BlockTEN (BTEN) — Proof-of-Tenure Token on BSC (Whitepaper)

Executive Summary

BlockTEN (BTEN) is a BEP-20 token on Binance Smart Chain (BSC) that implements a "Bitcoin in miniature" concept, combining Bitcoin's fixed supply and halving schedule with a novel Proof-of-Tenure (PoTen) mining mechanism. The token has a capped supply of 21,000,000 (21 million) with 8 decimal places, mirroring Bitcoin's supply and divisibility. However, unlike Bitcoin's proof-of-work, BTEN's mining does not require energy-intensive hashing; instead, holders earn new tokens based on how long and how much they hold BTEN, turning ownership × time into mining chance. This approach fosters fairness by allowing small holders to compete through patience while large holders gain by frequent attempts, striking a balance in opportunity.

BTEN's issuance follows a Bitcoin-like halving schedule. The mining reward starts at 50 BTEN per win and halves every 210,000 **wins** (successful mining events, analogous in count to Bitcoin's block interval). An automatic difficulty adjustment algorithm targets a network-wide "win" roughly every ~10 seconds. As a result, the entire emission (from 50 BTEN rewards down to zero) plays out over a much shorter period than Bitcoin's (on the order of a few years rather than a century), while preserving a scarcity-driven tokenomics model. The difficulty adjusts after each win to maintain the ~10-second target interval between wins, scaling up if wins happen too quickly and scaling down if too slowly. The smart contract is immutable and role-free: there are no special admin controls, no taxes or fees on transfers, and no upgradability, ensuring the system operates transparently according to code without privileged interference.

BTEN was launched in a community-centric manner. There was no presale, no ICO, and no team allocation of tokens. Instead, a small genesis supply (50 BTEN) was minted to the deployer for seeding initial liquidity and bootstrapping the mining process. Early distribution was conducted via free airdrops, and ongoing supply is distributed entirely through on-chain mining that anyone can participate in. The airdrops were executed in a push-only fashion – tokens were delivered directly to users' wallets to eliminate phishing risks, and the official website never asks users to connect wallets or claim tokens. All mining events and token issuance occur on-chain and are logged via events, allowing the community to audit the process in real time. In summary, BTEN offers a fast, fair, and transparent token economy: a limited-supply, self-regulating mining token that rewards patience and participation rather than expensive hardware or large initial investments.

Introduction and Motivation

Cryptocurrencies like Bitcoin pioneered the concept of digital scarcity through proof-of-work mining, but at the cost of high energy usage and entry barriers for average users. BlockTEN (BTEN) is designed to capture the essence of Bitcoin's economic model – fairness, fixed supply, halving-driven scarcity – in a more accessible and efficient form. BTEN is described as "Bitcoin in miniature on BSC," retaining the 21 million maximum supply and halving schedule concept of Bitcoin, yet compressing the halving interval to every 210,000 **wins** (instead of Bitcoin's ~4-year, 210,000-block cycle) to engage the community over a shorter period. By leveraging BSC's high throughput and low fees, BTEN enables frequent on-chain

participation (targeting a win every ~10 seconds) without prohibitive cost, making the mining experience swift and user-friendly.

The key motivation behind BTEN is fair token distribution and simplicity. The project launched with no fundraising rounds or private allocations – there was no presale, no ICO, and no reserved team tokens. Instead, the initial supply beyond a tiny genesis mint was given out freely via airdrops to the community, and all further issuance is earned by users through the mining process. This ensures a community-driven network from the outset, avoiding the centralization that can occur when insiders or investors hold large pre-mined stakes. Additionally, BTEN's PoTen mechanism was conceived to lower barriers to entry: anyone with a token balance can mine new tokens by simply holding and transacting on-chain, without specialized hardware or large capital lockup. This proof-of-tenure model inherently favors neither wealthy participants nor active traders unfairly – a small holder can win by being patient, while a large holder can win by trying more frequently, and both strategies have proportional influence. The design aims to encourage an engaged community of holders who are incentivized to HODL and support the network long-term, aligning with the ethos of a fair launch and decentralized ownership.

In summary, BTEN's introduction of Proof-of-Tenure mining on BSC is motivated by the pursuit of a more inclusive and energy-efficient mining model that still preserves the fundamental scarcity and fairness principles of Bitcoin. By replacing brute-force computation with a time-and-balance based lottery, BTEN transforms time itself into a mining resource, offering a novel experiment in token distribution that could broaden participation in decentralized economies.

Token Architecture and Parameters

Supply and Decimals

BTEN has a fixed maximum supply of 21,000,000 tokens, enforced at the smart contract level. Each token is divisible into 10^8 smaller units (8 decimal places), the same precision as Bitcoin. This means 1 BTEN = 100,000,000 base units, often referred to as "satoshis" by analogy. The choice of 21 million as the cap and 8 decimals is intentional to mirror Bitcoin's tokenomics, instilling a familiar sense of scarcity and granularity.

Genesis Mint

At deployment, the contract performed a one-time genesis mint of 50 BTEN to the deployer's address. This tiny initial supply (~0.000238% of the total cap) was used strictly for bootstrapping purposes – for example, to provide initial liquidity and to ensure that the first mining attempts had some token balance to operate with. No other premine or allocation was made; aside from this 50 BTEN, 100% of the remaining supply must be mined or distributed via airdrops to the community. The genesis tokens do not confer any lasting advantage to the team, as they were subject to the same mining rules and primarily served to "jump-start" the system.

Emission Logic

New BTEN enters circulation through the on-chain mining mechanism (detailed in the next sections) at a controlled pace. BTEN's emission follows a halving schedule analogous to Bitcoin's mining rewards, but uses **win-based epochs** rather than chain block height. The initial mining reward is 50 BTEN per win (matching Bitcoin's initial 50 BTC block reward). This reward amount is not constant; it halves periodically so that over time the emission rate decreases and approaches zero as the cap is reached. In BTEN, the halving progression is determined by the cumulative count of wins (successful mining events)

since launch. The interval HALVING_WINS = 210,000 defines one epoch. The current epoch number is calculated as **floor((totalWins + 1) / 210,000)**, where *totalWins* is the number of Win events since launch. In other words, each epoch spans 210,000 wins. For example, during epoch 0 (from launch until the first 210,000 wins) each win yields 50 BTEN; in epoch 1 (the next 210,000 wins) each win yields 25 BTEN; epoch 2 yields 12.5 BTEN; and so on. This halving sequence continues until the reward would drop below the smallest unit or the 21 million cap is reached – at that point, the contract sets any further reward to 0, and mining effectively stops once all BTEN are minted. This ensures the hard cap is never exceeded.

Halving Cycle: The choice of a 210,000-win epoch (mirroring Bitcoin's 210,000-block count per halving) means BTEN's halving schedule is dramatically accelerated in real time compared to Bitcoin's ~4-year cycle. *(At the target rate of ~10 seconds per win, 210,000 wins would occur in roughly 24 days.)* This compressed schedule was chosen to encourage early participation and to allow the full mining distribution to play out over a much shorter horizon. Despite the quick cycle, the quantitative progression of rewards (50 → 25 → 12.5 → …) and the total supply limit remain identical in proportion to Bitcoin's, yielding a familiar diminishing emission curve that is simply compressed in time.

Token Standard and Transfers

BTEN is implemented as a standard BEP-20 token (BSC's equivalent of ERC-20). It supports the usual interfaces: transferable balances, allowances for delegated spending, and the conventional Transfer and Approval events on-chain. There are no built-in taxes or fees on BTEN transfers – sending BTEN from one address to another only costs the normal BSC network gas fee, with the BTEN contract itself not charging or burning any amount. This was a deliberate design choice to keep the token behavior simple and "pure" (no deflationary tricks or complex fee redistribution), in line with the simplicity principle. The token's decimals = 8 setting is slightly unusual in the BSC/ETH ecosystem (which commonly uses 18 decimals), but it poses no integration issues and simply reflects the Bitcoin-like design. Exchanges, wallets, and explorers handle 8-decimal tokens seamlessly, and this parameter was verified on testnet/mainnet for compatibility.

In summary, BTEN's architecture is straightforward: a capped, halving-emission token with standard BEP-20 functionality. All critical parameters – the cap (21 million), decimals (8), initial reward (50), halving interval (210,000 wins), target interval (~10 seconds) – are encoded as constants in the smart contract. This immutability means the economic rules cannot be changed post-deployment, cementing the commitment to Bitcoin-like tokenomics and fairness.

PoTen Mining Model (Proof-of-Tenure)

BTEN introduces Proof of Tenure (PoTen) as its mining model, which fundamentally differs from both proof-of-work and proof-of-stake. The core idea of PoTen can be stated succinctly as: **Ownership × Time = Chance.** In other words, every address holding BTEN accumulates a mining potential over time proportional to its balance and the duration of holding. Instead of contributing hash power or locking tokens, a user's "stake" in mining is literally the act of HODLing: the longer an address holds a given balance of BTEN without interruption, the greater its probability of minting new BTEN when it initiates a mining attempt.

Holding Age and Resets

Crucial to the PoTen mechanism is the concept of **holding age** (also called tenure or simply *age*). Every address has a timer that tracks how long it has continuously held its current balance. This age is

measured from the last time the address either sent or received a BTEN transfer (above 0). Any external transfer, whether sending tokens out or receiving tokens from someone else, will reset the address's age to zero, because the continuous holding period is considered broken. Similarly, whenever an address performs a mining attempt, it will reset its own age immediately after the attempt. The rationale is that an attempt "uses up" the accrued time potential. (Even a self-transfer of 0 BTEN, which is allowed and counts as a mining attempt, will reset the age despite not moving any balance.) The only events that do not reset an address's age are actions that do not involve that address's balance changing or attempting – for example, someone else's transfers have no effect on your address, and purely reading data from the contract doesn't affect age.

In effect, an address's age represents the continuous idle time since its last activity involving BTEN. These rules ensure no one can pre-load "time" unfairly across multiple addresses. If a user tries to split their tokens among many addresses to farm age, the moment they consolidate those tokens by sending to one address, the recipient address's age resets (because an incoming transfer occurred). Likewise, one cannot load up an address with time first (by not attempting) and then fund it with tokens at the last second – that incoming funding resets the age to zero, nullifying the accumulated time. This design keeps the mining field level: every address must decide between holding or moving tokens, and any movement clears the slate in terms of mining potential. It aligns incentives such that miners are encouraged to "sit tight" and avoid unnecessary transfers. In practical terms, an aspiring BTEN miner will want to maintain a static balance on their address for as long as possible to build up age, and when ready to attempt mining, they will trigger an attempt (which by definition uses that same address, not involving any outside party).

Mining Attempts

A mining attempt in PoTen is the action that converts an address's accumulated holding age into a chance of winning new BTEN. The BTEN contract provides two equivalent ways to attempt mining: (1) calling the dedicated function `attempt()` on the contract, or (2) sending a self-transfer (a transaction where the from and to address are the same) with any amount of BTEN (including 0). Both methods achieve the same result: they invoke the internal mining logic for the sender's address. The self-transfer method is particularly user-friendly, as it means users can mine using a standard wallet by simply sending their BTEN to themselves (which most wallets allow), without needing a special interface. Even a 0-value self-transfer is considered a valid attempt and costs only gas.

When an attempt is made, the contract immediately calculates the address's win probability based on its current balance (B) and effective holding time (Δt_eff) since the last reset. Here Δt_eff is essentially the address's age that has not yet been "used" in a previous attempt – effectively, the time since the later of (a) the last reset or (b) the last attempt. If it's the first ever attempt for that address, Δt_eff is the entire holding time since the address acquired its BTEN. If the address attempted once before, then only the time since the previous attempt counts as new potential (since the prior attempt would have reset the age).

The mining potential P is defined as $P = B \times \Delta t\_eff$ (balance times effective time, measured in "token-seconds"). This potential is then compared against a global difficulty parameter (denoted D). The contract draws a pseudo-random number for the attempt and declares the attempt a win if the random number falls below the potential (scaled against difficulty). More concretely:

- The contract computes a random hash h for the attempt.
- The attempt succeeds (a win) if `h mod D < P`, where D is the current difficulty parameter. *(Note: P is capped at a maximum of D; if an address's potential exceeds D, it effectively has a 100% chance for that attempt, and any potential beyond* D *is wasted. No address can have more than a*

100% chance on a single attempt, which provides a natural diminishing return for extremely large holders or extremely long waits.)

This condition is equivalent to saying that the probability of success on that attempt is **p = min(P, D) / D**. If the address has very little balance or has not waited long, P will be small relative to D, yielding a low chance of success. If the address has a large B×T product, P can approach or reach D, giving a significant chance (up to 100% if P ≥ D). Importantly, any excess potential beyond D doesn't increase the success probability further, enforcing a practical limit on advantage.

After the win probability is evaluated, regardless of win or loss, the contract performs some housekeeping: it resets the attempting address's age to zero (since an attempt was made) and it updates the address's attempt count (an internal counter used to ensure randomness uniqueness for each try). If the attempt was a win, the contract immediately mints the reward to the address and emits a `Win` event; if it was a loss, no mint occurs. Either way, the attempt is recorded by emitting an `Attempt` event with details of the outcome. The result is that an address must now begin accumulating age afresh if it wants to try again. This creates a "mining loop" for each participant: **accumulate potential over time by holding → attempt → reset → accumulate again → attempt again**, and so on. Each user can choose how long to wait between attempts to optimize their success odds versus attempt frequency. Those who wait longer have a higher chance per attempt, whereas those who attempt more frequently have a lower chance each time but more "draws" at the lottery.

Win and No-Loss Property

If a user "wins" an attempt, the contract mints the current epoch's reward directly into their balance. This is a net gain of BTEN for the user, analogous to a miner finding a block reward in Bitcoin. If the user does not win, importantly, they lose nothing except the small gas fee for the transaction. The user's BTEN balance remains intact – unlike some lotteries or staking mechanisms, there is **no token expenditure or burn to attempt mining**. This means participating in PoTen has no downside risk to one's token holdings aside from the opportunity cost of time and the gas fees. A user can attempt as many times as they like; unsuccessful attempts do not consume BTEN – they only reset the clock. This encourages participation because even if you fail to win, you can try again later after accruing more age. In all cases, every attempt and its outcome is transparently logged via on-chain events (`Attempt` events for each try, and `Win` events for successes), so the entire mining process is auditable in real time.

Randomness Source

The fairness of PoTen's lottery relies on an unpredictable random draw for each attempt. In the current implementation, the contract uses a hash involving recent block data and the address's attempt count to generate a pseudo-random number. For example, it can incorporate the previous block's hash (or the block's `prevrandao` field) combined with the address and a nonce, making the outcome not easily manipulable by the user. However, it should be noted that block producers (miners/validators on BSC) have some influence over block hashes, so this on-chain randomness is not as secure as a dedicated oracle-based random beacon. In practice, given the high frequency of wins (~every 10 seconds network-wide) and the relatively low stakes per attempt, the risk of manipulation is low – there is no single "jackpot" event that a malicious miner could target without significant cost and coordination.

(While a more secure randomness source like a VRF could further improve trustlessness, the BTEN contract is immutable and does not have an upgrade mechanism; as noted later, introducing such an improvement would require a new contract deployment.)

In essence, the PoTen mining model turns BTEN into a kind of on-chain time-lottery: every token holder accumulates "tickets" (potential) by holding tokens over time, and a mining attempt is like buying a lottery draw with those tickets. But unlike a traditional lottery, the tickets are free (they come simply from holding the asset) and there's no punishment for losing aside from starting to accumulate again. This mechanism incentivizes healthy network behavior: users are rewarded for stability (holding) and for participating in the protocol (attempting mining), and there is no advantage to out-of-band behaviors like frequent trading or hoarding multiple addresses (since those tactics are nullified by the age reset rules). It democratizes mining – any user can mine with 1 BTEN just as with 1,000,000 BTEN; the only difference is the scale and speed of their probability accrual. Fairness is maintained by the fact that probability scales linearly with both balance and time, so a 10× larger balance is equivalent to waiting 10× longer with a smaller balance. This balance–time symmetry is a key innovation of PoTen, offering a new avenue in consensus-light token distribution.

Adaptive Difficulty Tuning

To maintain a consistent rate of token emission, BTEN employs an automatic difficulty retargeting algorithm similar in spirit to Bitcoin's difficulty adjustment but operating on a much faster timescale and using a continuous formula. The goal is to ensure that wins occur at an average interval of about 10 seconds network-wide. Without such a mechanism, if many users suddenly start attempting frequently, the win rate could spike (many more wins per second, accelerating emission), or if activity drops, wins could become too infrequent. The adaptive difficulty keeps the system self-regulating by dynamically adjusting the mining difficulty after each win.

Difficulty Parameter

In the context of PoTen, **difficulty** (*D*) can be thought of as the threshold that the random number must overcome relative to a miner's potential *P*. A higher difficulty means each attempt is harder to win (it requires a larger *P* or more luck), analogous to how Bitcoin's mining difficulty works. In the BTEN contract's implementation, this is realized via a target value for the random hash: initially, the target is set such that roughly 1 in 16 attempts would succeed (a "soft start" giving a moderate success probability). As the network progresses, the target (or difficulty) is continually adjusted to keep the win frequency near the 10 s target interval.

Retarget Formula

After every win, the contract computes the time Δt that elapsed between this win and the previous win (the inter-win interval). It then updates the difficulty based on Δt relative to the target interval (TARGET = 10 seconds). BTEN uses an exponential moving average-style update with clamped step sizes for stability. The retarget process can be summarized as follows:

1. **Compute ratio:** Calculate $r = \Delta t / TARGET$.
2. **Adjust difficulty if fast:** If $r < 1.0$ (wins were faster than target), increase the difficulty (make mining harder).
3. **Adjust difficulty if slow:** If $r > 1.0$ (wins were slower than target), decrease the difficulty (make mining easier).
4. **Apply bounds:** Limit the adjustment factor so that difficulty can at most double (×2) or halve (×0.5) in one step. In other words, clamp *r* to the range [0.5, 2.0] before applying it. For example, if the last win came after 5 seconds (Δt = 5, which is half the target), then r = 0.5 but the clamp sets a minimum of 0.5, so difficulty doubles (making future wins harder). If the last win took 30 seconds (Δt = 30, three times the target), r = 3 but the clamp maxes it at 2.0, so difficulty is halved (making future wins easier).

Note: A smaller smoothing factor α (e.g. 20%) was considered to blend new data with old, but the final formula effectively uses the immediate ratio (α = 1) with the above clamps, simplifying the adjustment logic. After computing the new difficulty, the contract updates the recorded last win timestamp and emits a `DifficultyAdjusted` event with details of the change.

Overflow-Safe Implementation

The retarget calculation is done in integer arithmetic on-chain, which requires care to avoid overflow given that difficulty (or its internal target threshold) can be a 256-bit number. The contract's implementation uses a safe technique of splitting the multiplication into parts (quotient and remainder) when applying the factor in basis points. It essentially computes `newDifficulty = prevDifficulty * factorBps / 10000` in a way that cannot overflow a 256-bit number, by splitting *prevDifficulty* into multiples of 10000 and a remainder. This ensures that even at extreme difficulty values, the multiplication will not wrap around. The end result is then clamped between predefined constants TARGET_MIN and TARGET_MAX, which represent the absolute bounds of difficulty in the system (these correspond to probabilities of essentially 0% and 100% respectively, with the initial TARGET_MAX set to allow a generous ~1/16 chance). By construction, difficulty is also never allowed to fall below 1 (the minimum divisor for probability). This careful arithmetic design guarantees that the difficulty adjustment is numerically stable and cannot err even under sudden changes in Δt.

Time-to-Difficulty Feedback

The feedback loop can be summarized in simpler terms:

- If wins are happening too **quickly** (Δt < 10 s on average), the difficulty increases, making each attempt's chance smaller. This slows down the rate of wins.
- If wins are happening too **slowly** (Δt > 10 s on average), the difficulty decreases, giving each attempt a higher chance. This speeds up the win rate.

In equilibrium, this negative feedback aims to achieve a steady state where Δt hovers around ~10 seconds on average. Not every win will be exactly 10 seconds apart (there will naturally be variance), but over many wins the average should stabilize near the target. The chosen clamp approach ensures the system reacts to changing conditions but is constrained by the 0.5–2× adjustment limits per step, preventing oscillations or runaway difficulty.

Startup Conditions – High Initial Chance

At genesis, the network has no prior wins, so difficulty starts at a preset level to ensure early miners have a reasonable chance. In BTEN's case, the initial target threshold (difficulty inverse) was set such that the first win probability was quite high (on the order of a few percent per attempt). This "warm start" means the very first mining attempts after deployment are likely to produce a win quickly (within a handful of attempts), rather than forcing a long, uncertain initial wait. Once the first win occurs, the normal retarget logic takes over.

- If initial participation is low (few people attempting at launch), after the first win the difficulty will actually drop over time (since wins come slowly), thereby further increasing everyone's chances until activity picks up. In effect, the protocol guarantees that mining will get easier and easier (down to a defined minimum difficulty) until someone wins, so that the network doesn't stall early on.
- Conversely, if many users try at once right at launch and wins come faster than 10 s apart, the difficulty will rapidly increase to maintain control.

This adaptive behavior ensures a smooth start and protects the emission schedule from wild deviations. Through this adaptive difficulty tuning, BTEN achieves a self-regulating emission of tokens. No matter how participant behavior changes, the system will modulate itself to keep the overall output on track for ~10-second wins and 210,000-win epochs (the intended halving interval). This is essential for long-term stability, preventing the supply from being exhausted prematurely during surges of activity and ensuring that during lulls the remaining supply still trickles out at a reasonable pace. The design does not require any manual intervention or governance to adjust the difficulty – it is entirely algorithmic, much like Bitcoin's difficulty adjustment but on a faster, continuous cycle. The on-chain record of `DifficultyAdjusted` events provides transparency into this process, allowing anyone to observe how the difficulty evolves over time and to verify that it responds correctly to network conditions.

Security and Simplicity Principles

BTEN's smart contract was developed with a strong bias toward security, simplicity, and minimizing trust assumptions. Several guiding principles and design choices reflect this philosophy:

No Privileged Roles

The contract has no owner-only or privileged functionality that can interfere with day-to-day operation. After deployment, the core parameters (like cap, rewards, difficulty adjustment logic) are fixed and cannot be altered by anyone. There is no concept of an admin who can, for instance, arbitrarily mint tokens or pause the mining; the deployer's role was only to initialize the contract. In fact, after roughly the first ~20,000 wins (including the genesis mint), the deployer's address was excluded from mining further to eliminate any perception of unfair advantage. The deployer simply stopped mining after this bootstrap phase and holds no special power thereafter. This absence of special roles means the BTEN contract operates as a credibly neutral protocol – all participants face the same rules, and no one can centrally modify those rules.

No Upgradability

BTEN is deployed as a single, non-proxy, immutable contract on BSC. It does not use an upgradeable proxy pattern. While upgradeable contracts can offer flexibility, they introduce a trust vector (administrators could change the code later) and complexity that can lead to bugs. By choosing immutability, BTEN guarantees that the code hash and logic that were audited and open-sourced remain the exact code that governs the token forever. This eliminates the risk of "rug pulls" via contract upgrade and simplifies the mental model for security – one only needs to analyze the static code. Of course, this means any future improvements (like switching to a more secure randomness source such as a VRF) would require a new contract deployment or auxiliary mechanisms, but that trade-off was deemed worth the increased security and trustlessness.

Minimalistic Code Surface

The BTEN contract's code is concise and based on well-known standards. It implements the basic ERC-20/BEP-20 functions (`transfer`, `approve`, `transferFrom`) in a straightforward manner, without any custom modifications that could introduce vulnerabilities. The novel part is the `attempt()` function and the internal mining logic, which were written from scratch with an emphasis on clarity. The entire contract (including comments) is only on the order of a few hundred lines. This small code size and relative simplicity make it easier to audit and reason about. Standard security pitfalls like re-entrancy are avoided: there are no external calls during an attempt (it's just internal calculations and state updates), and the contract does not hold or manage BNB or other tokens that

could invite complex attacks. The state variables related to mining (ages, attempt counts, difficulty, etc.) are updated in a single cohesive flow with proper checks, reducing the chance of inconsistencies. Moreover, invariants were identified and enforced, such as difficulty never dropping below 1, age resets happening exactly as specified, and total supply never exceeding the cap.

Audit and Testing

Prior to mainnet release, the BTEN contract underwent rigorous testing, including deployment on BSC testnet and simulation of the mining process through an entire halving cycle (one full epoch of 210,000 wins) to ensure stability. The development team also produced scripts to automate rapid mining attempts and monitor contract behavior in real-time, which helped validate the difficulty adjustment and halving logic under high load. The deterministic nature of the contract (no external oracles needed for core functions) made it straightforward to test by replaying scenarios and verifying that outcomes (like total supply at each epoch transition) matched expected values. An external third-party audit was planned or underway at the time of launch, focusing on potential edge cases such as integer overflow (which was mitigated in the difficulty adjustment as described), random manipulation, and any denial-of-service vectors. The simplicity of the design (no complex math beyond some 64-bit arithmetic for rewards and 256-bit math for difficulty) means that formal verification of key properties was feasible. For example, one could formally verify that `totalSupply` never exceeds the cap given the coded logic, or that difficulty adjustments converge as intended. From a security standpoint, the biggest reliance is on the underlying blockchain's randomness and honest execution; aside from that, BTEN's contract tries to minimize features that could be misused.

No External Dependencies

BTEN does not depend on any external contracts or governance systems for its core operation. There are no oracles (aside from optionally using block hashes for on-chain randomness), no price feeds, no reliance on off-chain processes. This reduces points of failure and attack. The push-only airdrop mechanism (described later) also means users never have to sign messages or interact with potentially malicious dApps to receive tokens, greatly reducing phishing risk. The official web infrastructure is deliberately kept simple (no wallet connections, just static information) to avoid introducing security issues outside the blockchain. This all contributes to an overall posture that prioritizes user safety and trustlessness.

Transparent Operation

Every significant action in the BTEN system is broadcast as an event on-chain, which is integral to security and community oversight. As detailed earlier, events like `Attempt`, `Win`, `DifficultyAdjusted`, and `PhaseChanged` (for epoch/phase shifts) are emitted in real time. This means anyone can subscribe to these events (e.g., via a block explorer or a custom dashboard) and verify the fairness of what's happening: which address won, how difficulty changed, how often attempts are occurring, etc. If anything anomalous were to occur (say, a sudden unexpected change in difficulty or an address winning disproportionately often), it would be immediately visible to the community, allowing scrutiny. This transparency is a core security feature because it crowd-sources the monitoring of the system to all interested parties.

In summary, BTEN's design reflects a "security-first, simplicity-first" mantra. By eliminating admin controls, avoiding upgradability, keeping the code lean, and exposing all mechanics on-chain, BTEN reduces both the likelihood of bugs and the impact of any single actor's malicious behavior. The result is a token system that one can use with a high degree of confidence that the rules will not change and

that there are no hidden pitfalls. What you see in the code is what you get in the live system – a property crucial for a decentralized token aiming to earn users' trust.

Network Integration (BSC Compliance and Mining Setup)

BTEN is deployed on the BNB Smart Chain (BSC) and adheres fully to the BEP-20 token standard, which ensures compatibility with the broader BSC ecosystem (exchanges, wallets, explorers, etc.). In practical terms, BTEN behaves like any other BSC token for transactions and balances, which made integration and verification straightforward. The contract was verified on BscScan (the BSC block explorer), allowing anyone to inspect the source code and the state of the contract on-chain. Standard BEP-20 calls such as `balanceOf(address)` and `transfer(address, uint256)` are implemented and function as expected, which means existing wallet software can send and receive BTEN without special modifications.

BEP-20 Compliance

The BTEN contract includes the required public variables and events that make it BEP-20 (and ERC-20) compliant: name, symbol ("BTEN"), decimals (8), totalSupply, and the standard `Transfer` and `Approval` events. It also implements `transferFrom` and `approve` for delegated transfers. Because BTEN forgoes any custom transfer taxes or complex logic, every transfer is a simple deduction from the sender and addition to the recipient, making it behave exactly like a conventional cryptocurrency token. This compliance was validated on BSC testnet before mainnet release – for example, the token could be added to MetaMask, transactions showed up correctly on testnet explorers, and developer tools (such as Foundry's *forge* and *cast*) were used to interact with the contract to ensure all standard interfaces responded properly.

Deployment

The deployment of the BTEN contract was executed via a standard Ethereum-style transaction, and the constructor set initial values (such as capturing the genesis timestamp and minting the 50 BTEN to the deployer). The contract was configured with an initial difficulty target (TARGET_MAX) that gave a moderate success probability to bootstrap mining, as discussed above. Right after deployment, the team ran a series of mining attempts using automated scripts to simulate a bootstrap phase. The aim was to generate the first ~20,000 wins internally (by the deployer address) to "warm up" the difficulty adjustment mechanism and progress the emission into later epochs. This was done transparently on-chain; the events during this phase were observable by anyone, and the same rules (reward halving, etc.) applied as in the public phase – the team had no special advantage beyond acting early. Once roughly 20k initial wins were mined as planned, the deployer ceased mining further. By that point, the difficulty had adjusted to an equilibrium level reflecting active mining, and the contract was effectively in a state ready for anyone to participate. The transition to the public phase was not enforced by the contract code (there was no explicit lock on attempts by others), but practically, the contract was only announced to the public once the bootstrap was done. At that moment, the team's address was voluntarily **excluded from further mining**, cementing the guarantee that the team would not continue to capture rewards once the community joined. An official announcement marked this handover, aligning with the ethos of decentralization.

Mining Loop Setup

To facilitate continuous mining, the team provided example scripts and guidance for users who wanted to run a mining loop. Because mining in BTEN simply requires calling `attempt()` (or making self-

transfers) repeatedly with some delay between attempts, one can automate this with a simple program or script. For instance, the team released a script using Foundry's tools that calls `attempt()` in a loop for a specified number of iterations. They also provided a basic shell script to run these batches in perpetuity with a small sleep interval, effectively creating a background miner that continuously tries, prints updated stats (block number, total wins, current reward, etc.), waits, and repeats. This allowed even non-technical users to use a ready-made tool to participate in mining by only setting up a BSC node connection and their wallet key. The miner loop script took care of using a minimal gas price to ensure transactions kept flowing even under varying network conditions. All these artifacts were part of the public release, underscoring the project's commitment to openness and community involvement in the mining process.

For the average user, mining can also be done manually or via wallet automation. (Some community members created simple bots or used wallet scheduling features to send periodic self-transactions.) The key point is that the BTEN contract does not require any special role or permission for mining – any address holding BTEN can call `attempt()` directly. This means BTEN mining is as permissionless as transferring tokens. Users just have to pay the BSC network gas fee for each attempt. At ~10-second target intervals, an extremely dedicated miner might attempt every few seconds or every block; however, many users will find an optimal cadence (perhaps one attempt every few minutes or hours, depending on their balance and patience). Because BTEN is on BSC, the low transaction fees (typically only a few cents) make such frequent attempts economically feasible, which is a critical factor for the viability of PoTen. (An analogous system on a high-fee chain like Ethereum mainnet would be impractical for users due to gas costs.) BSC's environment thus provided the right balance of speed and cost for BTEN to thrive, and integration with BSC's tooling (like BscScan's event logs and APIs) means the community can build dashboards to monitor the mining in real time. Indeed, one envisioned community tool was **TenSight**, an analytics dashboard that would consume BTEN's on-chain events to display live metrics like win rate, difficulty changes, and the distribution of wins across addresses. Such integrations are straightforward given the event transparency of the system.

In summary, BTEN's integration with the BSC network was smooth due to adherence to standards and the inherent simplicity of the token's interface. The project provided both the infrastructure and knowledge for users to engage in mining (from one-click self-transfers to fully automated loops), ensuring that participating in the network is accessible. The token contract's design choices (like 8 decimals and no fees) have been accounted for and verified to pose no compatibility issues. Thus, BTEN operates as a first-class citizen on BSC, leveraging the network's capabilities to deliver its fast mining cadence while being readily monitorable and usable with existing BSC ecosystem tools.

Launch Strategy and Airdrop Distribution

BTEN's launch was orchestrated to prioritize fairness and community involvement. The distribution strategy ensured that no centralized party had a disproportionate advantage and that early participants could join without barriers:

Bootstrap Mining Phase

After contract deployment, the team conducted a short bootstrap mining phase (as described earlier) to kickstart the network. Roughly 20,000 wins were mined by the deployer in quick succession to initialize the difficulty and progress the emission into later epochs. During this phase, all wins and difficulty adjustments were on-chain and transparent, and the same rules applied as in the public phase – the team had no special advantage beyond acting early. This staged approach ensured that at launch, there

were as few moving parts as possible – essentially just the BTEN contract and the social coordination around it – which is a boon for security and reliability.

Public Mining Announcement

Once the bootstrap phase achieved its goal, the contract and mining process were announced to the public (via social channels like Telegram, Twitter/X, and the official website). At this point, mining was "open" to everyone, though in reality it was open from the start – the announcement simply marked when the team stepped back and the community was formally invited in. The deployer's address ceased mining further and became effectively just another holder. The contract, having no whitelists or allowlists, did not prevent anyone from attempting mining at any time; the controlled release was a matter of communication rather than code. This approach gave the community confidence that by the time they joined, the system was already stabilized and no one (including the team) was continuing to accumulate excessive advantage.

Airdrops

To decentralize initial token distribution, BTEN employed a series of free airdrops. Instead of selling tokens or requiring any action from recipients, the team selected various community members and addresses to receive BTEN at no cost. These airdrops were delivered directly to users' wallets ("push" distribution) – meaning recipients did not have to claim or perform any task beyond being active community members or being part of certain early supporter groups. This strategy served two purposes: it avoided common phishing risks (users never had to connect wallets to unknown sites or sign messages to claim tokens) and it spread BTEN holdings widely to kickstart network effects. Airdrop recipients now had BTEN which they could hold or use to start mining attempts, thus engaging them in the project.

The airdrops were executed in multiple waves, with varying criteria to include different segments of the community. For example, one wave targeted active participants in BTEN's community channels; another included people who had interacted with certain related projects, etc. Each wave was publicly communicated after execution, with transaction hashes provided for transparency. The amounts were relatively small per address (to stay fair and not overshoot the cap), but collectively these airdrops distributed a meaningful portion of the early supply to thousands of addresses.

Importantly, the airdrop process was fair and surprise-free: every distribution was done by the team's deployer address using a simple batch transfer script, and no special conditions were attached. Since BTEN has 8 decimals, even fractional token amounts could be airdropped to maximize reach. The community could verify on-chain which addresses received tokens and how much, ensuring there was no hidden allocation. The team also ensured that no single person or entity received an outsized share via airdrops – the distribution was as broad as feasible.

No Token Sale or Fundraise

BTEN did not hold any ICO, presale, or fundraising event. The project was entirely self-funded and community-driven. This means there was no influx of venture capital or private buyers obtaining BTEN at launch – the initial value of BTEN was established purely organically through trading on the open market after launch. The lack of a token sale ensured regulatory simplicity and, more importantly, meant that the community could obtain tokens either for free (via airdrop) or by mining, rather than buying from the team. It eliminated any pressure on the project to serve outside investors and allowed focus on the mining game and community growth.

Liquidity and Trading

After the initial airdrops, a small amount of BTEN (including the 50 BTEN genesis and some mined tokens from the bootstrap) was used to create liquidity on a BSC decentralized exchange so that price discovery could begin. The team provided a BTEN/BNB liquidity pool on a popular DEX with a modest amount of tokens and BNB. This ensured that early community members who wanted to trade could do so, and those who missed the airdrops had an avenue to acquire BTEN if they wished (albeit in small quantities due to the low initial liquidity). The team did not lock liquidity or perform any elaborate market making – the goal was simply to seed an initial market and let it grow naturally. Over time, as community miners earned BTEN, more liquidity was added by users. The market remained fairly small in the MVP stage, which aligned with the ethos that BTEN was not hyped as a speculative asset but rather as a mining experiment.

Community Building

Alongside the technical launch, the team focused on building a community around BTEN. Official channels (Telegram, Twitter/X, a simple website) were set up to disseminate information and updates. Educational material on how to participate in mining was shared, including guides for sending self-transactions, using the provided scripts, or checking the on-chain events. Because BTEN's model is somewhat novel, explaining the "wait and attempt" strategy was important. The team emphasized that small holders could participate meaningfully by accumulating age, and large holders would not win every time due to the probabilistic nature. This messaging helped set expectations and encouraged a culture of sharing experiences (for example, community members often announced when they got a win, fostering excitement and camaraderie).

The launch strategy deliberately kept things minimal and fair. By foregoing any token sale and making the launch process transparent, the team aimed to establish trust and a level playing field. The airdrop approach rewarded early supporters and interested users without excluding those who arrived slightly later (since mining was always available as an entry). In essence, BTEN's launch was a practical demonstration of its principles: fairness (no insider privileges), transparency (all actions on-chain and openly announced), and a community-first approach (letting users earn and own the token through participation).

As BTEN continues to evolve post-MVP, this early history serves as a foundation of goodwill and grassroots support. The token's distribution is now entirely in the hands of the community via mining, and the initial conditions set by the launch have been designed to give the project the best chance at a vibrant, decentralized future without the weight of unfair advantages or misaligned incentives.
