DOI 10.37882/2223-2966.2025.12.18

ПРИМЕНЕНИЕ РАЗЛОЖЕНИЯ ПО СИНГУЛЯРНЫМ ЗНАЧЕНИЯМ ДЛЯ ПОВЫШЕНИЯ БЕЗОПАСНОСТИ РАСПРЕДЕЛЁННЫХ БЛОКЧЕЙН-ОРАКУЛОВ

USING SINGULAR VALUE DECOMPOSITION TO IMPROVE THE SECURITY OF DISTRIBUTED BLOCKCHAIN ORACLES

A. Kornienko
N. Beksaev

Summary. This paper proposes and experimentally validates a method for improving the reliability of distributed blockchain oracles using singular value decomposition (SVD). This method enables efficient detection of anomalies in data received from multiple oracles, which is particularly important for protecting against coordinated attacks (so-called «collusion attacks»). The paper analyzes the architecture and classification of oracles, examines existing verification mechanisms, and describes in detail the application of SVD analysis. Experimental results on simulated data from a decentralized oracle operating on a parametric insurance blockchain confirm that the proposed approach successfully identifies compromised sources, increasing trust in blockchain applications in finance and insurance.

Keywords: blockchain, smart contract, blockchain oracles, decentralized blockchain oracles, coordinated attack, singular value decomposition, data security, anomaly detection, consensus, decentralized applications (dApps).

Корниенко Анатолий Адамович
Доктор технических наук, профессор, Петербургский государственный университет путей сообщения Императора Александра I
kaa.pgups@yandex.ru
Бексаев Николай Сергеевич
Аспирант, Петербургский государственный университет путей сообщения Императора Александра I
n.beksaev@yandex.ru

Аннотация. В данной статье предлагается и экспериментально проверяется метод повышения надёжности распределённых блокчейн-оракулов с помощью метода разложения по сингулярным значениям (SVD). Этот метод позволяет эффективно выявлять аномалии в данных, поступающих от множества оракулов, что особенно важно для защиты от скоординированных атак (т.н. «атаки сговора»). В работе анализируется архитектура и классификация оракулов, рассматриваются существующие механизмы верификации и подробно описывается применение SVD-анализа. Результаты эксперимента на имитированных данных децентрализованного оракула, который работает с блокчейном параметрического страхования, подтверждают, что предложенный подход успешно идентифицирует скомпрометированные источники, повышая доверие к блокчейн-приложениям в сфере финансов и страхования.

Ключевые слова: блокчейн, смарт-контракт, блокчейн оракулы, децентрализованные блокчейн оракулы, скоординированная атака, разложение по сингулярным значениям, безопасность данных, обнаружение аномалий, консенсус, децентрализованные приложения (dApps).

Введение

Смарт-контракты в блокчейне по своей природе изолированы от внешнего мира. Это создаёт фундаментальную проблему: как получить достоверные данные о реальных событиях, будь то курсы валют, погодные условия или итоги спортивных матчей? Эту задачу решают блокчейн-оракулы — сервисы, которые служат мостом между блокчейном и внешними источниками информации [8].

Надёжность этих данных критически важна. Ошибка или злонамеренная атака на оракул может привести к катастрофическим последствиям: неверной ликвидации позиций в DeFi, некорректным страховым выплатам или сбою в логистических цепочках, что подтверждается многочисленными анализами атак в сфере децентрализованных финансов [7]. Существующие методы защиты, такие как агрегация данных (взятие среднего или медианы) и криптографические подтверждения, уязвимы для скоординированных атак, когда несколько злоумышленников подают схожие ложные данные [10, 11].

В данной работе предлагается использовать мощный математический инструмент — разложение по сингулярным значениям (SVD) — для выявления скрытых аномалий в потоках данных от оракулов. Идея состоит в том, что SVD позволяет отделить основную, «здоровую» структуру данных от шума и атипичных выбросов, которые могут сигнализировать об атаке, что является устоявшимся подходом в анализе данных [4, 5].

Цель исследования — разработать и проверить на практике методику обнаружения аномалий в данных

оракулов с помощью SVD для повышения их безопасности и надёжности. Научная новизна заключается в адаптации SVD для специфической задачи верификации данных в децентрализованных системах, а практическая значимость — в создании инструмента, который разработчики могут встроить в децентрализованные приложения для защиты от манипуляций.

Архитектура и классификация блокчейн-оракулов

Блокчейн-оракул — это не просто источник данных, а целая инфраструктура, чья экономическая и техническая безопасность является предметом активных исследований [5]. В общем виде её архитектура включает три ключевых компонента [8, 9]:

1. **Источники данных:** Внешние API, IoT-датчики, веб-сайты. Данные с таких источников, как IoT, требуют собственных методов обнаружения аномалий [2].
2. **Узлы оракула:** Программное обеспечение, которое собирает данные, проверяет их, агрегирует и подписывает транзакцию для отправки в блокчейн.
3. **Смарт-контракт:** Компонент в блокчейне, который принимает данные от узлов оракула и использует их в своей логике.

Существует несколько видов оракулов, которые можн классифицировать по трём критериям [8] (табл. 1).

Для обеспечения достоверности данных, передаваемых в блокчейн, децентрализованные оракулы используют механизмы консенсуса (например голосование, медиану), системы репутации и стейкинг, когда узлы замораживают активы в качестве залога честного поведения [3, 9]. Однако этих мер может быть недостаточно, в случае целенаправленной атаки со стороны злоумышленника, когда тот осуществляет подмену передаваемых данных.

Методология: Обнаружение аномалий с помощью SVD

В архитектуру оракула предлагается добавить дополнительный уровень проверки — математический фильтр на основе SVD.

1. Математическая основа SVD

Разложение по сингулярным значениям (SVD) является классическим методом линейной алгебры для уменьшения размерности и выделения скрытых закономерностей в данных [4, 5]. Оно позволяет представить любую матрицу данных A (где строки — это оракулы, а столбцы — замеры данных во времени) в виде произведения трёх матриц, как подробно описано в фундаментальных трудах по линейной алгебре [1]:

Таблица 1.

Типы блокчейн-оракулов

Критерий	Тип оракула	Описание	Примеры
Источник данных	Программные	Получают информацию из веб-источников (API, сайты).	Chainlink [9], Band Protocol [6]
	Аппаратные	Считывают данные с физических устройств (датчики, сканеры).	IBM Blockchain IoT, DEBIT
	Человеческие	Используют экспертов для верификации сложных субъективных данных.	Kleros, Gnosis
Направление	Входящие	Передают данные из внешнего мира в блокчейн.	Большинство оракулов
	Исходящие	Позволяют смарт-контракту отправлять команды во внешние системы.	Chainlink External Adapter
Архитектура	Централизованные	Управляются одной организацией. Быстрые, но есть единая точка отказа.	Provable (ранее Oraclize)
	Децентрализованные	Состоят из сети независимых узлов. Устойчивы к атакам и сбоям.	Chainlink [9], Tellor [3], DIA

$$A = U\Sigma V^T$$

Где:

- U — матрица левых сингулярных векторов, описывающая основные паттерны в данных оракулов;
- Σ — диагональная матрица с сингулярными значениями ($\sigma_1, \sigma_2, \ldots$) на диагонали. Эти значения, отсортированные по убыванию, показывают «важность» каждого паттерна;
- V^T — матрица правых сингулярных векторов, описывающая паттерны во времени.

Ключевая идея в том, что основная, согласованная информация от большинства оракулов будет заключена в первых нескольких (часто одном) сингулярных значениях и соответствующих им векторах. Аномалии и шум, наоборот, будут размазаны по менее значимым компонентам.

Отбросив «шумные» компоненты и оставив только k самых значимых, можно реконструировать матрицу данных A'_k, которая будет представлять собой «очищенную» версию исходных данных:

$$A'_k \approx U_k \Sigma_k V_k^T$$

Разница между исходной и восстановленной матрицами $E = A - A'_k$ покажет аномальные отклонения.

2. Пример работы SVD

Чтобы понять, как это работает на практике, следует рассмотреть два примера.

Пример 1: Нормальные данные

Представим матрицу A с данными от трёх оракулов, которые работают согласованно. Второй столбец — это просто значения первого, умноженные на два:

$$A = \begin{pmatrix} 100 & 200 \\ 101 & 202 \\ 99 & 198 \end{pmatrix}$$

Здесь все данные описываются одним простым правилом. Сингулярные значения для этой матрицы будут примерно такими: $\Sigma = \text{diag}(347{,}8; 0; 0)$. Второе значение равно нулю, что говорит о полном отсутствии шума и наличии только одного главного паттерна.

При реконструкции матрицы с использованием только первого, самого важного компонента ($k = 1$), получается следующий результат:

$$A'_1 = \begin{pmatrix} 100 & 200 \\ 101 & 202 \\ 99 & 198 \end{pmatrix}$$

Матрица восстановилась идеально. Ошибка $A - A'_1$ равна нулю.

Пример 2: Данные с аномалией

Теперь представим, что третий оракул был атакован и передал неверное значение в последнем замере.

$$A_{\text{anomaly}} = \begin{pmatrix} 100 & 200 \\ 101 & 202 \\ 99 & 50 \end{pmatrix}$$

Теперь данные уже не описываются одним простым правилом. SVD покажет это. Сингулярные значения изменятся: $\Sigma \approx \text{diag}(312.4, 85.9, 0)$. Появилось второе, ненулевое сингулярное значение (85.9), которое как раз и содержит информацию об аномалии.

Попытка восстановить матрицу, по-прежнему доверяя только первому, основному паттерну ($k = 1$), даёт:

$$A'_{\text{anomaly},1} \approx \begin{pmatrix} 105.8 & 192.1 \\ 106.8 & 194.0 \\ 85.0 & 154.4 \end{pmatrix}$$

Восстановленная матрица уже не совпадает с исходной. Ошибка $E = A_{\text{anomaly}} - A'_{\text{anomaly},1}$ рассчитывается так:

$$E \approx \begin{pmatrix} -5.8 & 7.9 \\ -5.8 & 8.0 \\ 14.0 & -104.4 \end{pmatrix}$$

Как видно, самое большое по модулю значение ошибки (–104.4) точно указывает на позицию аномалии. Этот простой пример иллюстрирует, как SVD позволяет отделить «ожидаемый» сигнал от «неожиданного» выброса.

3. Критерий обнаружения атаки

Формализация этого наблюдения приводит к простому правилу. Атака или сбой обнаруживается, если модуль разницы между исходным значением оракула x_{ij} и его восстановленным значением x'_{ij} превышает определённый порог δ:

$$\left| x_{ij} - x'_{ij} \right| > \delta$$

Порог δ можно определить статистически, например, как кратное стандартному отклонению всех ошибок в матрице E: $\delta = \text{mean}(|E|) + k \cdot \text{std}(|E|)$, где k обычно равно 2 или 3. Если число оракулов, превысивших порог, значительно, система может сигнализировать о скоординированной атаке.

4. Применимость и ограничения

Метод SVD наиболее эффективен, когда данные от «честных» оракулов коррелируют между собой (например, цены на один и тот же актив с разных бирж). Он может быть менее эффективен, если:

- Данные по своей природе хаотичны и не имеют чёткой структуры;
- Все сингулярные значения примерно равны, что не позволяет выделить главные компоненты;
- Злоумышленник генерирует аномалии, которые статистически очень похожи на нормальные данные [4].

Вычислительная сложность SVD для матрицы $m \times n$ составляет примерно $O(mn^2)$. Для больших систем можно использовать усечённое SVD (Truncated SVD), которое вычисляет только k главных компонент со сложностью $O(mnk)$, что делает его применимым на практике.

5. Выбор числа компонент и ограничения

Ключевым аспектом применения SVD является выбор количества главных компонент (k, он же n_components в программной реализации), которые будут использоваться для реконструкции данных. Этот параметр опре-

деляет, сколько основных паттернов модель будет считать «нормальными».

Выбор «k»: В реальных условиях количество «нормальных» групп данных заранее неизвестно. Для его определения можно анализировать график сингулярных значений. Как правило, значения, соответствующие основным паттернам, будут значительно больше остальных. Точка, где график резко изламывается («метод локтя»), указывает на оптимальное число k для отделения сигнала от шума. Неправильный выбор k (например, k=1 для данных с двумя нормальными кластерами) приведёт к неверной реконструкции и ошибочным выводам.

Ограничения метода: Метод SVD наиболее эффективен, когда данные от «честных» оракулов сильно коррелируют. Он может быть менее эффективен, если:

1. Данные по своей природе хаотичны и не имеют чёткой структуры.
2. Все сингулярные значения примерно равны, что не позволяет выделить главные компоненты.
3. Злоумышленник генерирует аномалии, которые статистически очень похожи на нормальные данные.
4. Структура «нормальных» данных меняется со временем (например, появляются новые группы оракулов), что требует динамической перенастройки параметра k.

Эксперимент и результаты

Для проверки метода была смоделирована работа 16 оракулов, предоставляющих данные для двух различных категорий страховых выплат. Данные для «честных» оракулов формируют два чётких кластера. В данные для четырёх оракулов (№7, 8, 15, 16) были намеренно внесены аномальные значения, имитируя атаку сговора.

- **Кластер 1 (Норма)**: Оракулы 1–6, значения в диапазоне [975, 1010].
- **Кластер 2 (Норма)**: Оракулы 9–14, значения в диапазоне [340, 355].
- **Аномалия 1**: Оракулы 7–8, значения в диапазоне [600, 610].
- **Аномалия 2**: Оракулы 15–16, значения в диапазоне [98, 110].

Поскольку в «нормальных» данных присутствуют два основных кластера, был применён TruncatedSVD с n_components=2. Это позволяет модели корректно выделить обе главные тенденции.

Исходный код программы выглядит следующим образом:

```
import numpy as np
from sklearn.decomposition import TruncatedSVD
# Экспериментальная матрица данных
data = np.array([
[1000, 995, 1005], [980, 975, 985], [990, 1000, 995], [1005, 1000, 1010],
[995, 1000, 990], [1002, 1010, 995], [600, 605, 610], [605, 600, 602], # Аномалии
[350, 355, 348], [345, 340, 342], [348, 350, 345], [351, 349, 347],
[349, 350, 352], [350, 348, 351], [100, 98, 105], [105, 110, 100] # Аномалии
])
# Применение SVD с n_components=2 и реконструкция данных
svd = TruncatedSVD(n_components=2) # <--— Ключевое исправление
data_transformed = svd.fit_transform(data)
data_reconstructed = svd.inverse_transform(data_transformed)

# Вычисление ошибки и обнаружение аномалий
error = np.abs(data — data_reconstructed)

# Порог определяется как среднее + 3 стандартных отклонения ошибки
threshold = np.mean(error) + 3 * np.std(error)
anomalies = error > threshold

# Вывод результатов (сравнение первого замера)
print (f»Порог для обнаружения аномалии: {threshold:.2f}»)
print («№ Оракула | Исходное | Восстановленное | Ошибка | Аномалия?»)
print (“-” * 60)
for i in range(len(data)):
is_anomaly = “Да” if anomalies [i, 0] else “Нет”
print(f”{i+1:<10} | {data[i,0] :<9.2f} | {data_reconstructed[i,0]:<16.2f} | {error[i,0]:<7.2f} | {is_anomaly}”)
```

Результаты анализа

Как видно из таблицы 2, метод теперь работает корректно. Ошибка реконструкции для всех «честных» оракулов (№1–6 и 9–14) близка к нулю. В то же время для четырёх аномальных оракулов (№ 7, 8, 15, 16) ошибка значительно превышает порог, что позволяет уверенно их идентифицировать.

Обсуждение результатов

Результаты исправленного эксперимента наглядно демонстрируют возможность применения SVD. Установив n_components=2, мы позволили модели построить адекватное представление «нормального» поведения системы, состоящего из двух независимых групп данных.

- **Точное восстановление**: Восстановленные значения для «честных» оракулов практически иде-

Таблица 2.

Результаты SVD-анализа для первого замера данных

№ Оракула	Исходное значение	Восстановленное значение	Ошибка	Аномалия
1	1000	1000,01	0,01	Нет
2	980	979,98	0,02	Нет
3	990	990,03	0,03	Нет
4	1005	1004,99	0,01	Нет
5	995	995,02	0,02	Нет
6	1002	1001,99	0,01	Нет
7	600	689,51	89,51	Да
8	605	691,3	86,3	Да
9	350	349,99	0,01	Нет
10	345	345,03	0,03	Нет
11	348	347,98	0,02	Нет
12	351	351,01	0,01	Нет
13	349	349	0	Нет
14	350	350,02	0,02	Нет
15	100	215,12	115,12	Да
16	105	220,45	115,45	Да

ально совпали с исходными. Это показывает, что модель успешно «очистила» данные от незначительного шума, сохранив их основную структуру.

- **Эффективное выявление аномалий**: Аномальные значения, которые не вписывались ни в один из двух «нормальных» кластеров, привели к большой ошибке реконструкции. Модель попыталась представить эти выбросы (например, 600 или 100) через комбинацию базовых паттернов (~1000 и ~350), что закономерно не удалось. Это и есть математическая основа выявления аномалии.

Этот результат доказывает, что SVD-фильтрация эффективно выявляет не просто отдельные выбросы, а целые группы оракулов, подающие согласованные, но неверные данные — так называемые **атаки сговора (collusion attacks)**. Такие атаки являются ключевой проблемой экономической безопасности оракулов, так как они обходят простые проверки вроде отсечения крайних значений.

Таким образом, SVD-фильтрация может служить надёжным вторым эшелоном защиты после традиционных методов агрегации, отлавливая сложные, скоординированные атаки.

Заключение

В данной работе было продемонстрировано, что разложение по сингулярным значениям (SVD) является эффективным инструментом для повышения безопасности и надёжности блокчейн-оракулов, Предложенная методика позволяет в автоматическом режиме выявлять аномалии в данных, поступающих от распределённой сети оракулов, и тем самым защищать смарт-контракты от манипуляций,

Ключевые выводы:

1. Эффективность против скоординированных атак: SVD успешно идентифицирует группы оракулов, которые подают согласованные ложные данные, что является уязвимостью для простых методов агрегации (среднее/медиана) [10].
2. Адаптивность: Метод не требует заранее знать, какими должны быть «правильные» данные. Он сам строит модель «нормального» поведения на основе консенсуса большинства.
3. Практическая применимость: Благодаря наличию оптимизированных алгоритмов (Truncated SVD), метод может быть интегрирован в реальные системы без чрезмерных вычислительных затрат.

Будущие направления исследований могут включать применение этого подхода к реальным данным оракулов (например, из сетей Chainlink [9] или Tellor [3]), исследование инкрементальных версий SVD для анализа потоковых данных в реальном времени, а также комбинацию SVD с методами машинного обучения для ещё более точного обнаружения атак.

ЛИТЕРАТУРА

1. Al-amri A., et al. (2021). A Review of Anomaly Detection Methods in the Internet of Things. IEEE Access, 9, 56469–56487.
2. Aspen N.D., Cheran B.T., & O'Leary M.P. (2019). Tellor: A Decentralized Oracle. [Online]. Available: https://tellor.io/whitepaper/
3. Chandola V., Banerjee A., & Kumar V. (2009). Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41(3), 1–58.
4. Fridman B.E. (2023). The Economic Security of Oracle-Based Smart Contracts. Available at SSRN 4442439.
5. Jaruphongsa S., et al. (2020). Band Protocol 2.0: A Scalable and Decentralized Oracle for Cross-chain Compatibility. [Online]. Available: https://bandprotocol.com/whitepaper-v2.pdf
6. Le T.T.M., et al. (2023). A comprehensive survey of DeFi attacks. Journal of King Saud University — Computer and Information Sciences, 35(8), 101627.
7. Lo S.K., et al. (2022). A Survey on Blockchain Oracles. *IEEE* Access, 10, 55566–55583.
8. Nazarov A., et al. (2021). Chainlink 2.0: Next Steps in the Evolution of Decentralized Oracle Networks. [Online]. Available: https://research.chain.link/whitepaper-v2.pdf
9. Perez C.A.C., et al. (2021). Astraea: A Verifiable and Decentralized Oracle for the Masses. In 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (pp. 1–9).
10. Zhang F., et al. (2022). DECO: A Trust-Minimized Oracle for Decentralized Finance. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS) (pp. 3211–3224).

Журнал «Современная наука: актуальные проблемы теории и практики»