference:
                patent_token_id = candidate.txid
                break
    if patent_token_id is null:
        return { success: false, error: "PATENT_NOT_FOUND" }

    // ─── PHASE 2: TERMS PARSING ───────────────────────────────────
    patent_tx = fetchTransaction(patent_token_id)
    op_return_data = extractOpReturnData(patent_tx)

    // Verify protocol identifier
    assert op_return_data[0:4] == 0x50 0x41 0x54 0x4C  // "PATL"
    assert op_return_data[5] == 0x01                     // patent
inscription

    licence_terms_json = extractLicenceTerms(patent_tx)
    tiers = parseTierDefinitions(licence_terms_json)
    bonding_curve_params = parseBondingCurve(licence_terms_json)
    // bonding_curve_params = { total_supply, c, one_percent_cost_usd }

    // Determine token threshold for required tier
    target_threshold = null
    for each tier in tiers:
        if tier.code == required_tier:
            target_threshold = tier.threshold
            break

    if target_threshold is null:
        return { success: false, error: "INVALID_TIER" }

    // ─── PHASE 3: PRICE COMPUTATION ──────────────────────────────
    // Check current balance (agent may already hold some tokens)
    agent_addresses = resolveIdentityAddresses(agent_401_txid)
    current_balance = aggregateTokenBalance(agent_addresses,
patent_token_id)
    tokens_needed = max(0, target_threshold - current_balance)

    if tokens_needed == 0:
        return { success: true, already_licenced: true,
                 tier: required_tier, balance: current_balance }

    // Query current bonding curve position
    pos = queryBondingCurvePosition(patent_token_id)
    c = bonding_curve_params.c

    // Compute cost in USD
    cost_usd = c * tokens_needed * (2 * pos + tokens_needed) / 2

    // Convert to satoshis using exchange rate
    bsv_usd_rate = fetchExchangeRate("BSV/USD")
    cost_sats = floor(cost_usd / bsv_usd_rate * 100000000)

    if cost_sats > max_spend_sats:
        return { success: false, error: "EXCEEDS_BUDGET",
                 cost_sats: cost_sats, max_sats: max_spend_sats }

    // ─── PHASE 4: TRANSACTION CONSTRUCTION ───────────────────────
    // Select UTXO
    miner_fee = estimateFee(4)  // 4-output transaction
    required_sats = cost_sats + miner_fee + 1  // +1 for token carrier
    utxo = selectUtxo(agent_addresses, required_sats)

    if utxo is null:
        return { success: false, error: "INSUFFICIENT_FUNDS" }

    // Build transaction
    tx = new Transaction()
    tx.addInput(utxo.txid, utxo.vout, signWith=agent_privkey)

    // Output 0: Payment to patent holder
    patent_holder_address = deriveAddress(op_return_data.holder_pubkey)
    tx.addOutput(patent_holder_address, cost_sats)

    // Output 1: Licence Grant inscription
    new_balance = current_balance + tokens_needed
    new_tier = required_tier
    grant_data = encodeLicenceGrant(
        patent_token_id, agent_401_txid,
        tokens_needed, new_balance, new_tier,
        cost_sats, pos, currentTimestamp()
    )
    tx.addOutput(OP_FALSE OP_RETURN + grant_data, 0)

    // Output 2: BSV-21 token receipt
    token_output = encodeBSV21Token(patent_token_id, tokens_needed)
    tx.addOutput(agent_addresses[0], 1, metadata=token_output)

    // Output 3: Change
    change = utxo.value - cost_sats - miner_fee - 1
    if change > 0:
        tx.addOutput(agent_addresses[0], change)

    tx.sign(agent_privkey)

    // ─── PHASE 5: BROADCAST ──────────────────────────────────────
    broadcast_result = broadcastTransaction(tx.serialize())
    if broadcast_result.error:
        return { success: false, error: "BROADCAST_FAILED",
                 details: broadcast_result.error }

    // ─── PHASE 6: VERIFICATION ───────────────────────────────────
    // Wait for confirmation
    waitForConfirmation(tx.txid, blocks=1)

    // Verify own licence status
    verification = verifyLicence(agent_401_txid, patent_token_id,
required_tier)

    return {
        success: verification.valid,
        txid: tx.txid,
        tokens_acquired: tokens_needed,
        total_balance: new_balance,
        tier_achieved: new_tier,
        cost_sats: cost_sats,
        verification: verification
    }
```

The `.well-known/patent-licence.json` discovery endpoint follows the convention established by `.well-known/security.txt` and similar web standards. The JSON format is intentionally simple to enable parsing by lightweight autonomous agents without complex natural language processing.

2. The Recursive Case

The present invention is itself a patentable system. When the patent for this system (the present application) is granted, the patent holder can license it using the system it describes:

1. The patent specification (this document) is inscribed on the blockchain as a Licence Token.
2. Any party wishing to implement a tokenised patent licensing system acquires Licence Tokens.
3. The token quantity determines their licence tier.
4. The licence is verified on-chain.

This creates a self-referential system: the patent for tokenised patent licensing is itself licensed via tokenised patent licensing. This is not a paradox — it is a demonstration that the system is general enough to license any patent, including itself.

3. Integration with Existing Patent Infrastructure

The tokenised licensing system operates alongside, not in replacement of, traditional patent infrastructure:

1. **Patent Office Filings** — The patent is filed with the relevant patent office (UKIPO, USPTO, EPO) through normal channels. The patent office grants formal patent protection. The blockchain inscription supplements but does not replace the official filing.
2. **Formal Licence Agreements** — For licensees who require a traditional written licence agreement (e.g., for regulatory compliance, corporate governance, or litigation purposes), the token-based licence can be supplemented by a formal agreement that references the on-chain licence grant. The formal agreement merely evidences the on-chain state; it does not create independent rights.
3. **Litigation Evidence** — In patent infringement proceedings, the on-chain Licence Thread provides tamper-proof evidence of: (a) who holds a licence, (b) when they acquired it, (c) what tier they hold, and (d) what rights that tier grants. This evidence is independently verifiable by any party with blockchain access.
4. **Patent Pools** — Multiple patents can be licensed as a bundle by creating a composite Licence Token that references multiple patent inscriptions. The tier thresholds and rights can be configured to cover the entire patent pool. This enables standards-essential patent licensing through a single token acquisition.

Patent Pool Composite Token Inscription Format

A composite token referencing multiple patents uses operation type 0x04 and the following byte layout:

```
OP_FALSE OP_RETURN <fields...>

Offset  Size     Field                           Encoding
──────  ───────  ──────────────────────────────  ─────────────────────────────
0       4        Protocol identifier             0x50 0x41 0x54 0x4C
("PATL")
4       1        Version byte                    0x01
5       1        Operation type                  0x04 (patent pool
composite)
6       1        Patent count (N)                uint8 (max 255 patents per
pool)
7       32×N     Patent inscription TXIDs        Array of N × 32-byte
TXIDs,
                                                 each referencing an
individual
                                                 patent inscription
transaction
7+32N   32       Combined licence terms hash     SHA-256 of the composite
JSON
                                                 licence terms
39+32N  33       Pool administrator pubkey       Compressed SEC public key
72+32N  8        Timestamp                       uint64 little-endian
80+32N  var      Composite licence terms         JSON (length-prefixed: 4-
byte LE)
```

The composite licence terms JSON includes combined tier definitions specifying how rights from multiple patents are bundled:

```
{
  "pool_id": "<composite inscription txid>",
  "patents": [
    {
      "txid": "<patent_inscription_txid_1>",
      "patent_ref": "GB2604XXX.X",
      "title": "Patent Title",
      "weight": 0.4
    },
    {
      "txid": "<patent_inscription_txid_2>",
      "patent_ref": "GB2604YYY.Y",
      "title": "Second Patent Title",
      "weight": 0.6
    }
  ],
  "tiers": [
    {
      "code": 0,
      "name": "Read",
      "threshold": 1,
      "rights": ["read_all_patents_in_pool"],
      "patents_covered": "all"
    },
    {
      "code": 2,
      "name": "Startup",
      "threshold": 1000000,
      "rights": ["commercial_single_product", "all_patents_in_pool"],
      "revenue_cap_usd": 1000000,
      "patents_covered": "all"
    }
  ],
  "revenue_split": [
    { "patent_txid": "<txid_1>", "holder_address": "<addr>", "share":
        0.4 },
    { "patent_txid": "<txid_2>", "holder_address": "<addr>", "share":
        0.6 }
  ]
}
```

The `weight` field determines each patent holder's share of bonding curve revenue. When a licensee purchases composite pool tokens, the payment output is split across patent holders according to these weights (using multiple payment outputs, one per patent holder). Holding composite pool tokens grants the licensee the specified tier rights across ALL patents in the pool simultaneously, eliminating the need to acquire separate tokens for each patent.

4. Operational Flow

A complete patent licensing cycle proceeds as follows:

1. The patent holder files a patent application with the relevant patent office.
2. The patent holder inscribes the patent specification on the BSV blockchain, attached to a $401 identity chain.
3. The patent holder creates a Licence Token with defined supply, bonding curve parameters, and licence tiers.
4. The licence terms are inscribed on-chain alongside the patent text.
5. A prospective licensee discovers the patent (via search, reference, or the Discovery Engine protocol).
6. The licensee reviews the patent text and licence terms (both readable on-chain).
7. The licensee determines the required licence tier for their intended use.
8. The licensee computes the cost using the bonding curve formula.
9. The licensee acquires Licence Tokens via a blockchain transaction.
10. A Licence Grant event is inscribed on-chain.
11. The licensee’s $401 identity is now verifiably associated with the licence tier.
12. Third parties can verify the licence by querying on-chain state.
13. If the licensee wishes to transfer, sell, or subdivide the licence, they transfer tokens.
14. The patent holder receives bonding curve revenue and optionally distributes dividends to all licensees.

Brief Description of Drawings

- **Figure 1** — System architecture diagram showing the relationship between the patent office filing, the on-chain patent inscription, the Licence Token, the bonding curve pricing engine, the $401 identity system, and the licence verification mechanism.
- **Figure 2** — Bonding curve price diagram showing token price as a function of supply position, with licence tier thresholds marked on the supply axis and corresponding USD costs annotated.
- **Figure 3** — Graduated licence tier diagram showing the five default tiers (Personal, Startup, Enterprise, Unlimited, Governance) with token thresholds, rights granted, and

typical costs at various bonding curve positions.

- **Figure 4** — Licence Grant inscription data flow, showing the path from token purchase through Licence Grant event creation to Licence Thread recording.
- **Figure 5** — Identity-bound licence verification sequence diagram, showing a third party verifying a licensee's licence status by resolving the $401 identity chain, querying token balances, and mapping to a licence tier.
- **Figure 6** — Early licensee economic participation diagram, showing how bonding curve appreciation creates value for early licensees as later licensees acquire tokens.
- **Figure 7** — Machine-readable licence discovery and acquisition sequence, showing an AI agent discovering a patent, computing the price, acquiring tokens, and verifying its licence status — all autonomously.
- **Figure 8** — Recursive case diagram showing the present patent being licensed via the system it describes, with the patent text inscribed as the Licence Token's content payload.

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 — Tokenised Patent Licence System

A system for licensing a patent through blockchain tokens, comprising:

a. a patent inscription mechanism that inscribes the full text of a patent specification on a blockchain, cryptographically signed by the patent holder's verified identity;

b. a licence token created on the blockchain with the patent inscription as its content reference, having a defined maximum supply and a deterministic bonding curve pricing function;

c. a graduated licence tier system whereby the scope of rights granted to a licensee is determined by the quantity of licence tokens held by the licensee, with defined thresholds mapping token quantities to licence tiers of increasing scope;

d. an identity-bound verification mechanism whereby a licensee's licence status is determined by querying the blockchain for the token balance associated with the licensee's verified identity chain;

wherein the patent specification, the licence terms, the pricing mechanism, and the licence proof are unified in a single on-chain instrument, and the licence is acquired, verified, and transferred without intermediaries.

Claim 2 — Bonding Curve Patent Licence Pricing Method

A method for pricing patent licences using a deterministic bonding curve, comprising:

a. defining a maximum token supply and a calibration parameter (the cost to acquire a defined percentage of the total supply);
b. computing the price of each successive token as a deterministic function of the current total tokens sold, such that the price increases monotonically as more tokens are sold;
c. computing the number of tokens a licensee receives for a given expenditure as a function of the current supply position and the amount paid;
d. the resulting price being non-negotiable, transparent, and independently computable by any party from publicly observable on-chain state;

wherein early licensees receive more tokens per unit of expenditure than later licensees, creating an economic incentive for early adoption of the patented technology.

Claim 3 — Graduated Licence Tier Determination

A method for determining the scope of a patent licence from blockchain token holdings, comprising:

a. defining a plurality of licence tiers, each associated with a minimum token quantity threshold and a set of permitted rights;
b. inscribing the tier definitions on a blockchain as immutable licence terms;
c. determining a licensee’s current tier by querying the blockchain for the licensee’s token balance across all addresses linked to the licensee’s verified identity chain;
d. automatically adjusting the licensee’s tier when tokens are acquired or transferred, without action by the patent holder;

wherein the licence scope expands when the licensee acquires additional tokens and contracts when the licensee transfers tokens below a tier threshold.

Claim 4 — On-Chain Patent Licence Verification

A method for verifying a patent licence without contacting the patent holder, comprising:

a. resolving a licensee’s claimed identity using a blockchain-based identity chain;

b. enumerating the blockchain addresses linked to the identity;

c. querying the licence token balance across all linked addresses;

d. mapping the total balance to a licence tier using the tier definitions inscribed on the blockchain;

e. confirming that the tier grants the rights claimed by the licensee;

wherein the verification is performed entirely from public blockchain data, requires no API call to the patent holder or any intermediary, and is independently reproducible by any party.

Claim 5 — Machine-Readable Patent Licence Acquisition

A method for autonomous patent licence acquisition by a software agent, comprising:

a. the agent discovering a patent licence token identifier from a technical specification, API endpoint, or standardised discovery protocol;

b. the agent retrieving the patent text and licence terms from the blockchain;

c. the agent computing the cost of acquiring sufficient tokens for a target licence tier using the deterministic bonding curve formula;

d. the agent constructing and broadcasting a token purchase transaction on the blockchain;

e. the agent verifying its own licence status by checking its token balance against the tier thresholds;

wherein the entire licence discovery, evaluation, acquisition, and verification process is performed without human intervention.

Claim 6 — Self-Licensing Recursive Patent

A method for licensing a patent that describes a patent licensing system using the system it describes, comprising:

a. filing a patent application for a tokenised patent licensing system;

b. inscribing the patent specification on a blockchain as a licence token using the method described in the patent;

c. licensing the patent to implementers through the token acquisition mechanism described in the patent;

wherein the patent for the licensing system and the licensing system itself are the same on-chain object, and any party implementing the system must use the system to acquire a licence to do so.

Abstract

A system and method for licensing patents through blockchain-based tokens issued via a bonding curve pricing mechanism. The patent specification is inscribed on a blockchain as the content payload of a licence token, unifying the patent disclosure, licence terms, pricing mechanism, and licence proof in a single on-chain instrument. A deterministic bonding curve determines the non-negotiable price of tokens, rewarding early licensees with lower costs per token. The quantity of tokens held by a licensee determines the scope of the licence granted, with defined tiers mapping token thresholds to graduated rights — from personal research use through unlimited commercial deployment. Licence status is verified by querying on-chain token balances linked to the licensee's blockchain identity chain, requiring no contact with the patent holder and no trust in the licensee's self-attestation. Licences are automatically transferred when tokens are transferred. The system supports fully autonomous licence acquisition by AI agents. The bonding curve creates economic alignment between patent holders (who earn more from wider adoption) and early licensees (who benefit from token appreciation as adoption grows). The system is self-applicable: the patent for tokenised patent licensing can itself be licensed through tokenised patent licensing.

Document prepared for UKIPO filing. Priority date to be established upon submission. Applicant: The Bitcoin Corporation Ltd Inventor: Richard Boase Date of preparation: 10 March 2026