FomoRush Whitepaper

FomoRush Team

2025-09-01

Contents

1 Abstract

FomoRush is a decentralized social experiment and game on the TON blockchain, combining game theory, strategy, and community engagement. Participants compete and cooperate in cyclical rounds, activating keys to extend a timer and grow a prize pool. The project features a robust token (FRSH), a staking system, and a transparent, AI audited smart contract architecture. FomoRush is designed to test human behavior under time pressure and resource constraints, rewarding both strategic play and community participation.

2 Introduction

FomoRush is a blockchain-based game and social experiment where participants collectively influence the duration of a round and compete for a growing prize pool. The game is fully decentralized, with all logic enforced by smart contracts, ensuring transparency and fairness.

---

3 Concept and Social Experiment

FomoRush is built on the principles of game theory. Participants must balance cooperation and competition, making strategic decisions about when to act, how much to invest, and whom to invite. The game tests behavior under time pressure and limited resources, with the fear of missing out (FOMO) driving action as the timer nears zero.

---

4 Game Mechanics

4.1 Keys Activation

- **Activation:** The primary action in the game. Each key activation adds 30 seconds to the round timer.
- **Dividends:** Keys holders receive 43% of the TON from each subsequent keys activation, distributed proportionally. Dividends are capped at x2 the activation cost per key.
- **Restrictions:**
  - While the summary keys activation amount is <1000 TON
    - No consecutive activations by the same participant.
    - Single activation limit: 20 TON.
  - Up to 0.06 TON per activation is reserved for blockchain fees.
- **Distribution:**
  - 43% — dividends to key holders
  - 37% — prize pool (pot)

  - 10% — referral program, in case no referrer comes to jackpot
  - 6% — staking pool
  - 2% — community
  - 1% — next round's pot
  - 1% — jackpot

4.2 Timer and Rounds

- **Round Start:** Timer set to 24 hours.
- **Timer Reduction:** Every 24 hours, the max timer decreases by 4 hours (down to 4h), then by 30 minutes every 4h (down to 30m).
- **Round End:** When the timer reaches zero, the round concludes.

4.3 Prize Pool (Pot)

- **Growth:** 37% of each key activation goes into the pot.
- **Distribution:** At round end:
  - 48% to the last key activator.
  - 22.5% for registered key holders in Stage 3.
  - 17.5% to the next round's pot.
  - 10% to the staking pool.

4.4 Referral Program

- **Invite:** Participants can invite friends via referral links.
- **Reward:** 10% of each referred participant's key activation goes to the referrer.
- **Immutability:** Referrer is set on first activation and cannot be changed.
- **Self-Referral Protection:** Users cannot refer themselves - the smart contract explicitly prevents self-referral through address validation.
- **Security:** Referral relationships are immutable and protected against manipulation attempts.
- **Referral Allocation Clarification:** In cases where a participant has no referrer, the 10% referral reward is automatically redirected to the jackpot pool.

4.5 Jackpot

- **Eligibility:** Keys activated with ≥10 TON qualify.
- **Chance:** 0.3314% chance to win per qualifying activation.
- **Distribution:**
  - 10–20 TON: 15% of jackpot
  - 20–50 TON: 25%
  - 50–100 TON: 50%
  - 100+ TON: 75%

4.6 Game Round Stages

- **Stage 1: Keys activation**
  - Duration: Until timer reaches 0.
  - Actions: Key activation, referral, dividend withdrawal (up to x2 cap).
- **Stage 2: Rewards & Registration (2 days)**
  - Actions: Withdraw dividends (up to x2 cap).
  - **Registration Requirement:** Only players who have NOT reached their dividend x2 limit can register their keys for Phase 3 rewards.
  - **What this means:** If you've already earned dividends equal to twice the total amount you spent on key activations during the current round, you won't be eligible to register for additional Stage 3 rewards.
  - Unclaimed dividends will be redistributed among registered participants on Stage 3

- **Stage 3: Reward Distribution (1 day)**
  - 22.5% of the pot + unclaimed dividends from Stage 2 are distributed to registered participants, proportional to their key count.

---

5 Tokenomics

5.1 FRSH Token - Investment Opportunity

- **Name:** FRSH
- **Total Supply:** 50,000,000 FRSH (50 million tokens)
- **Token Sale:** All tokens sold across 5 stages with increasing prices
- **Price Progression:** From 0.005 TON to 0.05 TON
- **Utility:** Staking for TON dividends from gaming ecosystem
- **Security:** Hard cap enforced, price floor protection (never decreases)

5.2 Token Sale Stages - Early Bird Advantage

Stage	Price per FRSH	Investment Opportunity
1	0.005 TON	**Early access to lowest token price**
2	0.0125 TON	**2.5× higher than Stage 1 price**
3	0.02 TON	**4× price progression from Stage 1**
4	0.035 TON	**7× price from initial offering**
5	0.05 TON	**Final price tier at 10× initial level**

5.3 Staking Pool - Passive Income Generator

- **Funding Sources:**
  - 6% of each key activation (continuous income)
  - 10% of final prize pool at round end (bonus rewards)
- **Staking Cycle:** 9 days total
  - **Deposit Phase (2 days):** Stake FRSH tokens
  - **Lock/Accumulation (5 days):** Tokens locked, rewards accumulate
  - **Withdrawal Phase (2 days):** Withdraw tokens and dividends
- **Yield:** depending on ecosystem activity
- **Distribution:** Proportional to staked share
- **Security:** Unclaimed dividends redistributed to active stakers

5.4 Revenue Distribution - Transparent Allocation

Token Sale Revenue Distribution:

- **35% Prize Pool** - Fuels gaming ecosystem rewards
- **15% Marketing** - Drives user acquisition and growth
- **15% Development** - Ensures continuous platform improvement
- **15% Team** - Aligns team incentives with project success
- **10% Partners** - Strategic partnerships and ecosystem expansion
- **5% Reserve** - Future project needs and opportunities
- **5% Ecosystem** - User activity support and operational needs

5.5 Game Revenue Distribution - Continuous Value Creation

Each key activation generates value distributed as follows:

- **43% Dividends** - Direct rewards to key holders (up to 2x cap)
- **37% Prize Pool** - Growing pot for round winners
- **10% Referral Program** - Growth incentives (in case no referrer comes to jackpot)
- **6% Staking Pool** - Passive income for FRSH holders
- **2% Community** - Community development and engagement
- **1% Next Round** - Sustainable ecosystem funding
- **1% Jackpot** - Additional reward opportunities

5.6 Investment Highlights

Token Utility and Ecosystem Participation:

- Structured price tiers from 0.005 to 0.05 TON
- Earning opportunities linked to ecosystem activity
- FRSH holders can participate in regular staking rewards

Ongoing Utility and Participation Rewards:

- Token utility tied to in-game and staking activity
- Community-driven reward mechanisms

Protocol Design and Distribution:

- All tokens sold to investors (100% decentralization)
- No team token allocation (incentivized via TON)
- Hard cap enforced, price floor protection
- Transparent revenue distribution

5.7 Sustainability & Long-Term Utility

FomoRush is designed as more than just a single game. It serves as the foundation of a growing and evolving gaming ecosystem on the TON blockchain. With plans to introduce new game modes, expanded features, and user-to-user token markets, the project aims to sustain long-term engagement and diversify revenue streams. Core mechanisms such as staking, referral incentives, and transparent revenue sharing ensure ongoing utility for the FRSH token and continuous rewards for participants. Coupled with security AI audits, open-source development, and strategic partnerships, FomoRush is committed to fostering a vibrant, sustainable community and delivering lasting value to all stakeholders.

---

6 Smart Contract Architecture

6.1 MainGame Contract

- Orchestrates the game, manages rounds, stages, and pot.
- Handles key activations, timer logic, reward distribution, and stage transitions.
- Enforces security via Checks-Effects-Interactions, input validation, and sender checks.

6.2 UserGameWallet Contract

- Represents each player's in-game wallet.
- Tracks keys, spent/gained amounts, and round participation.
- Handles reward claims, referrer bonuses, and access control.

6.3 Jetton (Token) Contracts

- **JettonMinterICO:** Issues FRSH tokens, manages ICO, enforces hard cap and price floor.
- **JettonWallet:** Holds FRSH tokens for users, handles transfers, burns, and security checks.

6.4 StakeBank (Staking) Contract

- Accepts FRSH deposits for staking.
- Tracks user stakes, calculates/distributes rewards, and handles withdrawals.
- Integrates with MainGame for distributing staking rewards.

6.5 Randomness and Security

- Randomness for jackpot and other features is on-chain and verifiable, using logical time, contract balance, and address as entropy sources.

---

7 Security and Audit

- All contracts are written in FunC and thoroughly tested.
- Security AI audits address:
  - Cross-contract state consistency
  - Input validation
  - Randomness entropy
  - Gas management
  - Integer overflow/underflow
  - Access control
  - Bounce handling
- AI audit reports are public and the codebase is open source.

---

8 Business Model and Go-to-Market

- **Revenue Streams:**
  - A portion of each key activation and token sale funds project development and community initiatives.
  - Staking and referral programs incentivize long-term engagement.
- **Community Growth:**
  - Viral referral system and social sharing.
- **Sustainability:**
  - Hard cap on token supply and price floor prevent inflation and manipulation.
  - Transparent, open-source code and regular AI audits build trust.
- **Future Expansion:**
  - New games modes.
  - Partnerships with other TON ecosystem projects.
  - Direct user-to-user app tokens market.

---

9 Roadmap & Future Development

9.1 Short-term (Q3-Q4 2025)

- ICO completion across 5 stages

- Platform launch and user onboarding
- Staking mechanism activation
- Community building

9.2 Medium-term (Q1-Q2 2026)

- New game modes and features
- Partnership expansions
- Enhanced staking options

9.3 Long-term (Q3 2026+)

- Ecosystem expansion

---

10 Technical Appendix: Key Mechanisms and Code Snippets

10.1 Key Price Formula (Bonding Curve)

Formula (simplified):

```
price_next = price_start + delta * total_keys
```

Where:

- `price_start` is the initial key price (e.g., 0.001 TON)
- `delta` is the incremental price increase per key
- `total_keys` is the number of keys that have already been activated

FunC Example:

```
int get_key_price(int total_keys, int price_start, int delta) inline {
    return price_start + delta * total_keys;
}
```

10.2 Key Activation and Dividend Distribution

FunC Example:

```
int amount = in_msg_value();
int to_dividends = amount * 43 / 100;
int to_pot = amount * 37 / 100;
int to_referral = amount * 10 / 100;
int to_staking = amount * 6 / 100;
int to_community = amount * 2 / 100;
int to_next_round = amount / 100;
int to_jackpot = amount / 100;
```

10.3 Dividend Cap (x2 Rule)

FunC Example:

```
throw_unless(error::dividend_limit, user_dividends <= user_spent * 2);
```

10.4 Timer Reduction Logic

FunC Example:

```
int max_timer = 24 * 60 * 60; // 24 hours in seconds
if (round_duration > 4 * 60 * 60) {
    max_timer -= 4 * 60 * 60; // reduce by 4 hours
} else {
    if (round_duration > 30 * 60) {
        max_timer -= 30 * 60; // reduce by 30 minutes
    }
}
```

10.5 Security: Input Validation and Access Control

Sender Validation:

```
throw_unless(error::unauthorized, equal_slices_bits(main_game_address, sender_address));
```

Operation Code Validation:

```
int op = in_msg_body~load_op();
if (op != op::activate_key && op != op::withdraw_dividends) {
    throw(error::unknown_op);
}
```

10.6 Randomness for Jackpot

FunC Example:

```
int get_random_byte(slice address, int seed) inline_ref {
    cell mix = begin_cell()
        .store_slice(address)
        .store_uint(cur_lt(), 64) // logical time
        .store_coins(seed)
        .store_uint(balance(), 128) // contract balance
        .store_uint(my_address().preload_uint(256), 256) // contract address
        .end_cell();
    int hash = cell_hash(mix);
    int index = (hash % 30) + 1;
    return get_byte(hash, index);
}
```

10.7 Staking Pool Cycle Logic

FunC Example:

```
int cycle_time = now() % 9 * 24 * 60 * 60; // 9-day cycle
if (cycle_time < 2 * 24 * 60 * 60) {
    // Deposit phase
} else {
    if (cycle_time < 7 * 24 * 60 * 60) {
        // Lock/Accumulation phase
```

```
    } else {
        // Withdrawal phase
    }
}
```

10.8 Jetton Token Hard Cap and Price Floor

Hard Cap Enforcement:

```
throw_unless(error::hard_cap, total_supply + mint_amount <= HARD_CAP);
```

Price Floor Enforcement:

```
throw_unless(error::price, new_price >= current_price);
```

10.9 Cross-Contract Communication

Sending Internal Message:

```
send_raw_message(
    build_internal_message(
        to: user_wallet_address,
        value: toNano(0.05),
        body: build_send_keys_body(...)
    ),
    0
);
```

10.10 Error Handling and Bounces

Bounce Handler Example:

```
() on_bounce (slice in_msg_body) impure inline_ref {
    in_msg_body~skip_bounce_flag();
    int op = in_msg_body~load_op();
    if (op == op::ask_keys_share) {
        // unlock pending request
        // ... update state ...
    }
}
```

10.11 Self-Referral Protection

FunC Implementation:

```
if ((has_referrer == 0) & ~ in_msg_body.slice_empty?()) {
    slice referrer_candidate = in_msg_body~load_msg_addr();
    if (~ equal_slices_bits(owner, referrer_candidate)) {
        referrer = referrer_candidate;
        has_referrer = 1;
    }
}
```

Security Features:

- **Address Validation:** `~ equal_slices_bits(owner, referrer_candidate)` prevents self-referral
- **First-Activation Only:** `has_referrer == 0` ensures referrer can only be set once
- **Immutability:** Once set, referrer relationship cannot be changed
- **Manipulation Prevention:** Protects against sybil attacks and referral farming

---

11 Appendix: Contract Addresses and Links

- MainGame Contract: [Deployed Address / Source Link]
- UserGameWallet Contract: [Deployed Address / Source Link]
- JettonMinterICO Contract: [Deployed Address / Source Link]
- JettonWallet Contract: [Deployed Address / Source Link]
- StakeBank Contract: [Deployed Address / Source Link]
- Audit Reports: [MAINGAME_SECURITY_AUDIT_REPORT.md, TOKENS_SECURITY_AUDIT_REPORT.md, STAKEBANK_REAUDIT_REPORT.md]
- GitHub Repository: [Your Repo Link]

---

12 Legal Disclaimer

Decentralization and Governance

FomoRush is a decentralized application deployed on the TON blockchain. The platform operates autonomously through publicly available smart contracts that manage game logic, token flows, and reward distribution without reliance on centralized control. The codebase is open-source and intended for community use and experimentation. There is no legal entity, corporation, or centralized organization behind FomoRush; its development has been contributed by independent individuals. Participants engage with the system at their own discretion, with all functionality governed by the underlying smart contracts.

Disclaimer

This document is provided for informational purposes only and does not constitute an offer or solicitation to sell shares or securities in FomoRush or any related or associated entity. The information herein does not constitute investment advice, financial advice, trading advice, or any other sort of advice. Participation in FomoRush is entirely voluntary and at the user's own risk. As the project is fully decentralized and governed by smart contracts, no guarantees or warranties are provided regarding outcomes, rewards, or platform availability. Users are advised to conduct their own due diligence and comply with local laws and regulations before engaging.

For more information, visit fomorush.io or join our community on Telegram (@FomoRushOfficial).
