Алгоритм консенсуса платформы Tendermint и механизм Proof Of Lock Change

И.Ю. Герасимов, И.В. Чижов

***Аннотация*—В работе рассматривается алгоритм консенсуса, основанный на задаче о византийских генералах, в форме, которой он реализован в платформе Tendermint. Этот алгоритм применяется в реализации децентрализованных приложений и платежных системах с представлением данных на основе технологии блокчейна с целью достижения консенсуса между пользователями относительно очередного блока, добавляемого в цепочку блоков. Особенностью алгоритма консенсуса Tendermint является то, в нём участвует не любой желающий из числа абонентов сети, а только специальные участники, которые называются валидаторами. В работе приводится описание алгоритма консенсуса, структур данных, представляющих собой голоса валидаторов с некоторым весом, условия, с помощью которых формируется голос в рамках алгоритма консенсуса, а также механизма Proof Of Lock Change. Доказывается конечность, корректность и валидность этого алгоритма при условии, что количество валидаторов, которые изолированы от системы или которые совершают действия, не соответствующие алгоритму консенсуса, не превышает одной трети от общего числа валидаторов. Проведен анализ механизма Proof Of Lock Change. Показано, что возможные вариации алгоритма консенсуса, связанные с отказом от реализации механизма Proof Of Lock Change, имеют серьёзный дефект, связанный с нарушением свойства конечности, что может привести к отказу работы системы.**

***Ключевые слова*—блокчейн, протокол консенсуса, tendermint, механизм proof of lock change**

I. Введение

На сегодняшний день блокчейн является довольно известной структурой хранения данных, используемой в децентрализованных приложениях и платежных системах. Однако ввиду отсутствия единого центра управления в таких системах необходим механизм, позволяющий участникам сети достичь соглашения относительно каждого блока данных, который добавляется в блокчейн. В работе рассматривается алгоритм консенсуса, основанный на задаче о византийских генералах, Byzantine Fault Tolerance (BFT), в форме, которой он реализован в Tendermint [1]. Данный алгоритм консенсуса является альтернативой алгоритмам типа ProofofWork[4], которые требуют вычислительных затрат, и ProofofStake[5], в котором ресурс, необходимый для получения прибыли с добавления блоков в блокчейн, производится самой системой. Особенностью алгоритма Tendermint является то, что в процессе принятия решения о включении блока в цепочку принимается группой специально выбранных валидаторов. Это создаёт некоторую централизацию, что может повлиять на доверие пользователей такой системе.

Некоторые системы блокчейн, основанные на Tendermint, отказываются от реализации достаточно трудоёмкого в реализации механизма ProofofLock Change (PoLC). Главным результатом работы является доказательство того, что удаление этого механизма приведет к нарушению работы системы и может приводить к ситуации рассогласования данных в блоках блокчейна.

Статья получена 24 апреля 2019. Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 18 – 29 − 03124\18

И.Ю. Герасимов, Московский государственный университет имени М.В. Ломоносова (ilia_gerassimov@mail.ru)
И.В. Чижов, Московский государственный университет имени М.В. Ломоносова, (ichizhov@cs.msu.ru)

II. Описание алгоритма консенсуса Tendermint

Опишем алгоритм консенсуса, использующийся в Tendermint [2, 3]. Назовём высотой блокчейна длину, считая от корня блокчейна, самой длинной цепочки блоков.

Пусть в некоторый момент времени блокчейн имеет высоту H, имеется набор валидаторов V. Эти валидаторы в рамках Tendermint осуществляют процедуру проверки очередного блока, содержащего транзакции, и включение его в цепочку. Для этого валидаторы должны достичь согласия или консенсуса о рассматриваемом блоке. Алгоритм включения следующего блока проходит в течение нескольких раундов и заключается в голосовании валидаторов за блок. Голос каждого валидатора имеет свой вес. При голосовании валидаторы в каждом выполняют последовательность следующих шагов:

NewHeight→[Propose → Prevote → Precommit]→Commit.

Причём шаги NewHeight и Commit зависят только от текущей высоты блокчейна, а точнее от высоты, на которую добавляется новый блок. Шаги Propose, Prevote и Precommit могут быть выполнены последовательно несколько раз, каждое выполнение будем называть раундом. Таким образом, эти шаги зависят не только от высоты блокчейна, но и от номера раунда. Можно считать, что каждый шаг алгоритма консенсуса – это состояние конечного автомата, а сам алгоритм для каждого валидатора – конечный автомат. Под действием некоторых событий и условий перехода автомат валидатора «путешествует» по своим состояниям.

Прежде чем переходить к описанию каждого шага

дадим общие условия перехода. Если в каком-то раунде, на любом шаге выполняется условие: получено более 2/3 от общего веса голосов за некоторый блок, то необходимо перейти к шагу Commit(height); если выполняется условие: получено более 2/3 от общего веса голосов Prevote для раунда round+x, $x > 0$, то перейти к Prevote(height, round+x); и, наконец, если выполняется условие: получено более 2/3 от общего веса голосов Precommit для раунда round+x, $x > 0$, то перейти к Precommit(height, round+x).

Описание шагов протокола начнём с описания Commit(height). Он заключается в фиксировании момента времени перехода на этот шаг ($CommitTime$), ожидании получения блока и переходе на шаг NewHeight(height+1).

Шаг NewHeight(height) заключается в добавлении в блок информации о голосах и ожидании «отставших» голосов до момента времени $CommitTime + TimeoutCommit$ и, наконец, внесение блока в блокчейн. Перейдём к описанию шага Propose(height, round). На каждом раунде выбирается специальный валидатор Proposer, который отправляет сообщение Proposal. Функция, по которой определяется Proposer, известна всем, ее вычисление не требует взаимодействия с другими участниками. Таким образом, по высоте $height$ и номеру раунда $round$ однозначно определяется Proposer. Кроме того, каждый валидатор имеет следующие переменные: $lockedblock, lockedround$ – блок и раунд соответственно, для которых во время раунда $lockedround$ было получено более 2/3 от общего веса голосов Prevote за блок этого раунда. Начальное значение $lockedround = -1$, при этом $lockedround < round$.

Действия на шаге Propose(height, round) зависят от того является ли валидатор Proposer, если это так, то он отправляет в систему сообщение Proposal. Опишем сообщение Proposal. Оно зависит от значения $lockedround$. Если $lockedround == -1$, то оно содержит новый блок, полученный из текущего множества транзакций, и переменную $ProofofLock = -1$. Если $lockedround > -1$, то Proposal содержит $lockedblock,$ и $ProofofLock(lockedround, Prevotes(lockedround))$, где $Prevotes(lockedround)$ – множества Prevote голосов валидатора, полученных на раунде $lockedround$, а $ProofofLock$ – данные, используемые в механизме Proof of Lock Change (PoLC).
Если же это обычный валидатор, то он ожидает получения сообщения Proposal.

При получении на этом шаге сообщения Proposal, необходимо перейти на шаг Prevote(height, round), если же сообщение не пришло в течение промежутка времени timeoutPropose, то необходимо также перейти на шаг Prevote(height, round).

Задача валидатора на шаге Prevote(height, round) сформировать свой голос о блоке и отправить его другим валидаторам. Сообщение Prevote содержит блок. Если выполнены четыре условия:

1) $-1 < lockedround < Proposal.ProofofLock.lockedround < round$, здесь $Proposal.ProofofLock.lockedround$ – значение $lockedround$ из сообщения $Proposal$;
2) $lockedblock! = Proposal.block$, здесь $Proposal.block$ – блок данных из сообщения $Proposal$;
3) для полученных голосов $Proposal.ProofofLock.Prevotes(lockedround)$ на шаге $Proposal.ProofofLock.lockedround$ более 2/3 от общего веса голосов за $Proposal.block$;
4) $Proposal.block$ удовлетворяет всем требованиям, которые предъявляются конкретной реализацией значению блокчейна,

то валидатор в сообщении Prevote значение блока полагает равным значению $Proposal.block$. Если выполнены условия 1)-3), то выполняется присваивание: $lockedround := -1$. Если же хотя бы одно из условий 1)-3) нарушается, то в сообщении в качестве блока берётся значение $lockedblock$ валидатора. И, наконец, если условия 1)-3) выполняются, а условие 4) нарушается, либо если сообщение Propose не было получено в течение промежутка времени $timeoutPrevote$, то $lockedblock$ устанавливается в значение $NULL$.

В конце шага валидатор отправляет сообщение Prevote, которое содержит номер раунда, высоту блокчейна и сформированный блок. После отправки сообщения необходимо перейти на следующий шаг. Если валидатор получил более 2/3 от общего веса голосов для определенного блока или 2/3 от общего веса голосов со значением NULL, то валидатор переходит на шаг Precommit(height, round). В том случае, если произошло событие timeoutPrevote и получены любые голоса с весом более 2/3 от общего веса, то валидатор также переходит на шаг Precommit(height, round).

Опишем шаг Precommit(height, round). Валидатор должен сформировать сообщение Precommit, которое содержит блок данных, текущую высоту блокчейна и номер раунда. Формирование блока данных в сообщении выполняется следующим образом. Если для определенного блока $block$ получено более 2/3 от общего веса голосов Prevote, то $lockedblock := block$; $lockedround := round$; и в Precommit указывается блок $block$. Если получено более 2/3 от общего веса голосов Prevote со значением $NULL$, то полагается $lockedround := -1$, в Precommit в качестве блока указывается $NULL$. Во всех остальных случаях в сообщении Precommit в качестве блока указывается значение $NULL$.

Осталось описать для этого шага условие перехода на следующий шаг. Если валидатор получил более 2/3 от общего веса голосов со значением $NULL$, то валидатор должен перейти на шаг Propose(height, round+1). Если валидатор не завершил работу на этом шаге до того как истёк таймаут $timeoutPrecommit$ и получены любые голоса с весом более 2/3 от общего веса, то перейти на шаг Propose(height, round+1).

Сделаем небольшое замечание. Как видно, для каждого шага наложено временное ограничение

$timeoutPropose$, $timeoutPrevote$, $timeoutPrecommit$ соответственно. Они зависят от номера раунда и увеличиваются на некоторую величину с увеличением номера раунда.

III. Доказательство корректности работы алгоритма консенсуса

Рассмотрим множество валидаторов с общим весом голоса $n = 3f + 1$, где f – максимальный вес голоса валидаторов, не следующих алгоритму консенсуса. Докажем, что корректность алгоритма при условии, что $f < \frac{1}{3}n < f + 1$.

Лемма 1. *Пусть заданы два множества валидаторов - U_1 и U_2, каждое из которых имеет общий вес $2f + 1$. Тогда в пересечении этих множеств существует хотя бы один валидатор, следующий алгоритму консенсуса.*

Доказательство. Для мощности множеств U_1 и U_2 валидаторов справедливо равенство:

$$|U_1| + |U_2| = 4f + 2 = 3f + 1 + f + 1 = n + f + 1,$$

поэтому

$$|U_1 \cap U_2| \geq f + 1 > f.$$

Из последнего можно заключить существование процесса, следующего алгоритму консенсуса.□

Лемма 2. *Пусть $f + 1$ валидаторов, следующих алгоритму консенсуса, имеют $lockedblock == v$, $lockedround == r_0$. Тогда $\forall r > r_0$ в голосе $Prevote$ этих процессов всегда указан либо блок v, либо $NULL$.*

Доказательство. Доказательство будем вести по индукции. Рассмотрим $r = r_0 + 1$. Согласно алгоритму консесуса на шаге Prevote выполняется $lockedround > -1$ и $round = lockedround + 1$, то есть истинно условие, на основании которого валидатор голосует за блок $lockedblock == v$. Доказали базис индукции.

При этом можно заметить, что на раунде $r_0 + 1$, количество Prevote, для которых значение блока не равно v и не равно $NULL$ будет не больше $n - f - 1 = 2f < \frac{2}{3}n$. Значит выполняются следующие условия:

1) валидаторы, следующие алгоритму консенсуса, не изменят значение $lockedround$ на шаге Precommit и в голосе Precommit укажут $NULL$;
2) валидатор, следующий алгоритму консенсуса, и получивший более $\frac{2}{3}n$ голосов Prevote за блок v, выполнит присваивания $lockedblock \coloneqq v, lockedround \coloneqq r$, а в том случае, если голоса за этот блок не были получены, то значения $lockedblock$ и $lockedround$ не изменятся.

Рассмотрим теперь шаг индукции. Пусть утверждение леммы выполнено для некоторого $r_1 > r_0$. Рассмотрим $r = r_1 + 1$

Согласно условиям а) и б), в течение раундов $r' : r_0 < r' \leq r_1$ либо не существует блока $v' != v$, за который проголосовали более $\frac{2}{3}n$ валидаторов, либо даже если такой блок существует, то перехода к Commit(height+1) выполнено не было. Тогда валидаторы из условия леммы, следующие шагу Prevote, не выполнят присваивание $lockedround \coloneqq -1$. А значит дальнейшие рассуждения полностью аналогичны рассуждениям из доказательства базы индукции. □

Теорема 1. *Алгоритм консенсуса корректен, то есть не существует двух различных валидаторов, следующих алгоритму консенсуса, которые на одной и той же высоте добавляют два разных блока (в итоге обеспечивается целостность цепочки блокчейна).*

Доказательство. Пусть на некотором раунде r валидатор V получил более $\frac{2}{3}n$ голосов Precommit за блок v_V, а валидатор W получил более $\frac{2}{3}n$ голосов Precommit за блок v_W на раунде $r_0 \geq r$. Рассмотрим случай, когда $r_0 = r$. Тогда, согласно лемме 1 существует хотя бы один валидатор, следующий алгоритму консенсуса, который отправил Precommit и для v_V, и для v_W в раунде r. Но по алгоритму консенсуса Precommit в раунде отправляется 1 раз. Тогда $v_V == v_W$.

Пусть теперь $r_0 > r$. На раунде r валидатор V получил более $\frac{2}{3}n$ голосов валидаторов, то есть не менее $2f + 1$ сообщений Precommit за блок v, значит не менее $2f + 1 - f = f + 1$ валидаторов, следующих алгоритму консенсуса, имеют: $lockedblock = v_V$, $lockedround = r_0$. Тогда по лемме 2 на раунде r_0 они отправили голос Prevote за блок v_V или $NULL$.

Валидатор W получил $2f + 1$ голосов Precommit за блок v_W. Среди $2f + 1$ валидаторов есть те, которые следуют алгоритму консенсуса. Рассмотрим любой из этих валидаторов. Согласно алгоритму консенсуса, он мог отправить голос Precommit за блок v_W тогда и только тогда, когда получил не менее $2f + 1$ голосов Prevote за этот блок. При этом среди $3f + 1$ голосов Prevote $f + 1$ из них были за блок v_V. Но $f + 1 + 2f + 1 = 3f + 2 > 3f + 1$, поэтому $v_V = v_W$. □

Теорема 2. *Алгоритм консенсуса валиден, то есть валидатор, следующий алгоритму консенсуса, внесет блок, удовлетворяющий требованиям блокчейна.*

Доказательство. Проверка требований, которые накладываются на блок, в Tendermint является внешней функцией по отношению к модулю, в котором указан алгоритм консенсуса, поэтому принимаем то, что если эта функция вызывается и возвращает true, то блок требованиям удовлетворяет. Этот вызов происходит во время шага Prevote. Так как в алгоритме максимальное количество валидаторов, которые не работают согласно алгоритму консенсуса, равно $f < 2f + 1$, то невозможно получить $2f + 1$ голосов Prevote за блок, неудовлетворяющий требованиям. Тогда валидаторы, следующие алгоритму консенсуса, на любом раунде не будут отправлять голос Precommit за блок, неудовлетворяющий требованиям, и по аналогии получаем, что невозможно получить $2f + 1$ голосов Precommit за блок, неудовлетворяющий требованиям.□

Теорема 3. *Алгоритм консенсуса конечен, то есть валидатор, следующий алгоритму консенсуса, добавит блок за конечное время.*

Доказательство. В начале каждого последующего раунда длительность каждого шага увеличивается. Это означает, что валидатор через конечное число раундов сможет получить все голоса. При этом, если валидаторы

находятся на разных раундах, то за счет общих условий перехода валидатор перейдет на наибольший раунд, получив голоса, указанные в переходах. Остается ситуация, когда валидатор, следующий алгоритму консенсуса, не получает не менее $2f+1$ голосов Precommit за некоторый блок v. Рассмотрим 2 ситуации. Первая – пусть к началу раунда не существует двух множеств валидаторов U_1 и U_2, что $lockedblock_1 != lockedblock_2$ и $|U_1| > \frac{1}{3}n, |U_2| > \frac{1}{3}n$.

Валидатор, следующий алгоритму консенсуса, отправляет голос Precommit за блок v тогда и только тогда, когда он получил не менее $2f+1$ голосов Prevote за этот же блок. Рассмотрим некоторый валидатор, следующий алгоритму консенсуса, который отправил Precommit за блок v. За этот же блок получено менее $2f$ других голосов Precommit, блоков, следующих алгоритму консенсуса, что не менее $2f+1$. Тогда существует другой валидатор, следующий алгоритму консенсуса, который проголосовал Precommit за блок $v' != v$. Получаем, что эти валидаторы получили не менее $2f+1$ голосов Prevote за блоки v и v'. Тогда не менее $2f+1-f=f+1$ валидаторов, следующих алгоритму консенсуса, имеют: $lockedblock = v$, $lockedround = r$. И еще не менее $f+1$ валидаторов, следующих алгоритму консенсуса, имеют: $lockedblock = v'$, $lockedround = r$. Сумма мощностей этих множеств больше или равна $f+1+f+1=2f+2>2f+1$, все валидаторы этих множеств следуют алгоритму консенсуса, кроме того, пересечение этих множеств непустое, поэтому $v'=v$. Получаем противоречие с тем, что не существует раунда, на котором получено не менее $2f+1$ Precommit за определенный блок.

Вторая ситуация: к началу раунда существуют два множества валидаторов U_1 и U_2, для которых $lockedblock_1 != lockedblock_2$ и $|U_1| > \frac{1}{3}n$, $|U_2| > \frac{1}{3}n$. Такая ситуация возникает в случае, когда валидаторы не успевают получить все голоса и срабатывают таймауты.

Тогда на некотором раунде будет Proposer, который отправит блок из $lockedblock$ и согласно шагу Prevote блоки одного из этих множеств валидаторов выполнят присваивание $lockedround := -1$, т.е. возникает первая ситуация.□

IV. Анализ механизма Proof of Lock Change

В описанном протоколе консенсуса Tendermint валидатор, который на этом раунде становится Proposer'ом, в случае, если $lockedround != -1$, должен отправить блок из $blockedblock$ и номер раунда, на котором он был получен. Номер раунда передаётся в сообщении, которое называется ProofOfLock. Далее на шаге Prevote, если валидатор видит, что $ProofofLock > lockedround > -1$, то он должен убедиться, что действительно за текущий блок получено более $\frac{2}{3}n$ от общего веса голосов Prevote. Видится, что эта проверка является лишней (отказ от пункта 3 в условии шага Prevote). Действительно, в дальнейшем валидатор переходит на шаг Precommit только в том случае, если за блок проголосовало более $\frac{2}{3}n$ валидаторов. Однако это не так. Рассмотрим две различные вариации протокола, в которых механизм Proof of Lock Change не используется:

a) Не выполняется проверка пунктов 1)-3) на шаге Prevote, то есть полностью убрано поле ProofOfLock.

Тогда в новой реализации получаем, что на шаге Prevote отправляется голос либо за блок, указанный в $lockedblock$, если $lockedround > -1$, либо за полученный блок, удовлетворяющий всем требованиям системы, если $lockedround == -1$, либо не отдаёт свой голос ни одному блоку, отправляя в систему $NULL$.

Предположим, что к некоторому раунду возникло 2 множества валидаторов U_1 и U_2, у которых $lockedblock_1 \neq lockedblock_2$. Не ограничивая общности, будем считать, что $lockedround_1 < lockedround_2 < round$. Пусть также $|U_1| > \frac{1}{3}n, |U_2| > \frac{1}{3}n$. Такие множества могут образоваться за счет того, что валидаторы не будут успевать получать все голоса других валидаторов.

Затем после окончания раунда увеличится время длительности шагов, но для любого раунда будут два множества, которые всегда будут голосовать за свой блок, так как в модификации алгоритма изменение $lockedround$ может быть произведено только на шаге Precommit, но никогда не будет более $\frac{2}{3}n$ голосов хотя бы за один из конкурирующих блоков. В итоге нарушается условие конечности алгоритма консенсуса. Таким образом, увеличение длительности шагов теоретически может привести к нештатной ситуации.

b) Не выполняется проверка пункта 3) на шаге Prevote, то есть ProofOfLock посылается, но отсутствует механизм Proof of Lock Change.

Аналогично пункту а) предположим возникновение двух множеств U_1 и U_2 с теми же условиями. Пусть теперь $Proposer$ на некотором раунде не последует алгоритму консенсуса и отправит валидаторам блок $lockedblock_1$ с раундом $lockedround_2$. Тогда валидаторы из множества U_1, согласно алгоритму, меняют значение $lockedround := lockedround_2$, а валидаторы из множества U_2 не меняют значения, так как они уже «заблокированы» на $lockedround_2$. Получаем два множества U_1 и U_2, которые для одного и того же раунда $lockedround$ имеют разные $lockedblock$.

Теперь валидаторы из U_1 будут каждый раунд голосовать за $lockedblock_1$, а из U_2 за $lockedblock_2$, так как они заблокированы на этих блоках. Разблокировка возможна только в случае, если будет сформирован новый $ProofofLock$, т.е. набраны $\frac{2}{3}n$ $Prevote$ голосов за какой-либо блок, либо $\frac{2}{3}n$ голосов за $NULL$. Это невозможно из-за того, что ни значение $NULL$, ни некоторый валидный блок не наберут больше $n-\min(|U_1|,|U_2|) < \frac{2}{3}n$ голосов. Но отправка голоса на шаге $Precommit$ будет произведена только если за

некоторый блок получено более $\frac{2}{3}n\ Prevote$ голосов, т.е. не будет более $\frac{2}{3}n\ Precommit$ голосов за некоторый блок и опять нарушается условие конечности.

Таким образом, при отсутствии механизма Proof Of Lock Change, нарушается условие конечности алгоритма консенсуса.

V. Заключение

В работе рассмотрен алгоритм консенсуса Tendermint[1] и механизм Proof Of Lock Change, применяемый в этом алгоритме. Было доказано, что алгоритм конечен, валиден и корректен. Кроме того, в работе установлено, что критически важной частью этого алгоритма является механизма Proof Of Lock Change, т.к. отказ от него ведёт к потенциальному нарушению конечности алгоритма консенсуса.

Библиография

[1] www.tendermint.com: официальный сайт. URL: https://www.tendermint.com/ (дата обращения: 03.04.2019).
[2] www.tendermint.com/docs/: официальная документация проекта. URL: https://www.tendermint.com/docs/spec/consensus/consensus.html (дата обращения: 03.04.2019).
[3] Buchman E., Kwon J., Milosevic Z. The latest gossip on BFT consensus [электронный ресурс]. 2018. 24 сен. 14 с. // arxiv.org: электронный архив. 1991. авг. URL: https://arxiv.org/pdf/1807.04938.pdf (дата обращения: 03.04.2019).
[4] Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System [электронный ресурс]. 2008. 9 с. // bitcoin.org: Bitcoin - Open source P2P money URL: https://bitcoin.org/bitcoin.pdf (дата обращения: 03.04.2019).
[5] Wood G. Ethereum: a secure decentralised generalised transaction ledger [электронный ресурс]. 32 с. // gavwood.com: сайт разработчика URL: https://gavwood.com/paper.pdf (дата обращения: 03.04.2019).

Consensus algorithm of Tendermint platform and Proof Of Lock Change mechanism

Ilya Gerasimov, Ivan Chizhov

***Abstract*—This work represents a consensus algorithm in a form of Tendermint's realization which is based on Byzantine Generals problem. This algorithm is used in order to achieve consensus among the participants over the next block in blockchain in decentralized applications and payment systems with data representation in the form of blockchain. The main feature of the Tendermint consensus algorithm is that the protocol participants are special network nodes, who are called validators. The paper describes the consensus algorithm, data structures representing validator voices with some weight, and the Proof Of Lock Change mechanism. It has been proved the finiteness, correctness and validity of this algorithm in condition the number of validators which are isolated from the system or perform acts which do not correspond to the consensus algorithm does not exceed one third of the total number of validators. Analysis of Proof of lock Change mechanism was performed. It is shown that all possible variations of the consensus algorithm in which Proof of lock Change mechanism is not implemented have a serious defect the violation of finiteness property of consensus algorithm, which can lead to system failure.**

***Keywords*—blockchain, consensus protocol, tendermint, proof of lock change mechanism**

REFERENCES

[1] www.tendermint.com: official site. URL: https://www.tendermint.com/ (data obrashhenija: 03.04.2019).

[2] www.tendermint.com/docs/: official documentation. URL [jelektronnyj resurs]: https://www.tendermint.com/docs/spec/consensus/consensus.html (data obrashhenija:: 03.04.2019).

[3] Buchman E., Kwon J., Milosevic Z. The latest gossip on BFT consensus [jelektronnyj resurs]. 2018. URL: https://arxiv.org/pdf/1807.04938.pdf (data obrashhenija: 03.04.2019).

[4] Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System [jelektronnyj resurs]. 2008. 9 c. // bitcoin.org: Bitcoin - Open source P2P money URL: https://bitcoin.org/bitcoin.pdf (data obrashhenija: 03.04.2019).

[5] Wood G. Ethereum: a secure decentralised generalised transaction ledger [jelektronnyj resurs]. 32 c. // gavwood.com: official site URL: https://gavwood.com/paper.pdf data obrashhenija: 03.04.2019).