Technoeconomics. 2025, Vol. 4, No. 3 (14). Pp. 64–77.
Техноэкономика. 2025, Том 4, № 3 (14). С. 64–77.

Scientific article
UDC 330.47
DOI: https://doi.org/10.57809/2025.4.3.14.7

DECAY-WEIGHTED FAIR PRICE FOR SOLANA BLOCKCHAIN TOKEN BUYBACK

Jose Emilio Phinney Dominguez ✉

Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia

✉ josemilio.1995@gmail.com

Abstract. This article investigates the problem of determining a fair reclaim price in fractionalized token markets on the Solana blockchain. The research object is the reclaim process, where minority holders must be compensated without exposure to short-term manipulation. The method introduces a program-derived account structured as an 11.5 kB circular buffer storing 720 hourly market prices, combined with an exponential decay weighting scheme with a 24-hour half-life to compute a time-weighted average price. Simulation results demonstrate that extreme pump or dump attempts within the final 72 hours before reclaim have a negligible effect on the computed value. The study concludes that the proposed mechanism ensures fairness for minority holders while maintaining cost efficiency in storage and transaction fees.

Keywords: fractionalized tokens, manipulation-resistant time-weighted average price, exponential decay weighting, Solana program-derived address account, circular buffer, on-chain pricing, reclaim mechanism, OpenBook order book, gas-efficient storage, DeFi fairness

Citation: Phinney Dominguez J.E. Decay-weighted fair price for Solana blockchain token buyback. Technoeconomics. 2025. 4. 3 (14). 64–77. DOI: https://doi.org/10.57809/2025.4.3.14.7

Научная статья
УДК 330.47
DOI: https://doi.org/10.57809/2025.4.3.14.7

МОДЕЛЬ DECAY-ВЗВЕШЕННОЙ СПРАВЕДЛИВОЙ ЦЕНЫ ДЛЯ ВЫКУПА ТОКЕНОВ НА БАЗЕ БЛОКЧЕЙНА SOLANA

Хосе Эмилио Финни Домингес ✉

Санкт-Петербургский политехнический университет Петра Великого,
Санкт-Петербург, Россия

✉ josemilio.1995@gmail.com

Аннотация. Данная статья исследует проблему определения справедливой цены выкупа на рынке фракционных токенов в блокчейне Solana. Объектом исследования является процесс выкупа, при котором minor-держатели должны получить компенсацию без риска в краткосрочной перспективе. Метод включает использование аккаунта, структурированного в виде кольцевого буфера объемом 11,5 кБ, хранящего 720-часовую рыночную стоимость, в сочетании с схемой взвешивания с экспоненциальным затуханием в 24 часа для вычисления скользящей средней цены, взвешенной по времени. Результаты моделирования показывают, что экстремальные попытки pump-and-dump (памп-энд-дамп [“накачать и сбросить”]) в последние 72 часа перед выкупом оказывают незначительное влияние на рассчитанную стоимость. По результатам исследования, авторам удается заключить, что предложенный механизм обеспечивает справедливую цену для minor-держателей, при этом сохраняя эффективность затрат на хранение и комиссии за транзакции.

Ключевые слова: фракционные токены, манипуляционно-устойчивая взвешенная по времени средняя цена, взвешивание с экспоненциальным затуханием, аккаунт Solana, кольцевой буфер, ценообразование в блокчейне, механизм выкупа, OpenBook, энергоэффективное хранение, DeFi-справедливость

Для цитирования: Финни Домингес Х.Э. Модель decay-взвешенной справедливой цены для выкупа токенов на базе блокчейна Solana // Техноэкономика. 2025. Т. 4, № 3 (14). С. 64–77. DOI: https://doi.org/10.57809/2025.4.3.14.7

Introduction

Fractionalization Protocols

Fractionalization protocols (CoinMarketCap Academy) are decentralized systems that allow a digital asset, such as a token or NFT, to be split into many smaller fractions that can be traded individually (Chen, 2020; Mishra, 2024). For example, an artwork represented by a non-fungible token (NFT) (NFT Evening) can be fractionalized into hundreds or thousands of fungible tokens (Figure 1), enabling many users to hold partial ownership.

At some point, when a majority holder decides to reclaim the full asset, the protocol must determine a fair price to compensate the minority holders who give up their fractions. This reclaim price is critical, as it ensures fairness in the protocol and prevents disputes (Figure 2).

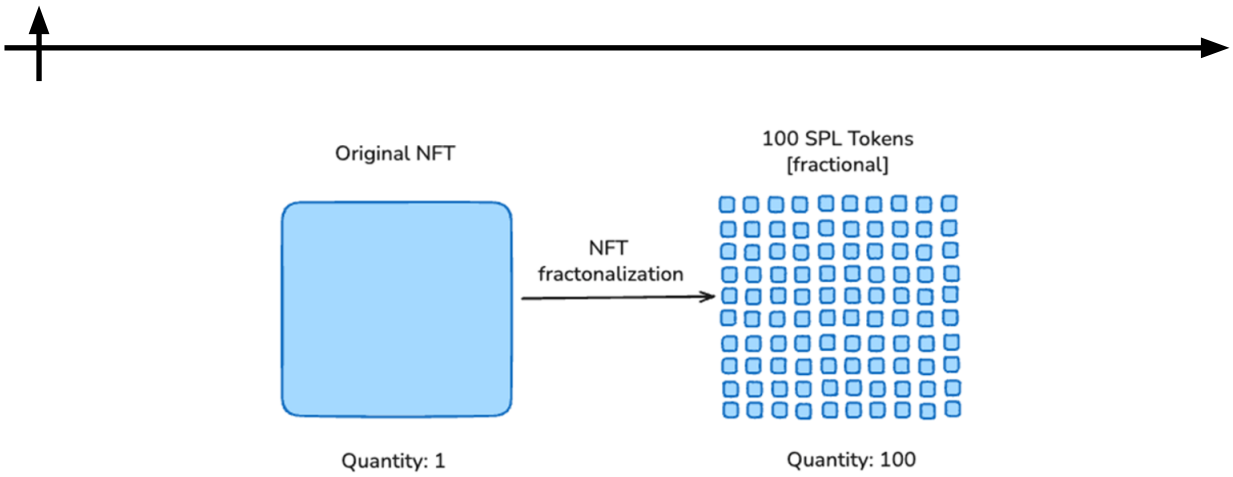


Fig. 1. Fractionalization.

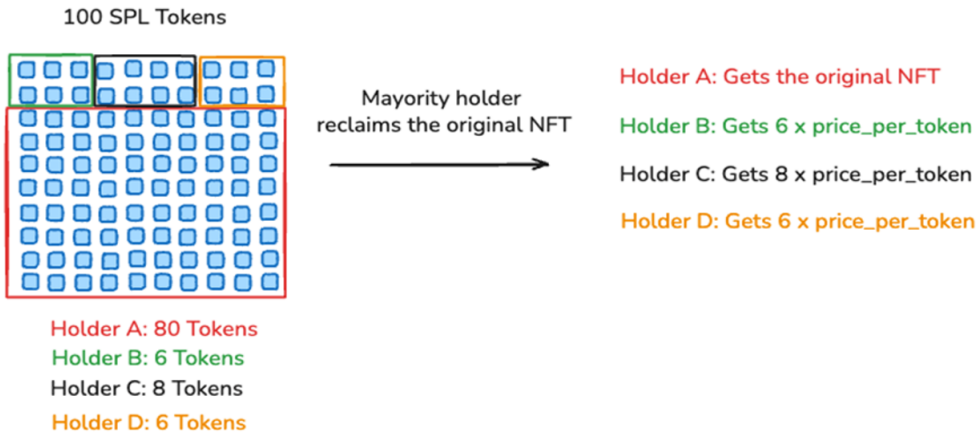


Fig. 2. Reclaim.

This work addresses the challenge of determining a fair, manipulation-resistant price for token reclaim events in fractionalized markets on the Solana blockchain (Solana Foundation). When tokens are fractionalized and later reclaimed, the reclaim price must reflect a stable and unbiased market value. Without safeguards, last-minute price manipulation, through artificial pumps or dumps (Fama, 1965; Faheem, 2024), could distort the average price, allowing majority holders to unfairly benefit at the expense of minorities. To solve this, we design a two-layer time-weighted average price (TWAP) system. First, we maintain a rolling 30-day window of hourly prices collected directly from the market. Second, we apply exponential decay weighting so that earlier values dominate, while the most recent values have negligible influence. This ensures resilience against manipulation attempts close to the reclaim event.

Requirement and Goal

The requirement is that a pool or market must exist for at least 30 days before a reclaim operation can be executed. During this period, hourly prices are recorded regardless of volume. The goal is to ensure that any reclaim price is derived from long-term market behaviour rather than short-term volatility (Lo, 1999; Painter, 2024; Pezzella, Pliushch, 2022).

Specifically, the system must:

— Collect and store 720 recent hourly prices (~30 days).

— Enforce exactly one price record per hour to avoid spam or manipulation.

— Provide a reclaim price that reflects long-term equilibrium.

— Prevent last-minute trades from altering the TWAP outcome.

Materials and Methods

The foundation of the design is a program-derived account (PDA) (Solana Foundation) called a PriceRing (Figure 3), which stores the most recent 720 hourly prices. The PriceRing is implemented as a circular buffer, meaning that once 720 entries are filled, the oldest entry is overwritten automatically by the newest price.

The account layout includes a pointer (head) to indicate the next write position and an array of 720 PricePoint structures, each containing a slot number and a price. The total memory requirement is approximately 11.5 kB, corresponding to 0.081125 SOL (Figure 4) of rent (Solana Foundation. Rent-Exemption and Accounts) on the Solana mainnet.

```rust
#[account(zero_copy)]
#[derive(Debug)]
pub struct PriceRing {
    pub head: u16,                // 2 bytes
    pub bump: u8,                 // 1 byte
    pub _pad: [u8; 5],            // 5 bytes
    pub points: [PricePoint; 720] // 720 * 16 bytes
}
const_assert_eq!(size_of::<PriceRing>(), 2 + 1 + 5 + 720 * 16);

#[derive(Copy, Clone, Debug, AnchorSerialize, AnchorDeserialize, Zeroable, Pod)]
#[repr(C)]
pub struct PricePoint {
    pub slot: u64,                // 8 bytes
    pub price: u64,               // 8 bytes  price in lots
}
```

Fig. 3. Program Derived Address (PDA).

```
joseemilio@MacBook-Air-de-Jose twitter-solana % solana config set --url https://api.mainnet-be
Config File: /Users/joseemilio/.config/solana/cli/config.yml
RPC URL: https://api.mainnet-beta.solana.com
WebSocket URL: wss://api.mainnet-beta.solana.com/ (computed)
Keypair Path: /Users/joseemilio/.config/solana/id.json
Commitment: confirmed
joseemilio@MacBook-Air-de-Jose twitter-solana % solana rent 11528
Rent-exempt minimum: 0.08112576 SOL

joseemilio@MacBook-Air-de-Jose twitter-solana %
```

Fig. 4. Rent-exempt on Solana mainnet for 11.528 kB.

Writing Logic and Circular Buffer

In the writing logic (Figure 5), we will store the newest 720 hourly prices in a circular buffer inside the PDA. That is, every ~9000 slots (1 hour) (Solana Foundation. Slots and Time), we call an instruction that reads the latest traded price and writes it into the ring.

```
// 1. Fetch price (lots)
let price: i64 = get_price(
    &ctx.accounts.event_heap,
    prev_price,          // PricePoint we stored 1 h ago (None for 1st hour)
    &ctx.accounts.bids,
    &ctx.accounts.asks,
)?;

// 2.  Circular write
let idx = ring.head as usize;        // where we write NOW
ring.points[idx] = PricePoint {
    slot: Clock::get()?.slot,        // current slot
    price: price as u64,             // lots
};
ring.head = (ring.head + 1) % 720;   // advance pointer (wraps at 719 => 0)
```

Fig. 5. Writing logic.

As we can see, after 720 calls, the pointer has wrapped once; in this case, ring.head represents the oldest entry and will be overwritten in the next hour, which means that we have an O(1) circular buffer. Furthermore, since this instruction writes the 16-byte struct PricePoint, it incurs a gas fee per update.

Slot-Gate (Anti-Spam)

Next, to enforce exactly one write per hour and prevent spam (someone calling multiple times in an hour), we add a slot gate at the beginning of the instruction (Figure 6):

```
// Every 9 000 slots (≈ 1 hour):

// 1. Gate: only one write per hour
const HOUR_SLOTS: u64 = 9_000; // ~1 h at 400 ms/slot
let clock = Clock::get()?;
let due_slot = market.creation_slot + (HOUR_SLOTS * (ring.head as u64));
require!(clock.slot >= due_slot, TooSoon);
```

Fig. 6. Slot gate.

As we can see:

— ring.head acts as an implicit hour-counter (mod 720).

— The formula gives the earliest slot at which the next price is allowed.

— Calling before that slot fails with TooSoon and costs only the transaction fee.

— Calling on or after that slot succeeds, stores the price, and advances ring.head.

Let us assume that the creation slot is slot number 100 (market.creation_slot = 100) for simplicity of illustration in Table 1.

Table 1. Example of writing logic.

ring.head (%-720)	hours	due_slot calculation	allowed succeeding slots	What happens after writing
0	0	100+9000x0=100	100 ... 8999	store at idx 0, head → 1
1	1	100+9000x1=9100	9100 ... 17999	store at idx 1, head → 2
2	2	100+9000x2=18100	18100 ... 26999	Store at idx 2, head → 3
...	...	...	...	...
719	719	100+9000x719=6471100	6471100 ... 6479999	store at idx 719, head → 0 (wraps)
0 (again)	720	100+9000x720=6480100	6480100 ... 6488999	overwrites idx 0, head → 1

So, anyone can call at any time, but the call will be processed only if it is in the range of the allowed succeeding slots.

Price Source (OpenBook)

Next, price sources come from OpenBook, an order book protocol that will allow us to create our market after fractionalization and trade the fractionalized tokens. The function get_price() in Figure 5 returns the last-traded price in lots from OpenBook using its FillEvent state and Market state. As we previously saw, the function is called once every hour and returns a single number we can directly store in the ring PDA. We can see the detailed implementation of the function in Figure 7.

```
/// Returns the price (in lots) we want to store this hour.
/// Falls back to the previous stored price if no trade happened.
/// Only uses mid-price for the very first point.
fn get_price(
    heap: &EventHeap,    // OpenBook event heap
    prev: Option<i64>,   // price we stored 1 hour ago (None for 1st point)
    bids: &BookSide,     // bids account
    asks: &BookSide,     // ask account

) -> i64 {

    // 1. Get the last trade in the past hour
    if let Some(fill) = heap
        .iter()
        .rev()
        .find_map(|(ev, _)| match ev.event_type.try_into().ok()? {
            EventType::Fill => Some(bytemuck::cast_ref::<FillEvent>(ev)),
            _ => None,
        })
    {
        return fill.price; // already in lots => store as-is
    }

    // 2. No trade => reuse previous hour price
    if let Some(p) = prev {
        return p; // keeps TWAP
    }

    // 3. First hour and still no trade => mid-price fallback
    let bb = bids.best_price()?; // best bid in lots
    let ba = asks.best_price()?; // best ask in lots
    (bb + ba) / 2                // mid-price in lots
}
```

Fig. 7. Function that gets the prices from OpenBook.

As we can notice, prices are sourced from the OpenBook market through the following hierarchy:

1. Last traded price from the event heap.
2. Previous stored price if no trade occurred.
3. Mid-price between best bid and best ask if no historical data is available.

This approach allows us that in case there is no trade in a specific hour, no slot is skipped because the ring simply duplicates the previous hour's price, which correctly shows the market's real feeling.

Once the buyback is triggered, for reading the prices data in our PDA for TWAP, we can start at the next write index (ring.head + 1) and walk forward 720 steps, wrapping around %720. Because the ring is circular, this gives us the entries from the oldest to the freshest. Figure 8 shows the reading logic.

```
TWAP window = [start, end]

let start = (ring.head + 1) % 720;
for i in 0..720 {
    let entry = ring.points[(start + i) % 720];
    // entry is chronologically ordered:
    // i=0 => oldest (720 h ago)
    // i=719 => newest (1 h ago)
}
```

Fig. 8. Reading logic.

For a better understanding, let's see an example. Let's we have a small ring of 4 slots.

Table 2. Example of reading logic with a small ring.

Index	0	1	2	3
Data	D	A	B	C

And let's suppose the head is at position 2 (head=2). Then, the iteration would be:

```
let start = (ring.head + 1) % 4;   // = 3
for i in 0..4 {
    let entry = points[(start + i) % 4];
}

i=0 => (3+0)%4 = 3 => C   (oldest)
i=1 => (3+1)%4 = 0 => D
i=2 => (3+2)%4 = 1 => A
i=3 => (3+3)%4 = 2 => B   (newest)

TWAP window = [C, D, A, B]
```

Fig. 9. Iration logic example.

That means that C, D, A, B is the chronological sequence we would need for the window prices. Same idea applies with 720 elements.

price_window=[price_1,price_2,price_3,...,price_719,price_720]

After 30 days (720 hourly entries) from the market start, the reclaim becomes possible. Now, the rolling prices window written in our PDA allows that we are always looking at the recent prices performances from the moment the reclaim is done (30 days = 720 hours = 6480000 slots). It doesn't matter if the claim is done right after it's unlocked or 10 years later. It will always show fresh market data.

2.5 Decay-Weight Formula

The next step, is calculate the weights w_i of each price in our window using exponential decay (Epstein, 1996; Mazieri, 2022) with a half-life parameter of 24 hours (every 24 hours, the weight gets cut in half). For that, we can use the formula (1):

$$w = 2^{-t/24} \tag{1}$$

Where t is the i-index of the price in the prices window. Weighting the prices using (1) we get:

Table 3. Price weighting process.

$Price_i$	$w_i = 2^{-t/24}$	w_i
$Price_1$	$w_1 = 2^{(-1/24)}$	$w_1 = 0.97$
$Price_2$	$w_2 = 2^{(-2/24)}$	$w_2 = 0.94$
$Price_3$	$w_3 = 2^{(-3/24)}$	$w_3 = 0.91$
...	...	...
$Price_{718}$	$w_{718} = 2^{(-718/24)}$	$w_{718} = 0.0000000009867018967$
$Price_{719}$	$w_{719} = 2^{(-719/24)}$	$w_{719} = 0.0000000009586124091$
$Price_{720}$	$w_{720} = 2^{(-720/24)}$	$w_{720} = 0.0000000009313225746$

Finally, the final price per token P for the minority pay-out will be calculated as follows:

$$P = \frac{\sum_{i=1}^{720}(p_i * w_i)}{\sum_{i=1}^{720} w_i} \tag{2}$$

Where p_i correspond to each price in the prices window and $\rightarrow$ the respective weight. As we can notice in Table 3, the latest prices in the window have less weight, hence less influence in the total average which prevents from any latest huge pump/dump influence in the total price. The Figure 10 below shows the graph of the equation (1). Notice that with 24 hours half-life parameter, after the first 25 hours (first 25 prices), the weight drops to 0.5, making practically impossible profitable to any attacker to influence the price in last minutes/hours before the reclaim.

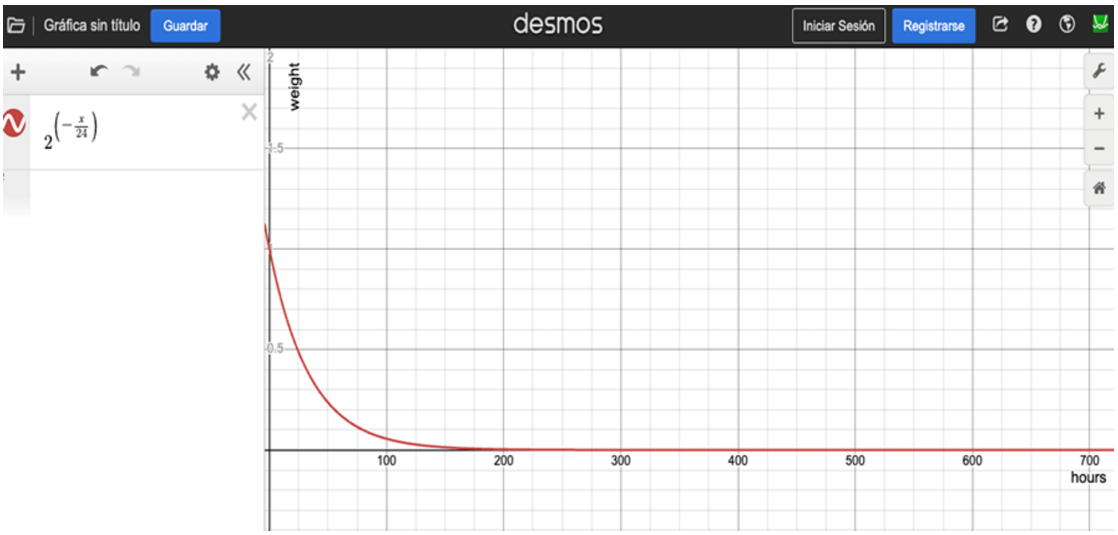


Fig. 10. Graph of equation (1).

Next, graphs in Figure 11 show that the smaller the half-life parameter, the faster the weight drops and the harder it becomes for an attacker to succeed.

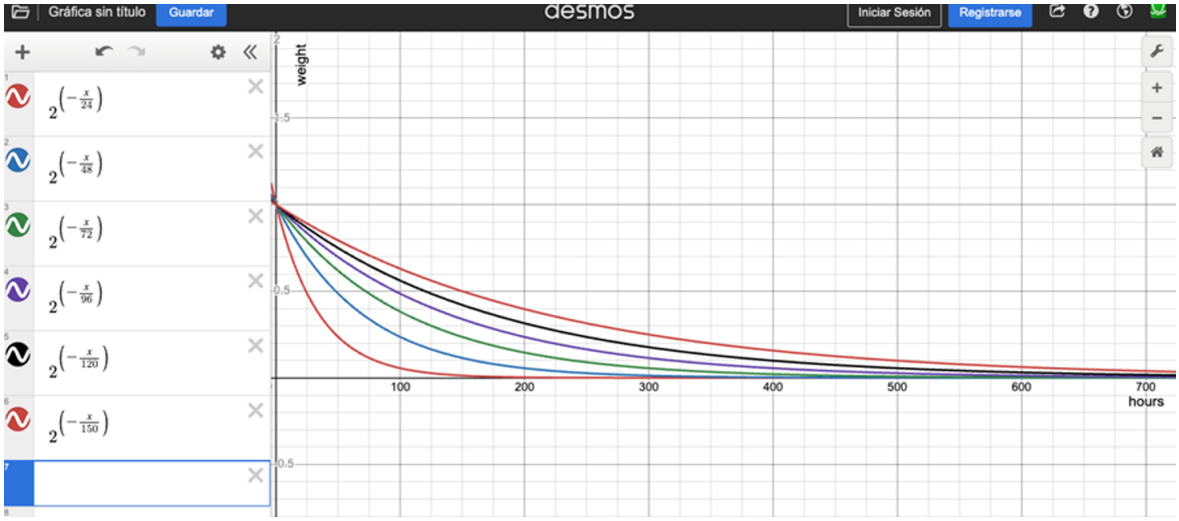


Fig. 11. Weights graphs for different half-life parameters.

Results and Discussion

Now, using the script in this repository, let us calculate, using the formula (2), the price per token in three different scenarios:

SCENARIO 1:

Let us assume that there is no attempt of price manipulation and the price per token in the 30-day window varies from $3.0 to $5.0 randomly. Figure 12 shows the simulation results:

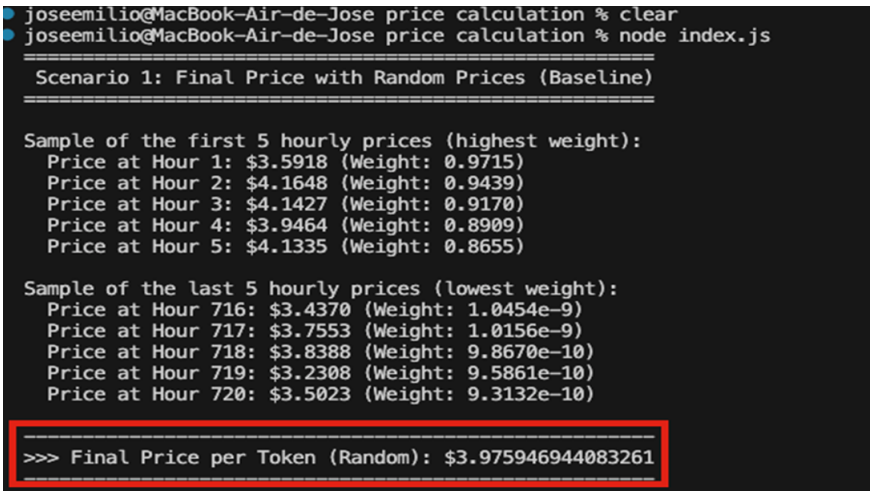


Fig. 12. Simulation results of scenario 1.

As we can notice in the figure above, the price in the first 5 hours after the start of the TWAP window have higher weights, and the prices in the last 5 hours have lower weights. Here, the final price per token is $3.975946944083261, reflecting the central tendency of the market.

SCENARIO 2:

The price per token has been moving from $3.0 to $5.0 randomly, but in the last 72 hours, the price has been intentionally pumped to $50.00 (pretty high compared to the range of $3.0-$5.0 that it has been moving in). The result is shown in Figure 13:

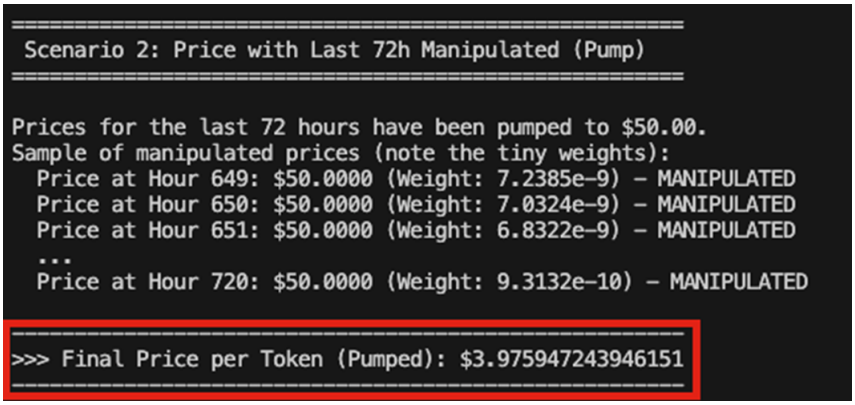


Fig. 13. Simulation results of scenario 2.

As we can notice, the extreme pump in the final 72 hours has a negligible effect on the final average price, so the final price per token is $3.975947243946151. This means that, even if the attacker intentionally pumps the prices by buying $3.00 - $5.00 tokens at $50 during the last 72 hours with the aim of having influence on the average price per token for when the reclaim is done, it has a completely negligible effect and at the same time is not profitable at all.

SCENARIO 3:

The price per token has been moving from $3.0 to $5.0 randomly, but in the last 72 hours, the price has been intentionally dumped to $0.10 (pretty low compared to the range of $3.0-$5.0 that it has been moving in). The result is shown in Figure 14:

Fig. 14. Simulation results of scenario 3.

Here, similar to the previous result, the extreme dump in the final 72 hours also has a negligible effect on the final average price, so the final price per token is $3.975946918635178.

These three scenarios confirm that exponential decay nullifies the effect of short-term anomalies while maintaining fairness for all participants.

Cost and Efficiency Analysis

— Storage: Each PDA requires 11.5 kB of space, equivalent to 0.081125 SOL for rent-exempt status. This cost is paid once and reclaimed when the account is closed.

— Transaction Costs: Each hourly update requires writing a 16-byte structure, costing about 5000 lamports [14].

— Performance: The circular buffer design ensures a constant time complexity (Cormen, 2009; Curley, 2022) for insertion and reading for TWAP calculation, which is computationally efficient on-chain.

Conclusion

The combination of a rolling 30-day window and exponential decay weighting provides a robust framework for reclaim pricing. Simulation confirms that manipulation attempts in the last 72 hours fail to meaningfully alter the outcome. Meanwhile, stale historical data is automatically excluded after 30 days, ensuring relevance.

This design balances efficiency and security. The PDA storage footprint is minimal, and the update cost is affordable. The reclaim mechanism guarantees fairness for minority holders while discouraging majority holders from attempting manipulative strategies.

1. A program-derived account (PDA) structured as a circular buffer efficiently stores 720 hourly prices, providing an on-chain dataset of 30 days with minimal memory footprint (~11.5 kB) and affordable rent cost (~0.081125 SOL).

2. The hourly update mechanism, combined with slot-gate validation, guarantees exactly one valid price record per hour. This prevents spam and manipulation, ensuring data integrity throughout the observation window.

3. The multi-source pricing method (last traded price, previous price reuse, and mid-price fallback) ensures that the system always produces a valid hourly price, even under conditions of low liquidity or no trading activity.

4. The application of exponential decay with a 24-hour half-life establishes a direct connection between time and price weight, making earlier prices significantly more influential than recent ones. This effectively neutralizes the impact of short-term volatility or last-minute manipulation attempts.

5. Simulated scenarios demonstrate that extreme price shocks (pump or dump within the last

72 hours) do not materially affect the final TWAP outcome, confirming the robustness of the weighting scheme.

6. The reclaim price, derived from the exponentially decay-weighted TWAP over the rolling 30-day window, reflects fair market value and ensures equitable treatment of minority holders during reclaim events.

7. The proposed method balances computational efficiency (O(1) insertion, O(720) retrieval) with economic feasibility (~5000 lamports per hourly update), proving its practical applicability in Solana-based fractionalization protocols.

REFERENCES

Chen Y., Bellavitis C. 2020. Blockchain disruption and decentralized finance: The rise of decentralized business models. Journal of Business Venturing Insights 13, 001512020. doi:10.1016/j.jbvi.2019.e00151

Cormen T., Leiserson C., Rivest R., Stein C. 2009. Introduction to algorithms. MIT Press.

Curley N. 2022. Blockchain disruption: digital assets are changing how we do business. SMU Science and Technology Law Review 25 (2), 265. doi:10.25172/smustlr.25.2.6

Epstein M., Mazzeo G. 1996. Exponential decay and its applications in economics and finance. Applied Mathematical Finance 3 (2), 119–134.

Faheem M., Kuusniemi H., Eltahawy B. 2024. A lightweight smart contracts framework for blockchain-based secure communication in smart grid applications. IET Generation, Transmission & Distribution 18 (3), 625-638. doi:10.1049/gtd2.13103

Fama E. 1965. The behavior of stock market prices. Journal of Business 38 (1), 34–105.

Lo A., MacKinlay A. 1999. A non-random walk down Wall Street. Princeton University Press.

Mazieri M., Scafuto I., Da Costa P. 2022. Tokenization, blockchain and web 3.0 technologies as research objects in innovation management. International Journal of Innovation 10 (1), 1-5. doi:10.5585/iji.v10i1.21768

Mishra D., Behera S. 2024. Solana blockchain technology: a review. International Journal of Informatics and Communication Technology 13 (2), 197. doi:10.11591/ijict.v13i2.pp197-205

Painter Z., Peterson Ch., Cook V., Dechev D. 2024. Blockchain scalability with proof of descriptor. Distributed Ledger Technologies: Research and Practice. doi:10.1145/3700148

Pezzella E., Pliushch E. G. 2022. Digital transformation of business: use of blockchain in the oil & gas industry. Technoeconomics 3 (3), 4-16. DOI: https://doi.org/10.57809/2022.3.3.1

CoinMarketCap Academy. Fractional NFTs: a complete guide to shared ownership of digital assets. URL: https://coinmarketcap.com/academy/article/what-are-fractional-nfts-f-nfts-and-how-do-they-work (accessed: 24.08.2025).

Decay-Weighted TWAP. GitHub Repository. URL: https://github.com/josephinney/decay-weighted-twap (accessed: 24.08.2025).

NFT Evening. What are Solana NFTs? Top Solana NFT projects in 2025. URL: https://nftevening.com/solana-nfts/ (accessed: 24.08.2025).

OpenBook Dex. Market State Module. GitHub Repository. URL: https://github.com/openbook-dex/openbook-v2/blob/master/programs/openbook-v2/src/state/market.rs (accessed: 24.08.2025).

OpenBook Dex. Orderbook Heap Module. GitHub Repository. URL: https://github.com/openbook-dex/openbook-v2/blob/master/programs/openbook-v2/src/state/orderbook/heap.rs (accessed: 24.08.2025).

Solana Foundation. Program Derived Addresses. Solana Documentation.URL: https://docs.solana.com/developing/programming-model/calling-between-programs#program-derived-addresses (accessed: 24.08.2025).

Solana Foundation. Rent-Exemption and Accounts. Solana Documentation. URL: https://docs.solana.com/developing/programming-model/accounts (accessed: 24.08.2025).

Solana Foundation. Slots and Time. Solana Documentation. URL: https://docs.solana.com/cluster/slots (accessed 24.08.2025).

Solana Foundation. Solana documentation: introduction to Solana. URL: https://docs.solana.com/ (accessed: 24.08.2025).

Solana Foundation. Transaction Fees. Solana Documentation. URL: https://docs.solana.com/transaction_fees (accessed: 24.08.2025).

СПИСОК ИСТОЧНИКОВ

Chen Y., Bellavitis C. 2020. Blockchain disruption and decentralized finance: The rise of decentralized business models. Journal of Business Venturing Insights 13, 001512020. doi:10.1016/j.jbvi.2019.e00151

Cormen T., Leiserson C., Rivest R., Stein C. 2009. Introduction to Algorithms. MIT Press.

Curley N. 2022. Blockchain Disruption: Digital assets are changing how we do business. SMU Science and Technology Law Review 25 (2), 265. doi:10.25172/smustlr.25.2.6

Epstein M., Mazzeo G. 1996. Exponential decay and its applications in economics and finance. Applied Mathematical Finance 3 (2), 119–134.

Faheem M., Kuusniemi H., Eltahawy B. 2024. A lightweight smart contracts framework for blockchain-based secure communication in smart grid applications. IET Generation, Transmission & Distribution 18 (3), 625-638. doi:10.1049/gtd2.13103

Fama E. 1965. The behavior of stock market prices. Journal of Business 38 (1), 34–105.

Lo A., MacKinlay A. 1999. A non-random walk down Wall Street. Princeton University Press.

Mazieri M., Scafuto I., Da Costa P. 2022. Tokenization, blockchain and web 3.0 technologies as research objects in innovation management. International Journal of Innovation 10 (1), 1-5. doi:10.5585/iji.v10i1.21768

Mishra D., Behera S. 2024. Solana blockchain technology: a review. International Journal of Informatics and Communication Technology 13 (2), 197. doi:10.11591/ijict.v13i2.pp197-205

Painter Z., Peterson Ch., Cook V., Dechev D. 2024. Blockchain scalability with proof of descriptor. Distributed Ledger Technologies: Research and Practice. doi:10.1145/3700148

Pezzella E., Pliushch E. G. 2022. Digital transformation of business: use of blockchain in the oil & gas industry. Technoeconomics 3 (3), 4-16. DOI: https://doi.org/10.57809/2022.3.3.1

CoinMarketCap Academy. Fractional NFTs: a complete guide to shared ownership of digital assets. URL: https://coinmarketcap.com/academy/article/what-are-fractional-nfts-f-nfts-and-how-do-they-work (дата обращения: 24.08.2025).

Decay-Weighted TWAP. GitHub Repository. URL: https://github.com/josephinney/decay-weighted-twap (дата обращения: 24.08.2025).

NFT Evening. What are Solana NFTs? Top Solana NFT projects in 2025. URL: https://nftevening.com/solana-nfts/ (дата обращения: 24.08.2025).

OpenBook Dex. Market State Module. GitHub Repository. URL: https://github.com/openbook-dex/openbook-v2/blob/master/programs/openbook-v2/src/state/market.rs (дата обращения: 24.08.2025).

OpenBook Dex. Orderbook Heap Module. GitHub Repository. URL: https://github.com/openbook-dex/openbook-v2/blob/master/programs/openbook-v2/src/state/orderbook/heap.rs (дата обращения: 24.08.2025).

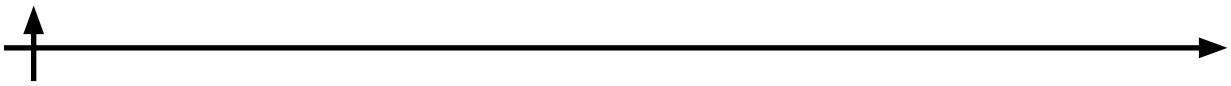
Solana Foundation. Program Derived Addresses. Solana Documentation.URL: https://docs.solana.com/developing/programming-model/calling-between-programs#program-derived-addresses (дата обращения: 24.08.2025).

Solana Foundation. Rent-Exemption and Accounts. Solana Documentation. URL: https://docs.solana.com/developing/programming-model/accounts (дата обращения: 24.08.2025).

Solana Foundation. Slots and Time. Solana Documentation. URL: https://docs.solana.com/cluster/slots (дата обращения: 24.08.2025).

Solana Foundation. Solana documentation: introduction to Solana. URL: https://docs.solana.com/ (дата обращения: 24.08.2025).

Solana Foundation. Transaction Fees. Solana Documentation. URL: https://docs.solana.com/transaction_fees (дата обращения: 24.08.2025).



INFORMATION ABOUT AUTHOR / ИНФОРМАЦИЯ ОБ АВТОРЕ

PHINNEY DOMINGUEZ Jose Emilio – student.
E-mail: josemilio.1995@gmail.com
ФИННИ ДОМИНГЕС Хосе Эмилио – студент.
E-mail: josemilio.1995@gmail.com
ORCID: https://orcid.org/0009-0005-0370-7313

Статья поступила в редакцию 03.09.2025; одобрена после рецензирования 09.09.2025; принята к публикации 11.09.2025.
The article was submitted 03.09.2025; approved after reviewing 09.09.2025; accepted for publication 11.09.2025.