

Научная статья

DOI 10.66424/2071-8217-2026-3-14

УДК 004.056

АРХИТЕКТУРА РАСПРЕДЕЛЕННОГО РЕЕСТРА НА БАЗЕ НАПРАВЛЕННОГО АЦИКЛИЧЕСКОГО ГРАФА С БЕЗОПАСНОЙ ФИНАЛИЗАЦИЕЙ ЗАПИСЕЙ И ОГРАНИЧЕННЫМ ПОТРЕБЛЕНИЕМ ПАМЯТИ

М. О. Калинин*, Р. А. Гартвиг

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

✉ *max@ibks.spbstu.ru

ДЛЯ ЦИТИРОВАНИЯ

Калинин М. О., Гартвиг Р. А.
Архитектура распределенного
реестра на базе направленного
ациклического графа
с безопасной финализацией
записей и ограниченным
потреблением памяти // Проблемы
информационной безопасности.
Компьютерные системы.
2026. № 3. С. 190–203.
DOI: 10.66424/2071-8217-2026-3-14

ПОСТУПИЛА 06.06.2026

ПРИНЯТА 03.08.2026

ОПУБЛИКОВАНА 13.10.2026

© Калинин М. О., Гартвиг Р. А.

Издатель: Санкт-Петербургский
политехнический университет
Петра Великого

АННОТАЦИЯ

Представлена новая архитектура распределенного реестра на основе направленного ациклического графа, предназначенная для гетерогенных IoT-сред с большим числом малоресурсных узлов. В отличие от классических блокчейн-решений и существующих DAG-реестров, предложенное решение сочетает параллельную фиксацию транзакций внутри эпох, BFT-финализацию состояния и механизм ротации генезис-состояния, позволяющий безопасно удалять историю. Экспериментальная оценка показала, что разработанный подход позволяет сократить объем хранимых данных в пять раз по сравнению с базовой DAG-архитектурой без сворачивания, а при конфликтной нагрузке сохраняет до 80 % честных транзакций, что подтверждает эффективность предложенной архитектуры для применения в распределенных реестрах умных городов и киберфизических систем.

КЛЮЧЕВЫЕ СЛОВА

Безопасность, направленный ациклический граф, распределенный реестр, генезис-блок, сокращение истории, финализация

ФИНАНСИРОВАНИЕ

Исследование выполнено за счет гранта Российского научного фонда № 24-11-20005, <https://rscf.ru/project/24-11-20005/>, грант Санкт-Петербургского научного фонда (договор № 24-11-20005 о предоставлении регионального гранта).

Original article

DOI 10.66424/2071-8217-2026-3-14

DISTRIBUTED LEDGER ARCHITECTURE BASED ON A DIRECTED ACYCLIC GRAPH WITH SAFE RECORD FINALIZATION AND LIMITED MEMORY CONSUMPTION

M. O. Kalinin*, R. A. Gartvig

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

✉ *max@ibks.spbstu.ru

FOR CITATION

Kalinin M. O., Gartvig R. A.
Distributed ledger architecture
based on a directed acyclic graph with
safe record finalization and limited
memory consumption. *Problems of
information security. Computer systems.*
2026. No. 3, pp. 190–203.
DOI: 10.66424/2071-8217-2026-3-14
(In Russian)

RECEIVED 06.06.2026

ACCEPTED 03.08.2026

PUBLICATION 13.10.2026

ABSTRACT

A new distributed ledger architecture based on a directed acyclic graph (DAG) is presented, designed for heterogeneous IoT environments with a large number of resource-limited nodes. Unlike classic blockchains and existing DAG ledgers, the proposed solution combines parallel transaction commits within epochs, BFT state finality, and a genesis state rotation mechanism that enables secure deletion of ledger's history. Experimental evaluation showed that the developed approach reduces the amount of stored data by five times compared to the genuine DAG architecture without rollup, and preserves up to 80 % of pure transactions under conflict load, confirming the effectiveness of the proposed architecture for use in distributed ledgers for smart cities and cyber-physical systems.

KEYWORDS

Security, directed acyclic graph, distributed ledger, genesis block, history pruning, finality

FUNDING

The study was supported by the grant of Russian Science Foundation No. 24-11-20005, <https://rscf.ru/project/24-11-20005/>; grant of St. Petersburg Science Foundation (agreement No. 24-11-20005 on the regional grant).

1. ВВЕДЕНИЕ

Развитие систем Интернета вещей, умных зданий и умных городов приводит к появлению распределенных киберсред, в которых большое число связанных мобильных устройств непрерывно создает, передает и обрабатывает данные. К таким данным относятся телеметрия от датчиков, сообщения транспортной инфраструктуры, уведомления о состоянии оборудования и другие транзакции, требующие проверки и согласованного хранения. Для подобных систем важны не только целостность данных и защита от подмены, но и низкая задержка подтверждения, устойчивость к сетевым сбоям и умеренные требования к ресурсам участников сети. Классические системы распределенного реестра плохо соответствуют таким требованиям. Линейная структура блоков ограничивает параллелизм обработки, а хранение постоянно растущей истории реестра увеличивает требования к памяти и дисковому пространству на узлах сети. Для IoT-среды это особенно критично, поскольку многие участники обладают

ограниченными вычислительными ресурсами, нестабильным соединением и не могут бесконечно хранить всю историю реестра [1]. Поэтому в качестве альтернативы все чаще предлагается использовать альтернативную архитектуру реестра, где транзакции образуют направленный ациклический граф (DAG-реестр, Directed Acyclic Graph), в котором новые объекты могут ссылаться сразу на несколько предыдущих вершин [2].

DAG-подход позволяет естественным образом поддерживать параллельное добавление транзакций и уменьшать зависимость от последовательного формирования блоков. Однако вместе с этим возникает другая проблема: каждая новая транзакция должна выбрать родителей, т.е. те вершины графа, на которые она будет ссылаться. Этот выбор влияет на то, какая часть DAG получит поддержку, какие транзакции приблизятся к подтверждению, а какие останутся без достаточного веса. В реальной P2P-сети задача усложняется тем, что узел не «видит» глобальный граф, а принимает решение только на основе локально доставленных транзакций [3].

Поэтому разные участники сети могут временно видеть по-разному DAG-реестр, по-разному узнавать о конфликтах и принимать разные локальные решения. Отдельной проблемой является неконтролируемый рост объема хранения графа. Если узел обязан хранить все транзакции с момента запуска системы, то даже эффективный механизм подтверждения со временем становится непригодным для малоресурсных узлов сети.

Предложено комплексное решение указанных проблем, обеспечивающее уменьшение объема памяти для структуры DAG-реестра с сохранением безопасной финализации записей в условиях воздействия конфликтной нагрузки.

2. АНАЛИЗ АРХИТЕКТУР DAG-РЕЕСТРОВ

Представленный анализ существующих DAG-реестров преследует цель систематизировать свойства, значимые для новой архитектуры, включая возможность параллельной фиксации транзакций, требования к объему хранимых данных, механизмы подтверждения и разрешения конфликтов, а также применимость в гетерогенных P2P-средах малоресурсных устройств.

Основу известного реестра IOTA [4] составляет DAG-структура Tangle, в котором новые блоки ссылаются не на один, а на несколько предыдущих блоков, благодаря чему подтверждение транзакций достигается через накопление ссылок и голосов в графе, а не через линейное добавление в цепочку. Консенсусный механизм представляет собой адаптацию традиционного консенсуса Накамото, адаптированного для DAG-структуры. Узлы голосуют за допустимую часть реестра, а валидаторы придают этим голосам вес с помощью специальных подтверждающих блоков. При этом каждый новый блок выражает «мнение» своего создателя о корректной части графа, а подтверждающие блоки обеспечивают более безопасное и эффективное голосование [5]. Структура блока включает информацию о выпустившем

узле, полезную нагрузку, набор родительских блоков, временной слот, ссылку на состояние слота и цифровую подпись. Так называемый *tip* – это блок, который еще не получил достаточного числа последующих ссылок. При добавлении нового блока узел выбирает несколько таких *tip*-блоков и ссылается на них, тем самым неявно подтверждая не только непосредственных родителей, но и всю их предшествующую историю. В каждой временной эпохе формируется комитет валидаторов с нормированными весами, которые выпускают подтверждающие блоки, служащие головами за определенную часть реестра.

Подтверждающий вес блока вычисляется как сумма весов валидаторов, ссылающихся на данный блок, и блок считается принятым, если этот вес превышает порог $2/3$. При возникновении конфликта, например, при двойной трате, предпочтение отдается той транзакции, которая получила больший вес от валидаторов и достигла порога. Дополнительно в системе применяются коммиты слотов: для каждого временного слота формируется коммит, фиксирующий его содержимое и ссылающийся на предыдущий коммит, что порождает линейную цепочку, финализируемую при достижении необходимого порога веса [6]. Если возникает несколько конкурирующих цепочек коммитов, узел выбирает ту, которая обладает наибольшим накопленным весом (по аналогии с правилом самой «тяжелой» цепочки). Таким образом, преимущества архитектуры IOTA заключаются в параллельном добавлении объектов и разделении пользовательских транзакций и процесса валидации, причем ключевыми концепциями выступают подтверждение через ссылки на предыдущие области графа и накопление валидаторного веса. Вместе с тем очевидны ограничения данной системы, связанные с неограниченным ростом активной истории реестра и зависимостью надежности подтверждения от выбора родителей участниками сети, что будет учтено в новой DAG-архитектуре.

Система Hashgraph [7] представляет собой DAG-реестр, в котором вершинами являются не блоки, а события. Каждое

событие фиксирует факт обмена информацией между участниками сети. Таким образом, Hashgraph хранит не только сами транзакции, но еще и трассу распространения информации в сети (протокол gossip about gossip). Это позволяет всем честным узлам локально восстановить единый граф коммуникаций и выполнить виртуальное голосование без отправки отдельных голосующих сообщений.

В системе задано множество участников консенсуса, каждый из которых может создавать события, принимать события от других участников и распространять известную ему часть графа. Hashgraph относится к классу BFT-протоколов [8] и предполагает, что более $2/3$ участников являются честными. Событие содержит информацию о создателе, набор включенных транзакций, ссылку на предыдущее событие того же создателя, ссылку на последнее событие другого участника, полученное при синхронизации, а также временную метку и подпись.

При этом каждое событие имеет двух родителей. Вводятся понятия предка и self-предка. Предком события называется любое событие, достижимое из него по родительским ссылкам. Self-предком называется событие, достижимое только по self-parent ссылкам, что отражает личную историю одного участника. Особым случаем является вилка, когда один участник создает две несовместимые ветви своей истории. Для обработки таких ситуаций в Hashgraph используются определения видимого и сильно видимого события. Событие «видит» другое, если оно знает о нем и не знает противоречащей вилки того же автора. Событие «сильно видит» другое, если оно может видеть более чем $2/3$ событий, созданных разными участниками, каждое из которых видит исходное событие. Именно это свойство заменяет явную рассылку голосов.

Каждому событию назначается номер раунда. Свидетель-событием называется первое событие своего создателя в данном раунде. Именно такие события участвуют в виртуальных выборах и используются для вычисления финального порядка [9]. Известным свидетелем называется сви-

детель, который по результатам виртуального голосования признается быстро увиденным значительной частью участников. Голосование начинается с того, что свидетель более позднего раунда голосует за более раннего свидетеля. Если он его «видит», голос считается положительным, в противном случае – отрицательным. Эти голоса не передаются по сети, а вычисляются каждым узлом локально на основе графа.

В каждом следующем раунде каждый свидетель смотрит на свидетелей предыдущего раунда, которых он «сильно видит». Затем он подсчитывает, сколько среди них проголосовало за и против рассматриваемого кандидата. Если количество голосов за или против превышает две трети от общего числа участников, решение считается окончательным. Если порог $2/3$ не достигнут, свидетель формирует свой промежуточный голос на основе простого большинства, и процесс продолжается в следующих раундах. Голосование завершается, когда некоторый свидетель обнаруживает $2/3$ голосов в ту или иную сторону.

После того как известные свидетели определены, для каждого события вычисляется раунд подтверждения – первый раунд, в котором все известные свидетели «видят» данное событие. Затем для события определяется консенсусная временная метка как медиана времен создания тех известных свидетелей, которые уже знают о нем. Наконец, все события упорядочиваются лексикографически: сначала по раунду подтверждения, затем по консенсусной временной метке, а при совпадении – по хешу события.

Hashgraph демонстрирует, что граф может использоваться не только для хранения транзакций, но и для восстановления того, кто какую информацию «видел» в момент принятия решения. Для представленной разработки это важно, поскольку новое решение также учитывает локальность знания узлов. Однако Hashgraph ориентирован на фиксированное множество участников консенсуса и требует интенсивного обмена служебной информацией, поэтому Hashgraph

не подходит в качестве базовой архитектуры для IoT-среды с большим числом узлов, но полезен как демонстрация того, насколько сильно консенсус зависит от пространства информации по сети.

PHANTOM и другие BlockDAG-архитектуры [10] расширяют концепцию консенсуса Накамото, заменяя линейную цепочку блоков на граф. Это позволяет обрабатывать параллельно созданные блоки и тем самым увеличивает пропускную способность системы. Ключевым моментом в таких моделях является упорядочивание блоков и разделение графа на честную область и область, которая может быть сформирована атакующим. Такое разделение обеспечивает защиту от потенциальных атак. BlockDAG-подходы важны как пример того, что увеличение параллелизма неизбежно требует дополнительных правил выбора и упорядочивания. Однако подобные системы чаще ориентированы на блоки и майнинг, а не на легкие транзакции в среде IoT. Кроме того, проблема хранения постоянно увеличивающейся истории графа для них также актуальна, поэтому PHANTOM и все BlockDAG-подходы не могут быть применены в качестве основы для разрабатываемого оптимального решения, но они подтверждают общий вывод, что в DAG-системах структура ссылок сама по себе может стать базой для механизма обеспечения безопасности распределенного реестра.

Архитектура Avalanche [11] использует семейство протоколов, основанных на повторяющемся случайном опросе участников и накоплении предпочтения в пользу одной из конфликтующих транзакций. В ряде реализаций данные могут организовываться в виде направленного ациклического графа, однако ключевая роль принадлежит вероятностному процессу голосования и сходимости предпочтений. В рамках проводимого исследования подход Avalanche полезен как пример быстрого подтверждения и разрешения конфликтов, но его логика обладает рядом недостатков. В Avalanche нет локального выбора родителей и подтверждающих блоков, поэтому эта архитектура рассматривается как альтернативный подход к быстрому консенсусу, но не как основа для создаваемой модели.

Сопоставление современных DAG-реестров (см. таблицу) показывает, что для сценария умного города, киберфизических систем и IoT-сетей недостаточно учитывать только скорость подтверждения [12]. Новая архитектура должна одновременно удовлетворять нескольким требованиям: не требовать майнинга, поддерживать параллельное добавление транзакций, быть пригодной для маломощных узлов, ограничивать рост активной истории реестра, а также сохранять устойчивость при конфликтных транзакциях и задержках распространения сообщений.

Сопоставление свойств DAG-реестров

Comparison of DAG registries properties

Подход	Скорость подтверждения транзакции	Требования к памяти	Масштабируемость	Майнинг	Безопасное удаление истории	Проблемы безопасности
IOTA	Высокая	Высокие	Высокая	Нет	Нет	Спам транзакций, разделение сети путем создания конфликтных транзакций
Hashgraph	Высокая	Средние	Низкая	Комитет	Да	Большое количество служебных транзакций в сети
PHANTOM/BlockDAG	Высокая	Средние	Среднее	Да	Нет	Ограничение пропускной способности k -кластером
Avalanche	Высокая	Средние	Среднее	Да	Да	Атаки на голосование

Из проведенного сравнения следует, что новая модель должна сочетать три свойства:

- сохранять DAG-структуру, так как она наилучшим образом подходит для параллельного потока транзакций;
- ограничивать рост истории за счет периодической фиксации состояния и перехода к новому генезис-состоянию;
- явно задавать правила локального выбора родителей и *past cone*, поскольку в P2P-сети узел не имеет глобального знания DAG и принимает решения только по локально видимым объектам.

С учетом этих факторов, наиболее пригодной базой для дальнейшего развития является IOTA-подобная DAG-архитектура, поскольку она совмещает параллельную структуру реестра с идеей поддержки прошлой области графа. Авторами доработана структура IOTA: для предотвращения бесконечного роста истории реестра введена ротация генезис-состояния, а выбор родителей и родительского подграфа (*past cone*) рассматривается как отдельная задача созданного протокола.

3. АРХИТЕКТУРА DAG-РЕЕСТРА С РОТАЦИЕЙ ГЕНЕЗИС-СОСТОЯНИЯ

В основе предложенной DAG-архитектуры реестра лежит идея плавающего генезис-блока. Вместо хранения всей растущей истории транзакций сеть периодически фиксирует новое состояние реестра – актуальный набор UTXO (Unspent Transaction Output). С этого момента все новые транзакции клиентов ссылаются на новое генезис-состояние. Чтобы такие генезис-состояния были проверяемыми и не появлялись в нескольких версиях, они должны образовывать однозначную цепочку доверия. Поэтому вводится последовательность $G_0 \rightarrow G_1 \rightarrow G_2 \rightarrow \dots$, где каждый новый генезис содержит ссылку на предыдущий и подтверждается кворумом валидаторов. Для удобства описания вводится понятие эпохи: эпоха i – интервал времени (или объема данных), в котором все транзакции относятся к одному

и тому же генезису G_i . Переход к следующей эпохе выполняется через процедуру ротации: подтвержденные транзакции текущего слоя сворачиваются в новое состояние G_{i+1} .

Используются два уровня протокола: локальный слой транзакций в виде DAG и глобальный слой фиксации состояния в виде цепочки генезисов $G_0 \rightarrow G_1 \rightarrow \dots$. На уровне DAG транзакции распространяются и формируют граф зависимостей, а на уровне генезисов сеть периодически фиксирует итоговое состояние. Ключевое требование данного двухуровневого подхода состоит в следующем: если транзакция считается подтвержденной внутри DAG текущей эпохи, то ее эффект обязан отразиться в следующем генезисе G_{i+1} . Механизм подтверждения в DAG не может быть полностью пользовательским и независимым от валидаторов. В «чисто пользовательской» модели, где подтверждение строится только на ссылках транзакций друг на друга, нет гарантий, что подтверждение внутри графа будет согласовано с тем, что именно валидаторы зафиксируют при ротации. В результате возможна ситуация, когда пользователи считают транзакцию подтвержденной по правилам DAG-структуры, но при формировании G_{i+1} валидаторы ее не включают в свертку состояния, например, из-за конфликтов, задержек распространения или разного локального вида графа.

В этой связи реализован гибридный подход, согласно которому DAG используется как структура данных для распространения и накопления транзакций внутри эпохи, а окончательное подтверждение (финальность) задается валидаторным консенсусом PoS-BFT. Важно отметить, что валидаторы не уничтожат справедливость сети, так как они будут подтверждать только те транзакции, которые набрали значительный вес в текущей эпохе сети DAG.

В каждой эпохе i задано множество валидаторов: $V_i = \{v_1, v_2, \dots, v_n\}$. Каждый валидатор имеет вес $w(v) \in (0, 1]$, при этом $\sum_{v \in V_i} w(v) = 1$. Вес валидатора интерпретируется как доля стека. При голосовании

учитывается не количество подписей, а их суммарный вес.

Состояние реестра задается множеством непотраченных выходов (UTXO). S – множество существующих выходов, которые можно использовать. Транзакция tx включает входы $tx.in = [op_1, \dots, op_k]$, где op_i – ссылка на вход UTXO; выходы $tx.out = [(addr_1, val_1), \dots, (addr_m, val_m)]$, где $addr_i$ – адрес, куда отправляются выходы, val_i – выходы; ссылки на предков $tx.parents$, метки эпохи $tx.epoch = i$ и метки генезиса $tx.gref = g_i$. В практической реализации структура транзакции может быть изменена.

Переход состояния по одной транзакции задается как

$$\delta(S, tx) = \begin{cases} S \setminus Spent(tx) \cup Created(tx), & \text{если } tx \text{ валидна.} \\ \perp, & \text{иначе} \end{cases}$$

Для эпохи i транзакции образуют ориентированный ациклический граф (DAG) $D_i = (V_i, E_i)$, где V_i – все транзакции эпохи i , E_i – ребра подтверждений.

Генезис-состояние G_i фиксирует состояние эпохи i и имеет идентификатор $g_i = H(G_i)$. Каждая транзакция эпохи i должна удовлетворять условию $tx.gref = g_i$. Это связывает все транзакции эпохи с одним начальным состоянием и позволяет затем отделить эпохи друг от друга. Генезисы образуют последовательность $G_0 \leftarrow G_1 \leftarrow G_2 \leftarrow \dots$, где каждый G_i содержит ссылку на предыдущий $H(G_{i-1})$.

Чтобы формально увязать финальность в DAG в глобальном состоянии, используется механизм чекпоинтов, аналогичный механизму milestone в IOTA. Чекпоинт в эпохе i предлагает один валидатор, выбранный с помощью PoS-процедуры. В одной эпохе может быть несколько раундов, в каждом из которых свой чекпоинт. Чекпоинт становится действительным только после того, как его поддержит кворум валидаторов. Это фиксируется кворум-сертификатом QC – набором подписей не менее чем $2f + 1$ валидаторов.

Валидаторы периодически выпускают подтверждающие блоки, отражающие ви-

димую ими часть DAG. Подтверждающие блок валидатора v имеет вид:

$$VB_j(v) = (v, j, ApprovedTips_j, prevVB_v, sig_v),$$

где $ApprovedTips_j$ – типсы (tips) DAG, которые валидатор считает актуальными на момент выпуска блока; $prevVB_v$ – ссылка на предыдущий подтверждающий блок того же валидатора; sig_v – подпись валидатора. Подтверждающий блок не финализирует состояние, а используется для накопления веса транзакций внутри DAG.

Поддержка транзакции tx в слоте $slot_{i,k}$ определяется как суммарный вес валидаторов, чьи актуальные подтверждающие блоки покрывают данную транзакцию: $support_{i,k}(tx) = \sum_{v: tx \in Past(ApprovedTips_v)} w(v)$.

Функция $support(tx)$ показывает, какая доля валидаторного веса «видит» транзакцию в своей актуальной границе DAG. Это значение используется не для единоличной финализации транзакции, а для детерминированного порядка применения транзакций внутри слот-коммита.

Для объекта x кворум-сертификат представляет собой набор подписей валидаторов суммарным весом $\geq 2f + 1$: $QC(x) = \{Sig_v(H(x)) \mid \sum w(v) \geq 2f + 1\}$.

Чекпоинт эпохи i раунда k – структура $CP_{i,k}$, содержащая ссылку на предыдущий чекпоинт $H(CP_{i,k-1})$; набор ссылок на «верхние вершины» $TIPS \subseteq V_i$, финализирующий сертификат $QC(CP_{i,k})$. В практической реализации структура транзакции может иметь больше полей с информацией.

Для каждого слота выбирается создатель коммита: $Proposer(i, k, view)$. Выбор создателя коммита выполняется с учетом PoS-весов валидаторов. Если создатель не выпустил корректный слот-коммит за отведенное время, начинается следующий $view := view + 1$, и выбирается новый создатель. Создатель только предлагает слот-коммит, но не может финализировать его самостоятельно. Финализация требует кворум-сертификата.

Для каждой «верхней» вершины $t \in TIPS$ существует покрытие: $Cone(t) = \{tx \in V_i \mid tx \text{ достижима из } t \text{ по ребрам } E_i\}$. Пусть $Conf_{i,k}$ – множество транзакций, подтвержденных

сеть после чекпоинта k : $Conf_{i,k} = Conf_{i,k-1} \cup \cup Cone(CP_{i,k}.tips)$, $Conf_{i,0} = \emptyset$. Соответственно, с каждым финализированным чекпоинтом подтвержденных транзакций становится больше.

Протокол консенсуса относится к классу PoS-BFT. Он является BFT-протоколом, поскольку все честные узлы детерминировано вычисляют один и тот же порядок обработки транзакций, подтвержденных чекпоинтами. Этот порядок задается следующим определением. Для каждой транзакции $tx \in Included_{i,k}$ вычисляется ключ $Key_{i,k}(tx) = (-support_{i,k}(tx), Hash(tx))$.

Транзакции сортируются лексикографически по следующему порядку: слот принадлежности транзакции; валидаторная поддержка в слоте; хеш транзакции. Такой порядок нужен для одинакового разрешения конфликтов всеми узлами.

Пусть $S_{i,0}$ – состояние на старте эпохи. После коммита $CP_{i,k}$ состояние обновляется: берется $New_{i,k}$ и сортируется по Ord , затем последовательно применяется δ . Если $\delta = \perp$, транзакция помечается как отвергнутая. Тогда состояние после k -го раунда – $S_{i,k} = Apply(S_{i,k-1}, New_{i,k})$, где $Apply$ – последовательное применение δ в порядке Ord .

Все раунды внутри эпохи завершаются финальным чекпоинтом. После его финализации узлы фиксируют итоговое состояние эпохи, формируют новый генезис-блок и переходят к следующей эпохе, начиная построение нового слоя DAG_{i+1} .

END-чекпоинт – чекпоинт специального типа $type = END$, который считается финализированным, если существует валидный кворум-сертификат $QC(E_i)$. Финализация E_i задает единую границу эпохи i . Конец эпохи определяется не локальным временем/памятью узла, а только фактом финализации кворумом. После финализации новое состояние определяется как $S_{i+1} := S_{i,k_{end}}$, где k_{end} – индекс END-чекпоинта.

Генерация нового генезис блока разбивается на следующие шаги:

1. $C_{i+1} = Commit(S_{i,k_{end}})$ – метainформация о новом состоянии, например, C_{i+1} –

корень дерева Меркля от UTXO, $Commit$ – функция создания дерева Меркля.

2. $G_{i+1} = \{prev = H(G_i), Conf = (H(S_{i+1,0}), QC(S_{i+1,0})), stateCommit = C_{i+1}, epoch = i + 1\}$.

Генезис G_{i+1} считается принятым, только если существует $QC(G_{i+1})$. После этого узлы переключаются на эпоху $i + 1$ и могут обрезать историю эпохи i .

Представленное описание DAG-реестра задает связь между локальным DAG, подтверждающими блоками, чекпоинтами и ротацией генезис-состояния. Однако корректность такой схемы зависит не только от самих формальных описаний, но и от того, как узлы принимают решения в условиях неполного знания сети. В этой связи разработанное решение исследовано в эмуляторе P2P-среды, где каждый участник имеет собственное локальное представление DAG и получает транзакции с задержками.

4. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ РАЗРАБОТАННОЙ АРХИТЕКТУРЫ DAG-РЕЕСТРА

Для оценки эффективности разработанной архитектуры по ограничению роста памяти проведено сравнение трех архитектур: DAG без сворачивания истории (архитектуры IOTA), DAG со сворачиванием (разработанной архитектуры) и классического PoS-блокчейна.

Эксперимент проводился на эмуляторе со 100 узлами в течение 110 слотов, объем потребляемой памяти измерялся на случайном узле-участнике сети (рис. 1). Фиолетовая линия на графике соответствует DAG без сворачивания истории. В этом случае объем памяти непрерывно увеличивается и к 110-му слоту достигает приблизительно 10 МБ. Такой рост объясняется тем, что каждая новая транзакция добавляет в граф не только саму запись, но и ссылки на родителей, а устаревшие объекты не удаляются, поскольку они необходимы для проверки целостности новых записей. Синяя линия соответствует

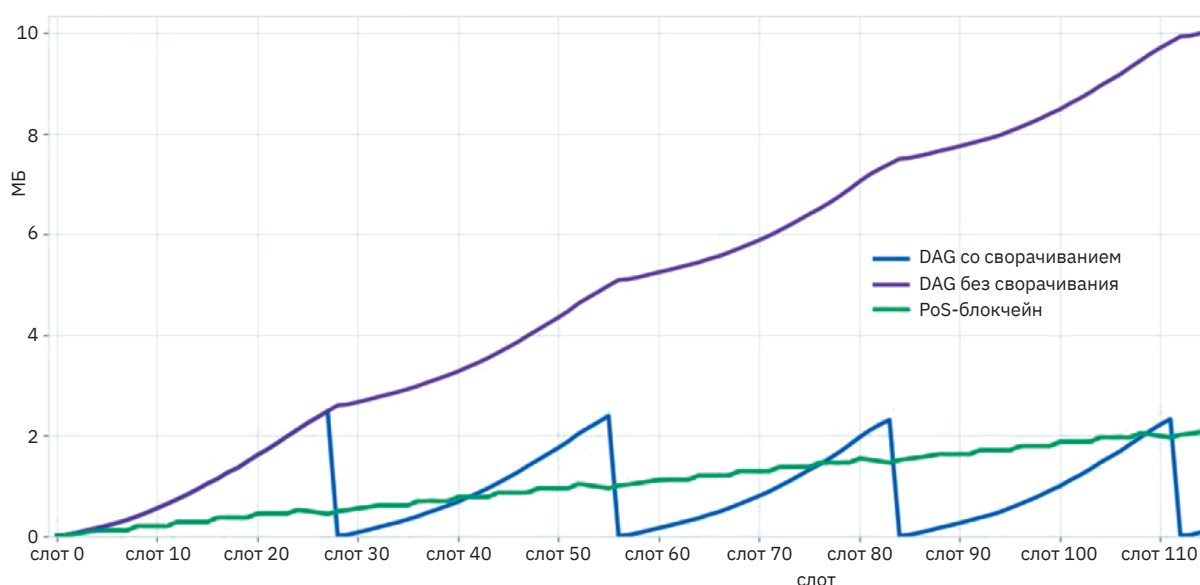


Рис. 1 | Оценка потребления памяти на узле

Fig. 1 | Memory consumption assessment at the node

разработанной модели DAG со сворачиванием. После завершения каждой эпохи финализированная история сворачивается в новое генезис-состояние, а устаревшая часть DAG удаляется. Поэтому потребление памяти периодически снижается после каждой эпохи и к 110-му слоту составляет около 2 МБ, что примерно в пять раз меньше, чем в модели без сворачивания. Зеленая линия показывает базовую PoS-блокчейн-модель, в которой история также продолжает накапливаться. К 110-му слоту потребление памяти превышает 12 МБ, что выше, чем в DAG без сворачивания, поскольку блокчейн-структура хранит каждый блок целиком без возможности удаления завершенных частей.

Таким образом, применение механизма сворачивания позволяет ограничить рост памяти и сократить объем хранимых данных в пять раз по сравнению с базовой ИОТА-подобной реализацией, что особенно важно для малоресурсных устройств.

Для оценки защищенности разработанной DAG-архитектуры рассмотрена атака конфликтующих транзакций. Цель эксперимента состоит не в том, чтобы проверить сам факт отклонения конфликтов, а в том, чтобы измерить, насколько конфликтная

нагрузка мешает подтверждению честных пользовательских транзакций. Это важно для DAG-реестров, поскольку конфликтная ветка может не только быть отклонена сама, но и повлиять на транзакции, которые успели построиться поверх нее из-за сетевых задержек.

Атакующий создает несколько транзакций, использующих одни и те же средства. Такие транзакции не могут быть одновременно применены к УТХО-состоянию, поэтому сеть должна выбрать одну из ветвей или отклонить конфликтные объекты. При этом из-за неполной синхронности разные части P2P-сети могут сначала «увидеть» разные варианты конфликта. Часть честных узлов может сослаться на одну ветку до того, как узнает о существовании другой.

В эксперименте используется волновая атака. Конфликтные транзакции отправляются не одним пакетом, а несколькими волнами внутри слота: в начале, середине и конце. Такая схема делает атаку более реалистичной, поскольку конфликтная информация приходит в разные части сети в разное время и создает временное расхождение локальных представлений DAG.

Выбраны три сценария тестирования:

- слабая атака: 2 конфликтные транзакции в каждой волне, всего 6 конфликтных транзакций за слот против 30 валидных;
- средняя атака: 6 конфликтных транзакций в каждой волне, всего 18 конфликтных транзакций за слот против 30 валидных;
- стрессовая атака: 10 конфликтных транзакций в каждой волне, всего 30 конфликтных транзакций за слот против 30 валидных.

Каждый эксперимент проводился на протяжении четырех эпох. В качестве основной метрики использована доля отклоненных обычных честных транзакций по слотам. Конфликтные транзакции атакующего в этой метрике не учитывались, так как целью являлась оценка влияния атаки именно на честный пользовательский поток.

Результаты экспериментов подтверждают работоспособность и эффективность разработанной архитектуры. При слабой атаке система сохраняет устойчивость. На рис. 2, а видно, что отклонение обычных транзакций почти отсутствует: среднее значение по эпохам составляет 0,2–0,5 %. Это показывает, что при слабой конфликтной нагрузке подтверждающие блоки успевают покрывать полезный *past cone*, а конфликтные ветви не вытесняют обычные транзакции из процесса подтверждения.

При средней атаке нагрузка становится заметной (рис. 2, б). В этом режиме появляются устойчивые потери честных транзакций: средние значения по эпохам находятся примерно в диапазоне 13–23 %. Это означает, что конфликтные ветви начинают занимать часть локального DAG, а подтверждающие блоки не всегда успевают одновременно подтвердить обычный поток и разрешить конфликтные зависимости. Однако средние потери составляют 18 %, что лучше, чем у основного конкурента IOTA.

При стрессовой атаке конфликтная нагрузка становится сопоставимой с честной (рис. 2, в). Такая атака в реальности достаточно редка, но возможна. В этом

случае система продолжает работать, однако потери обычных транзакций становятся значительными. Средние значения по эпохам находятся примерно в диапазоне 18–28 %. Такой результат показывает предел устойчивости выбранных параметров. Можно сделать вывод, что даже при худшем сценарии атаки большая часть честных транзакций остается в сети после финализации.

Результаты тестов показали, что поведение DAG-реестра существенно зависит от выбранных параметров архитектуры: правил выбора родителей, оценки *past cone*, приоритетов конфликт-ветвей, числа подтверждающих блоков и лага финализации. Если подтверждающие блоки недостаточно точно выбирают полезный *past cone*, часть честных транзакций не успевает набрать вес $2/3$ и отклоняется. Поэтому рост потерь при средней и стрессовой атаке демонстрирует не только растущее воздействие атаки, но и чувствительность архитектуры к настройке рабочих коэффициентов. Для дальнейшего улучшения необходимо адаптивно настраивать веса выбора *past cone*, число подтверждающих блоков и окно финализации в зависимости от текущей нагрузки сети.

5. ЗАКЛЮЧЕНИЕ

Представлена новая модель DAG-реестра с эпохами, BFT-финализацией и ротацией генезис-состояния. Внутри эпохи транзакции формируют направленный ациклический граф, а валидаторы с помощью блоков подтверждения накапливают поддержку выбранных областей графа. После достижения кворума фиксируется итоговое состояние, формируется новое генезис-состояние, а история завершённой эпохи может быть безопасно удалена. Предложенный механизм учитывает возраст транзакций, полноту известной истории, накопленную поддержку и риск конфликта. Это позволяет сохранять полезную часть DAG и одновременно ограничивать поддержку слабых конфликтующих ветвей.

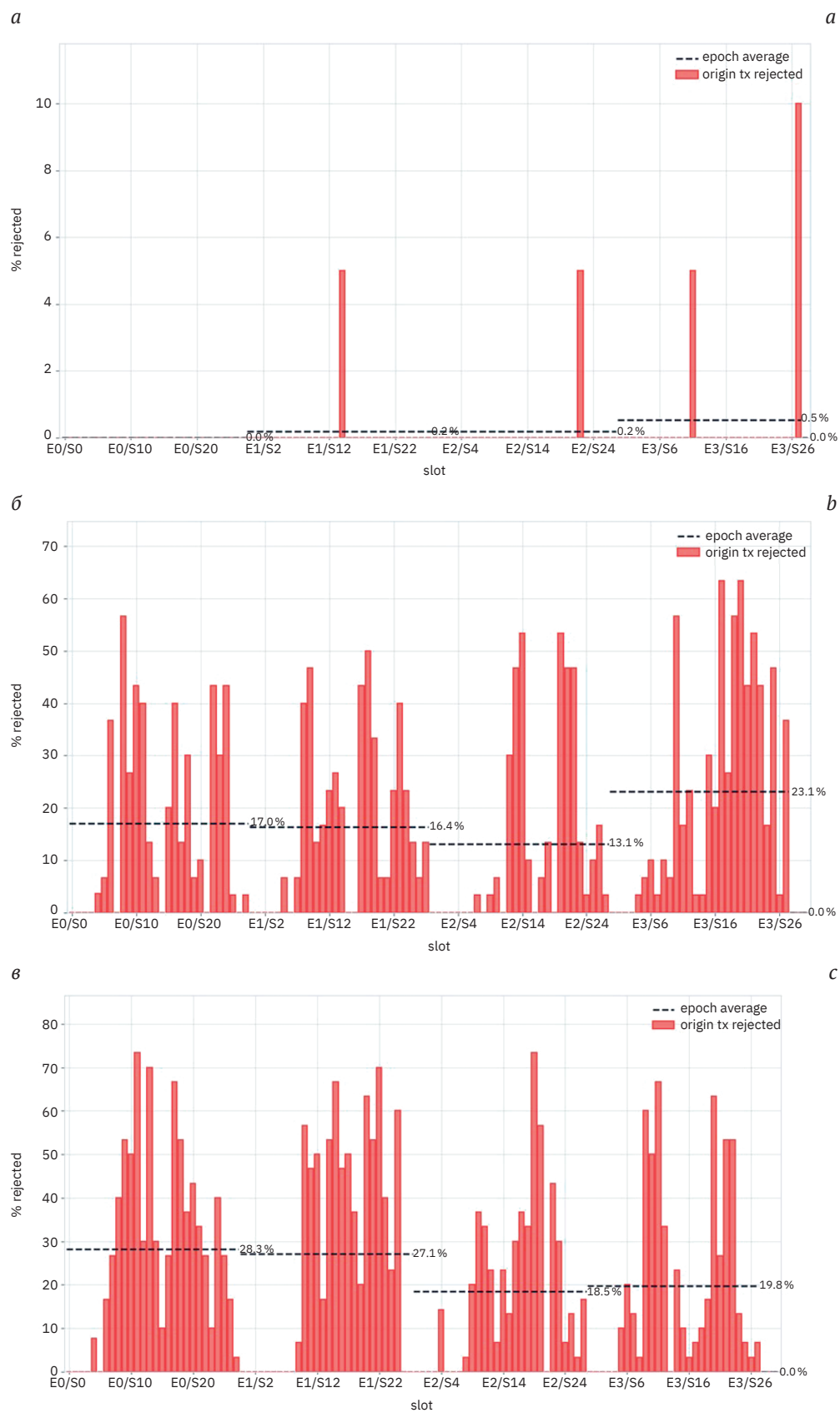


Рис. 2 | Анализ безопасности DAG-реестра: атака с 6 (а), 18 (б) и 30 (в) конфликтами на слот

Fig. 2 | DAG ledger security analysis: attack with 6 (a), 18 (b), and 30 (c) conflicts per slot

Проведена экспериментальная оценка разработанной модели. При увеличении доли конфликтующих транзакций потери честных операций возрастали примерно с 10 до 25 %, однако оставались ниже значения, наблюдаемого в исходной архитектуре IOTA. Доля принятых конфликтующих транзакций при этом оставалась низкой и составляла ниже 5 %. Дополнительно показано, что ротация генезис-состояния ограничивает рост памяти: в отличие от

стандартного DAG-реестра и классического блокчейна, разработанная модель периодически освобождает историю завершённых эпох, что позволяет в пять раз сократить объём хранения на узлах реестра.

Полученные результаты показывают, что сочетание DAG-подобной структуры реестра, BFT-финализации и ротации генезис-состояния позволяет одновременно повысить безопасность реестра и снизить требования к хранению его истории.

КОНФЛИКТ ИНТЕРЕСОВ / CONFLICT OF INTERESTS

Авторы заявляют об отсутствии конфликта интересов / The authors declare no conflict of interests.

СПИСОК ИСТОЧНИКОВ

1. **Lei Zhang, Hao Xu, Oluwakayode Onireti et al.** How Much Communication Resource is Needed to Run a Wireless Blockchain Network? // IEEE Network. 2022. Vol. 36. № 1. P. 128–135. DOI: 10.1109/MNET.101.2100208.
2. **Yixin Li, Bin Cao, Mugen Peng et al.** Direct Acyclic Graph-Based Ledger for Internet of Things: Performance and Security Analysis // IEEE/ACM Transactions on Networking. 2020. Vol. 28. № 4. P. 1643–1656. DOI: 10.1109/TNET.2020.2991994.
3. **Полтавцева М. А., Зегжда Д. П.** Анализ гетерогенных прецедентов в задачах информационной безопасности // Методы и технические средства обеспечения безопасности информации. 2022. № 31. С. 4–6.
4. **Popov S., Moog H., Camargo D. et al.** The Coordicide IOTA. URL: https://files.iota.org/papers/20200120_Coordicide_WP.pdf (дата обращения: 10.05.2026).
5. **Orel E. M., Moskvina D. A., Lyashenko A. A.** Attack graph-based data presentation for use in automated security analysis systems // Automatic Control and Computer Sciences. 2024. Vol. 58. № 8. P. 1436–1441.
6. **Зегжда Д. П., Москвина Д. А., Орел Е. М.** Автоматизация управления безопасностью гетерогенных систем на основе интеллектуального анализа графа кибератак // Методы и технические средства обеспечения безопасности информации. 2020. № 29. С. 53–54.
7. **Baird L., Luyckx A.** The Hashgraph Protocol: Efficient Asynchronous BFT for High-Throughput Distributed Ledgers // 2020 International Conference on Omni-layer Intelligent Systems (COINS), Barcelona, Spain. 2020. P. 1–7. DOI: 10.1109/COINS49042.2020.9191430.
8. **Baird L.** The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance. URL: <https://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf> (дата обращения: 10.05.2026).
9. **Poltavtseva M. A.** Heterogeneous data aggregation and normalization in information security monitoring and intrusion detection systems of large-scale industrial CPS // Proceedings of the Institute for System Programming of the RAS. 2020. Vol. 32. № 5. P. 131–142.
10. **Sompolinsky Y., Wyborski S., Zohar A.** PHANTOM GHOSTDAG: A scalable generalization of Nakamoto consensus // Proceedings of the 3rd ACM Conference on Advances in Financial Technologies (AFT '21). Association

for Computing Machinery, New York, NY, USA. 2021. P. 57–70. DOI: 10.1145/3479722.3480990.

11. Amores-Sesar I., Cachin C., Tedeschi E. When Is Spring Coming? A Security Analysis of Avalanche Consensus // 26th International Conference on Principles of Distributed Systems (OPODIS 2022). Leibniz International Proceedings in Informatics (LIPIcs). 2023. Vol. 253. P. 10:1–10:22. DOI: 10.4230/LIPIcs.OPODIS.2022.10.
12. Зубков Е. А., Москвин Д. А., Зегжда Д. П. Комплексные методы моделирования киберфизических систем в контексте кибербезопасности // Методы и технические средства обеспечения безопасности информации. 2023. № 32. С. 28–29.

REFERENCES

1. Lei Zhang, Hao Xu, Oluwakayode Onireti et al. How Much Communication Resource is Needed to Run a Wireless Blockchain Network? *IEEE Network*. 2022. Vol. 36. No. 1, pp. 128–135. DOI: 10.1109/MNET.101.2100208.
2. Yixin Li, Bin Cao, Mugen Peng et al. Direct Acyclic Graph-Based Ledger for Internet of Things: Performance and Security Analysis. *IEEE/ACM Transactions on Networking*. 2020. Vol. 28. No. 4, pp. 1643–1656. DOI: 10.1109/TNET.2020.2991994.
3. Poltavtseva M. A., Zegzhda D. P. Analysis of Heterogeneous Precedents in Information Security Tasks. *Metody i tehnicheskie sredstva obespecheniya bezopasnosti informacii*. 2022. No. 31, pp. 4–6. (In Russian)
4. Popov S., Moog H., Camargo D. et al. The Coordicide IOTA. URL: https://files.iota.org/papers/20200120_Coordicide_WP.pdf (accessed: 10.05.2026).
5. Orel E. M., Moskvin D. A., Lyashenko A. A. Attack graph-based data presentation for use in automated security analysis systems. *Automatic Control and Computer Sciences*. 2024. Vol. 58. No. 8, pp. 1436–1441.
6. Zegzhda D. P., Moskvin D. A., Orel E. M. Automation of security management in heterogeneous systems based on intelligent analysis of cyberattack graphs. *Metody i tehnicheskie sredstva obespecheniya bezopasnosti informacii*. 2020. No. 29, pp. 53–54. (In Russian)
7. Baird L., Luykx A. The Hashgraph Protocol: Efficient Asynchronous BFT for High-Throughput Distributed Ledgers. 2020 International Conference on Omni-layer Intelligent Systems (COINS), Barcelona, Spain. 2020, pp. 1–7. DOI: 10.1109/COINS49042.2020.9191430.
8. Baird L. The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance. URL: <https://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf> (accessed: 10.05.2026).
9. Poltavtseva M. A. Heterogeneous data aggregation and normalization in information security monitoring and intrusion detection systems of large-scale industrial CPS. *Proceedings of the Institute for System Programming of the RAS*. 2020. Vol. 32. No. 5, pp. 131–142.
10. Sompolinsky Y., Wyborski S., Zohar A. PHANTOM GHOSTDAG: A scalable generalization of Nakamoto consensus. Proceedings of the 3rd ACM Conference on Advances in Financial Technologies (AFT '21). Association for Computing Machinery, New York, NY, USA. 2021, pp. 57–70. DOI: 10.1145/3479722.3480990.
11. Amores-Sesar I., Cachin C., Tedeschi E. When Is Spring Coming? A Security Analysis of Avalanche Consensus. 26th International Conference on Principles of Distributed Systems (OPODIS 2022). *Leibniz International Proceedings in Informatics (LIPIcs)*. 2023. Vol. 253, pp. 10:1–10:22. DOI: 10.4230/LIPIcs.OPODIS.2022.10.
12. Zubkov E. A., Moskvin D. A., Zegzhda D. P. Comprehensive Methods for Modeling Cyber-Physical Systems in the Context of Cybersecurity. *Metody i tehnicheskie sredstva obespecheniya bezopasnosti informacii*. 2023. No. 32, pp. 28–29. (In Russian)

СВЕДЕНИЯ ОБ АВТОРАХ / INFORMATION ABOUT AUTHORS

КАЛИНИН Максим Олегович – д-р техн. наук, профессор, Санкт-Петербургский политехнический университет Петра Великого, Россия, 195251, Санкт-Петербург, ул. Политехническая, д. 29
E-mail: max@ibks.spbstu.ru
ORCID: 0000-0002-9732-0099

KALININ Maxim O. – Doctor of Engineering Sciences, Professor, Peter the Great St. Petersburg Polytechnic University, Russia, 195251, St. Petersburg, Polytechnicheskaya str., 29

ГАРТВИГ Роман Андреевич – студент, Санкт-Петербургский политехнический университет Петра Великого, Россия, 195251, Санкт-Петербург, ул. Политехническая, д. 29
E-mail: gartvig.ra@edu.spbstu.ru

GARTVIG Roman A. – Student, Peter the Great St. Petersburg Polytechnic University, Russia, 195251, St. Petersburg, Polytechnicheskaya str., 29