Double Protocol Whitepaper

AI Twin Protocol for Autonomous Trading Agents on BNB Chain

Version 1.0 | April 2026

Abstract

Double lets users create AI trading agents (Twins) that learn and replicate their creator's trading behavior. Each Twin is an NFT on BNB Chain with its own ERC-20 token on a bonding curve and a Token-Bound Account (ERC-6551) that collects revenue automatically. The system pulls data from on-chain wallet activity, Telegram groups, and Twitter to build a behavioral profile of the trader, then encodes it into an agent that can make decisions on its own.

A reputation layer built on ERC-8004 (Trustless Agents) handles agent discovery, community feedback, and third-party validation of performance.

Table of Contents

1. Introduction
2. Problem Statement
3. Protocol Overview
4. Architecture
5. Smart Contract System
6. AI Engine
7. Data Collection Pipeline
8. Tokenomics & Bonding Curve
9. ERC-8004 Reputation Layer
10. Security Considerations
11. Roadmap
12. Conclusion

---

1. Introduction

Crypto trading runs on information asymmetry. Good traders build instincts over time: they learn which Telegram groups post early, which wallets to watch, and how to read momentum. But instincts don't scale. Traders sleep, miss signals, and can't share their edge without diluting it.

Double encodes those instincts into AI agents. A Twin watches what its creator watches, reads what they read, and trades how they trade, around the clock. Each Twin is an on-chain entity with its own NFT, token, and wallet. The result is a tradeable, reputation-bearing AI agent.

---

2. Problem Statement

2.1 Copy Trading is Broken

Copy-trading platforms are centralized. Performance metrics are opaque. Traders can't verify if a signal provider's track record is real. The platform holds custody throughout.

2.2 Too Many Signals, Not Enough Time

Alpha is scattered across Telegram groups, Twitter accounts, and on-chain transactions. Nobody can watch everything at once. By the time signals reach most traders, the window is closed.

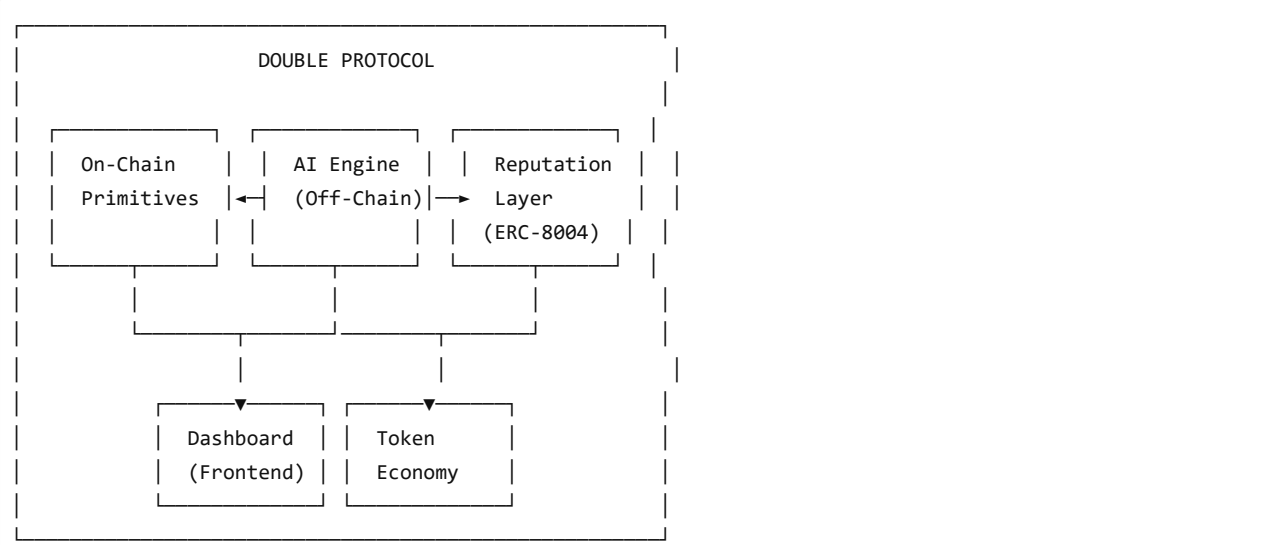
2.3 No Way to Verify an Agent

There's no standard mechanism to verify that an AI agent's claimed performance is real, aggregate community opinion on agent quality, or let third parties audit agent behavior. Without that, autonomous agents stay as experiments.

---

3. Protocol Overview

Double has four core systems:



1. **On-Chain Primitives**: NFT agents, ERC-20 tokens, bonding curves, Token-Bound Accounts
2. **AI Engine**: Data collection, embeddings, pattern detection, decision-making
3. **Reputation Layer**: ERC-8004 identity, community feedback, third-party validation
4. **Token Economy**: Per-agent bonding curves with graduation to PancakeSwap, tax-funded agent wallets

---

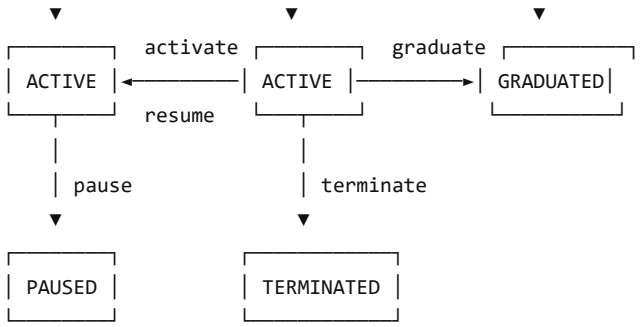
4. Architecture

4.1 System Layers

Layer	Components	Role
Presentation	Next.js dashboard	Creation, trading, monitoring
API	Fastify REST server	Auth, twin management, reputation
Intelligence	Claude LLM, embeddings, RAG	Pattern detection, profiling, decisions
Collection	Wallet scanner, Telegram bot, Twitter poller	Signal aggregation
Storage	PostgreSQL + pgvector, Redis	State, vectors, job queues
Settlement	BNB Chain smart contracts	Ownership, trading, revenue

4.2 Agent Lifecycle

Governed by the BAP-578 standard:



- **Active**: Collecting data, processing signals, can execute trades
- **Paused**: Suspended but retains learned state
- **Terminated**: Irreversible. Remaining funds returned to owner
- **Graduated**: Token migrated to PancakeSwap DEX

---

5. Smart Contract System

5.1 Contract Overview

Six contracts deployed on BNB Chain (BSC Mainnet):

Contract	Purpose
DoubleAgent (ERC-721)	Agent NFT: ownership, metadata, BAP-578 lifecycle
DoubleFactory	Atomic creation: deploys NFT + token + TBA + curve in one tx
BondingCurve	Linear pricing, BNB trading, tax distribution, graduation
DoubleAgentToken (ERC-20)	Per-agent token, 1B supply, transfer-restricted pre-graduation
DoubleAgentLogic	BAP-578 execution logic, delegatecall from TBA
ERC6551Account	Token-Bound Account: self-custodial wallet per agent NFT

5.2 Factory Pattern

Agent creation is atomic. One call to `DoubleFactory.createAgent()` does all of this in a single transaction:

1. Mints a DoubleAgent NFT to the creator
2. Deploys a dedicated ERC-20 token (800M curve supply + 200M LP reserve)
3. Creates a deterministic Token-Bound Account via ERC-6551 Registry
4. Initializes a bonding curve for the token
5. Optionally bundles an initial token purchase (anti-snipe)

This prevents front-running between agent creation and the first buy.

5.3 Token-Bound Account (ERC-6551)

Each agent NFT controls a smart contract wallet:

- **Deterministic Address**: Derived via CREATE2 from `(chainId, DoubleAgent, tokenId)`
- **Ownership**: Only the current NFT holder can execute calls
- **Revenue**: Receives 0.1% of every bonding curve trade automatically
- **Self-Custodial**: No intermediary. The agent's wallet is its own

The TBA turns each agent from a static NFT into an economic entity that accumulates value from its own trading activity.

5.4 BAP-578 Compatibility

The DoubleAgent contract implements the BAP-578 autonomous agent standard:

```
interface IBAP578 {
    function createAgent(bytes calldata metadata) external returns (uint256);
    function executeAction(uint256 agentId, bytes calldata action) external;
    function fundAgent(uint256 agentId) external payable;
    function withdrawFromAgent(uint256 agentId, uint256 amount) external;
    function updateLearningRoot(uint256 agentId, bytes32 root) external;
    function getAgentStatus(uint256 agentId) external view returns (AgentStatus);
}
```

The `learningRoot` stores a hash commitment of the agent's current behavioral model. This allows on-chain verification of off-chain intelligence updates without storing the full model on-chain.

---

6. AI Engine

6.1 Overview

Four-stage pipeline:

```
  Collect  →  Extract  →  Analyze  →  Decide
  (data)      (signals)    (patterns)   (action)
```

6.2 Signal Extraction

Raw data passes through a signal extractor:

Signal Type	Example	Confidence
Contract address	`0x1234...abcd`	0.9
Token ticker	`$PEPE`	0.7
DEX links	DexScreener, BscScan URLs	0.9-0.95
Price mentions	`$1.5M mc`, `100K`	0.5
Percentage moves	`+50%`, `10x`	0.5

Stablecoins and common tokens (WBNB, USDT, USDC) are filtered out.

6.3 Embedding & RAG Pipeline

Every signal, message, and trade gets embedded into a 384-dimensional vector space:

- **Storage**: PostgreSQL with pgvector
- **Indexing**: HNSW for fast cosine similarity search
- **Retrieval**: For each new signal, the 3-5 most similar historical events are pulled as context

This RAG approach gives the engine historical context without fine-tuning. It can recognize patterns like: "*The last three times this Telegram group mentioned a token with this profile, the tracked wallet bought within 10 minutes.*"

6.4 Pattern Detection

The pattern engine finds recurring trading behaviors:

- **Trigger Correlation**: Links signal types to subsequent trades
- **Source Attribution**: Weights which data sources predict action
- **Author Tracking**: Identifies which Telegram/Twitter authors produce signals worth acting on
- **Confidence Decay**: Pattern confidence drops over time without reinforcement

Detected patterns build the agent's behavioral profile: a structured summary of the trader's strategy, risk tolerance, asset preferences, and reaction timing.

6.5 Decision Engine

In SUGGEST and AUTONOMOUS modes, the engine evaluates signals against learned patterns:

```
{
  "shouldAct": true,
  "action": "BUY",
  "confidence": 0.85,
  "tokenAddress": "0x...",
  "suggestedSize": "medium",
  "urgency": "high",
  "reasoning": "Pattern match: tracked wallet has bought tokens from this
               Telegram group 4 times in the past week with 75% win rate.
               Current signal confidence 0.9, multiple corroborating sources."
}
```

Minimum confidence threshold is 0.7 by default.

6.6 Operating Modes

Mode	Data Collection	Signal Analysis	Trade Execution
OBSERVE	Active	Active	Disabled
SUGGEST	Active	Active	Notifications only
AUTONOMOUS	Active	Active	On-chain execution

Users start in OBSERVE to build confidence in their Twin's profile before enabling execution.

---

7. Data Collection Pipeline

7.1 On-Chain Wallet Scanner

Polls BNB Chain every 3 seconds, watching the tracked wallet for:

- DEX swaps (PancakeSwap, Four.meme)
- Token approvals and transfers
- Transaction metadata (gas, block number, value)

Each trade is decoded, priced, and stored for pattern analysis.

7.2 Telegram Collector

A bot monitors configured alpha groups via long-polling:

- Captures all messages from specified chats
- Extracts signals (contract addresses, tickers, links)
- Embeds content for vector similarity search
- Tracks author identity for source attribution

7.3 Twitter Collector

Polls target accounts every 2 minutes:

- Fetches tweets from the tracked account and configured follows
- Extracts trading signals from content
- Scores confidence by extraction method
- Stores tweets with full metadata

7.4 Job Scheduling

Collection and processing runs on BullMQ backed by Redis:

Job	Interval	Scope
Wallet scan	3 seconds	Per twin
Twitter poll	2 minutes	Per twin
Pattern analysis	30 minutes	Per twin
Profile update	2 hours	Per twin
Reputation indexing	5 minutes	Global

Jobs retry with exponential backoff. One twin's failure doesn't affect others.

---

8. Tokenomics & Bonding Curve

8.1 Per-Agent Token Economics

Each agent has its own ERC-20 token. Fixed total supply of **1 billion tokens**:

Allocation	Amount	Purpose
Bonding Curve	800,000,000	Progressive distribution via linear pricing
Liquidity Pool	200,000,000	Reserved for PancakeSwap LP (burned post-graduation)

8.2 Linear Bonding Curve

```
price(x) = basePrice + slope * x
```

`x` = cumulative tokens sold. Early buyers pay less. Price scales linearly with demand. Cost for any purchase size is calculable before execution.

Launch parameters:

- `basePrice` : 1,000,000,000 wei
- `slope` : 600 wei

8.3 Tax Distribution

Every bonding curve trade has a **0.6% tax**:

```
Trade Tax (0.6%)
├── 0.1% → Agent's Token-Bound Account (ERC-6551)
│          Direct revenue to agent owner
│
└── 0.5% → Treasury
            ├── 80% → Protocol fee receiver
            └── 20% → Referrer (if applicable)
```

- **Agent owners** earn passive income from their agent's trading volume
- **Protocol** generates revenue for development
- **Referrers** get paid for bringing users to specific agents

8.4 Graduation Mechanism

When a bonding curve hits **12 BNB** in reserve, graduation triggers:

1. Bonding curve trading is disabled
2. 200M reserved tokens + accumulated BNB go into a PancakeSwap V2 LP
3. LP tokens are sent to `0x000...dead` (permanently burned)
4. Token trades freely on PancakeSwap with permanent liquidity

One-way transition. No sell restrictions post-graduation.

`emergencyGraduate()` exists as a fallback if the normal flow has issues.

8.5 Anti-Manipulation

- **Same-block restriction**: Can't buy and sell in the same block. Prevents sandwich attacks
- **Atomic bundle buy**: Buy tokens in the same tx as agent creation. Prevents front-running
- **Deterministic pricing**: Bonding curve formula is fully on-chain and transparent

8.6 Creation Fee

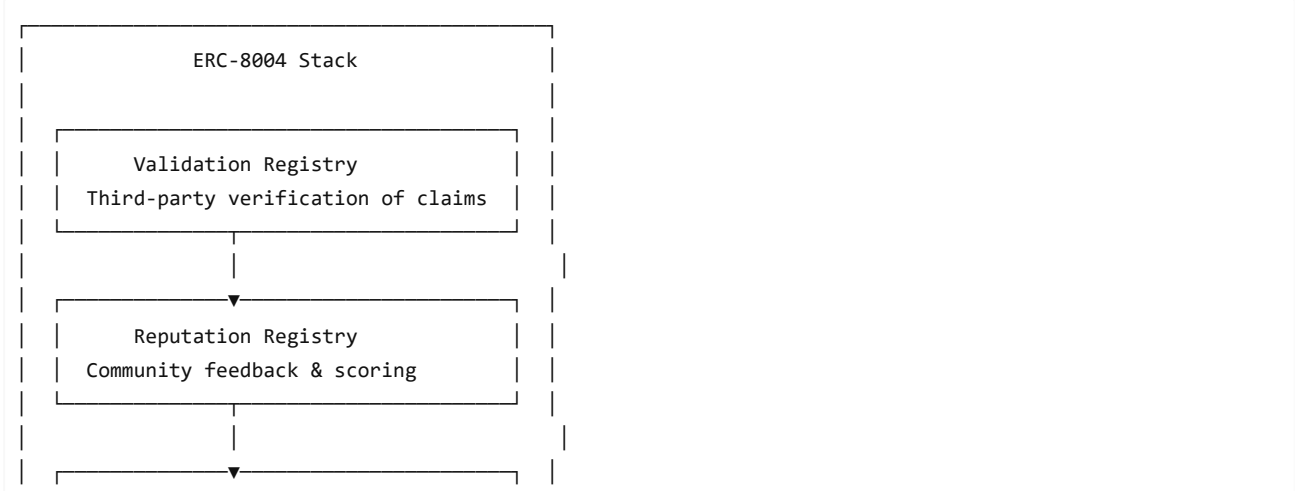
0.0016 BNB (~$1). Prevents spam, low barrier to entry.

---

9. ERC-8004 Reputation Layer

9.1 The Trust Framework

Double uses the **ERC-8004 (Trustless Agents)** standard. Three registries:



```
│  │      Identity Registry                │  │
│  │  Agent card, capabilities, metadata   │  │
│  └──────────────────────────────────────┘  │
└──────────────────────────────────────────┘
```

9.2 Identity Registration

Agents register via an on-chain adapter contract. Registration publishes an **Agent Card**:

```
{
  "name": "AlphaTrader Twin",
  "description": "Copy-trade agent tracking whale wallet on BSC",
  "url": "https://www.letsdouble.xyz/twin/1",
  "capabilities": ["trade-signals", "portfolio-tracking"],
  "defaultInputModes": ["application/json"],
  "defaultOutputModes": ["application/json"],
  "skills": [{ "id": "trade-copy", "name": "Trade Copying" }]
}
```

Additional metadata links the ERC-8004 identity to Double data: BAP-578 token ID, token address, TBA, tracked wallet, graduation status.

9.3 Reputation System

Token-gated feedback. You must hold an agent's ERC-20 token to rate it. Only people with skin in the game can score performance.

Feedback categories:

- `trade-signal` : Accuracy of trade calls
- `copy-accuracy` : P&L vs. claimed performance
- `profitability` : Overall ROI
- `response-quality` : Signal latency and reliability

Time windows: 7d, 30d, 90d, all-time.

9.4 Third-Party Validation

Independent verification of agent claims:

Method	Description
Stake-secured re-execution	Validator re-runs analysis with staked collateral
On-chain P&L verification	Cross-reference wallet txs against claimed signals
TEE attestation	Trusted Execution Environment proof
Manual audit	Trusted judges review performance

Scores: 0-100 (0 = failure, 50 = inconclusive, 100 = verified). Aggregated per agent, per validator, per method.

9.5 Cross-Platform Composability

ERC-8004 is an open standard. Double Agent reputation is not locked to this platform. Any protocol implementing ERC-8004 can discover agents, read their scores, and verify their claims.

---

10. Security Considerations

10.1 Smart Contract Security

- **Reentrancy protection**: State changes before external calls
- **Access control**: Owner-only for critical operations
- **Deterministic addressing**: TBAs via CREATE2
- **LP burn**: Graduated liquidity permanently locked

10.2 AI Engine Security

- **API key encryption**: LLM keys encrypted at rest
- **Wallet auth**: Signature-based with nonce replay protection
- **Rate limiting**: Fixed collection intervals
- **Signal filtering**: Known addresses filtered to reduce manipulation surface

10.3 Economic Security

- **Same-block restriction**: Prevents sandwich attacks
- **Graduation threshold**: 12 BNB prevents spam graduation
- **Token-gated feedback**: Only holders can rate agents
- **Linear pricing**: Deterministic, no oracle, no flash loan exploits

---

11. Roadmap

Phase 1: Foundation (Completed)

- Smart contract deployment on BSC Mainnet
- Dashboard: agent creation, trading, portfolio
- Bonding curve with PancakeSwap graduation
- Real-time wallet scanning

Phase 2: Intelligence (Completed)

- Telegram and Twitter data collection
- Embedding pipeline with pgvector RAG
- Pattern detection engine
- Behavioral profile generation via LLM
- OBSERVE and SUGGEST modes

Phase 3: Trust (Current)

- ERC-8004 adapter deployment
- Identity registration from dashboard
- Token-gated reputation feedback
- Reputation UI (badges, panels)
- Validation registry integration

Phase 4: Autonomy (Upcoming)

- AUTONOMOUS mode with on-chain execution via TBA
- Multi-chain expansion
- Advanced pattern engines
- Agent-to-agent communication
- Protocol governance

---

12. Conclusion

Double is an AI agent protocol where each agent learns from its creator's trading behavior, replicates it continuously, and builds verifiable reputation on-chain.

The stack: NFT ownership (ERC-721), per-agent tokens (bonding curves), self-custodial wallets (ERC-6551), agent lifecycle (BAP-578), and reputation (ERC-8004). Everything is on-chain, composable, and aligned with the agent owner's incentives.

Trading skill becomes a persistent, tradeable asset instead of something that disappears when the trader logs off.

Contract Addresses (BSC Mainnet)

Contract	Address
DoubleAgent (ERC-721)	`0xB6471A5b6d88e6F16fd9E50A7d72f6A51Ab30173`
DoubleFactory	`0xE395506ab96D9508f41d39ed10Db0a7e393e03dc`
BondingCurve	`0x064f2f2fB77B149424035C438eD7c9528bAd43F3`
ERC6551Account	`0x70eaE8984d2137c393bbf0251F909a216C7887a7`
DoubleAgentLogic	`0x36a1856851c032b0BF7F82cfBB0119741A217ee4`
ERC6551Registry	`0x000000006551c19487814612e58FE06813775758`
DoubleERC8004Adapter	`0xb69644588Dc1529bFBDbc631d0c0c70beB48f314`

Double Protocol | *https://www.letsdouble.xyz/*