АВТОМАТИЗАЦИЯ ОБРАБОТКИ ТЕКСТА

УДК 004.75.056

А.Ю. Щербаков

О разработке средств для формирования корпоративного распределенного реестра (блокчейн)

Обсуждается проблема разработки средств для формирования атомов корпоративного распределенного реестра (блокчейна), обеспечивающего защищенное распределенное хранение данных в связанной цепочке, рассматривается пример применения предлагаемых средств для реализации условного налогового платежа через смарт-контракт.

Ключевые слова: *блокчейн, распределенное хранение данных, информационная безопасность, электронное голосование, шифрование, контроль целостности, смарт-контракт, средства разработки для блокчейн*

ВВЕДЕНИЕ

Блокчейн как системная целостность должен состоять из отдельных элементов – звеньев, каждое из которых в свою очередь делится на элементарные компоненты (в [1] они названы **атомами блокчейна**). Атомы как элементы, ассоциированные или встроенные в компьютерную систему, могут быть пассивными и активными.

СТРУКТУРА БЛОКЧЕЙНА

Атомы блокчейна характеризуются идентификатором (именем) и имеют заданные свойства, необходимые как для формирования звеньев и всей цепочки, так и для выполнения свойств компьютерной (информационной) системы, которая использует блокчейн, например, свойства конфиденциальности пользовательских данных, помещаемых в распределенном реестре. Необходимо отметить, что существующие в настоящее время системы блокчейна не обеспечивают свойства конфиденциальности информации.

Выделим следующие типы атомов блокчейна.

1. Атом-граница (начало или конец) блокчейна, содержит ссылки на окончание предыдущего звена или начало следующего.

2. Атом-структура – определяет и описывает вектор идентификаторов атомов, имеющихся в звене и логично следует за атомом-граница (начало).

3. Атом-данные – содержит описание данных, интерпретируемых как пассивный компонент звена. Разновидностью атомов-данных являются:

- атом открытые данные (атом ОД) – характеризуется длиной и содержит данные, не подвергающиеся никакой обработке;
- атом целостные данные (атом ЦД) – характеризуется длиной данных зафиксированной целостности и идентификатором или ссылкой на процедуру контроля целостности данных;
- атом подписанные данные (атом ПД) – характеризуется длиной подписанных данных – идентификатором или ссылкой на процедуру проверки подписи под данными, а также ссылкой на ключ проверки подписи. Процедура проверки подписи может быть заданной внешне относительно блокчейна или быть атомом-субъектом;
- атом зашифрованные (закрытые) данные (атом ЗД) – характеризуется длиной зашифрованных данных – идентификатором или ссылкой на процедуру зашифрования/расшифрования данных, а также на ключ зашифрования или расшифрования;
- атом-подпись – содержит подпись атома или нескольких атомов – «подписанные данные» с заданным идентификатором (идентификаторами);
- атом-хеш – содержит эталон контроля целостности для атома или нескольких атомов «целостные данные» с заданным идентификатором (идентификаторами).

4. Атомы-субъекты – содержит описание данных, интерпретируемых как активный компонента звена.

Атом-субъект может быть:

- атомом-сценарием – содержащим интерпретируемый код для выполнения операций над атомами-данным;
- атомом-экзекутором – содержащим скомпилированный код для реального процессора, гипервизора компьютерной системы или атома-машины, в которой обрабатывается блокчейн;

- атомом-машиной – содержащим среду для выполнения атома-сценария или атома-экзекутора.

Далее рассмотрим элементарные модули, необходимые для формирования соответствующих атомов. Важно заметить, что в нашем случае речь идет о корпоративном распределенном реестре, поскольку в блокчейнах общего назначения атомы формируются иным образом. Однако предлагаемая технология позволяет защищать данные и в блокчейнах общего назначения, при этом информация предварительно формируется при помощи рассматриваемых здесь модулей, а затем уже помещается в блокчейн.

ЭЛЕМЕНТАРНЫЕ МОДУЛИ ДЛЯ ФОРМИРОВАНИЯ АТОМОВ КОРПОРАТИВНОГО БЛОКЧЕЙНА

Модуль первичного формирования исходного случайного числа и персонального идентификатора для пользователя

InitUser

Формат использования:

InitUser FileUserID UserPIN <RandomString>,

где:

FileUserID – имя файла с закрытым на пароле персональным идентификатором пользователя (может быть связано с именем пользователя),

UserPIN – пароль (пин-код или метод его ввода, например, чтения из USB-токена) для закрытия персонального идентификатора пользователя,

<RandomString> – необязательный параметр для улучшения работы датчика случайных чисел («разгонная строка»).

Модуль создает файл random.bin для дальнейшего использования датчика случайных чисел и файл с закрытым на пароле персональным идентификатором пользователя (фактически – защищенный контейнер для хранения и передачи персонального идентификатора пользователя).

Модуль возвращает типизированные ошибки, необходимые для интеграции вызовов модулей в смарт-контракт.

Возвращаемые ошибки:

-1 – ошибка при тестировании модулей защиты,

-2 – ошибка формата вызова,

-3 – файл идентификатора уже существует,

-4 – ошибка формирования случайного числа,

-5 – ошибка обновления случайного числа,

-6 – ошибка записи файла пользователя,

-7 – ошибка контрольного чтения файла пользователя.

Модули формирования атомов блокчейна

GenAtomX

Модуль формирует атом-закрытые данные из открытых данных с использованием персонального идентификатора пользователя – атом типа X.

Формат использования:

GenAtomX FileUserID UserPIN file_or_string AtomFile,

где:

FileUserID – имя файла с закрытым на пароле персональным идентификатором пользователя,

UserPIN – пароль для закрытия персонального идентификатора пользователя,

file_or_string – информация для помещения в атом-ЗД (закрытые данные), закрываемая на персональном идентификаторе пользователя,

AtomFile – файл, в который записывается атом.

Модуль использует файл с закрытым на пароле персональным идентификатором пользователя, открывает его и строит атом блокчейна согласно протоколу.

Возвращаемые ошибки:

-1 – ошибка при тестировании модулей защиты,

-2 – ошибка формата вызова,

-3 – файл идентификатора пользователя не существует,

-4 – ошибка формирования случайного числа,

-5 – атом-файл уже существует,

-6 – неверный пин-код,

-7 – ошибка записи атома,

-8 – ошибка контрольного чтения атома,

-9 – ошибка обновления случайного числа.

GenAtomY

Модуль формирует атом-закрытые данные из открытых данных с использованием случайных данных для достижения заданного времени (PoW) – атом типа Y.

Формат использования:

GenAtomY file_or_string AtomFile Power,

где:

file_or_string – информация для помещения в атом-ЗД (закрытые данные), закрытая на случайном числе заданной длины,

AtomFile – файл, в который записывается атом,

Power – двузначное число, обеспечивающее PoW (для тестирования рекомендуется значение 18-22).

Возвращаемые ошибки:

-1 – ошибка при тестировании модулей защиты,

-2 – ошибка формата вызова,

-4 – ошибка формирования случайного числа,

-5 – ошибка обновления случайного числа,

-6 – атом-файл уже существует,

-7 – ошибка записи атома,

-9 – ошибка обновления случайного числа.

GenAtomH

Модуль формирует атом-хеш (атом целостные данные) из открытых данных с использованием персонального идентификатора пользователя – атом типа H.

Формат использования:

GenAtomX FileUserID UserPIN file_or_string AtomFile,

где:

FileUserID – имя файла с закрытым на пароле персональным идентификатором пользователя,

UserPIN – пароль для закрытия персонального идентификатора пользователя,

file_or_string – информация для помещения в атом-хеш, функция от открытых данных, вычисленная при помощи персонального идентификатора пользователя,

AtomFile – файл, в который записывается атом.

Возвращаемые ошибки:

-1 – ошибка при тестировании модулей защиты,

-2 – ошибка формата вызова,

-3 – файл идентификатора пользователя не существует,

-4 – ошибка формирования случайного числа,
-5 – атом-файл уже существует,
-6 – неверный пин-код,
-7 – ошибка записи атома,
-8 – ошибка контрольного чтения атома,
-9 – ошибка обновления случайного числа.

Модули извлечения данных из звеньев блокчейна

ExcX

Модуль извлечения данных из атома типа X.

Формат использования:

ExcX FileUserID UserPIN AtomFile Result,

где:

FileUserID – имя файла с закрытым на пароле персональным идентификатором пользователя,

UserPIN – пароль для закрытия персонального идентификатора пользователя,

AtomFile – файл, в который помещен атом ЗД,

Result – файл с восстановленными данными.

Модуль имеет назначение тестирования пользователем своих действий и аудита пользователем транзакций.

Возвращаемые ошибки:

-1 – ошибка при тестировании модулей защиты,
-2 – ошибка формата вызова,
-3 – файл идентификатора пользователя не существует,
-4 – ошибка формирования случайного числа,
-5 – атом-файл не существует,
-6 – неверный пин-код,
-7 – ошибка записи атома,
-8 – ошибка контрольного чтения атома.

ExcY

Модуль извлечения данных из атома типа Y.

Формат использования:

ExcY AtomFile Result,

где:

AtomFile – файл, в который помещен атом ЗД,

Result – файл с восстановленными данными.

Модуль имеет назначение – выделение закрытых данных с заданной трудоемкостью PoW.

Возвращаемые ошибки:

-1 – ошибка при тестировании модулей защиты,
-2 – ошибка формата вызова,
-3 – файл идентификатора пользователя не существует,
-4 – ошибка формирования случайного числа,
-5 – атом-файл не существует,
-7 – ошибка записи результата.

CAtomH

Модуль проверяет целостность данных при помощи заранее вычисленного атом-хеш с использованием персонального идентификатора пользователя.

Формат использования:

GenAtomX FileUserID UserPIN file_or_string AtomFile,

где:

FileUserID – имя файла с закрытым на пароле персональным идентификатором пользователя,

UserPIN – пароль для закрытия персонального идентификатора пользователя,

file_or_string – информация для контроля целостности,

AtomFile – файл, в который был записан атом, содержащий контрольную информацию от данных.

Возвращаемые ошибки:

-1 – ошибка при тестировании модулей защиты,
-2 – ошибка формата вызова,
-3 – файл идентификатора пользователя не существует,
-4 – ошибка формирования случайного числа,
-5 – атом-файл уже существует,
-6 – неверный пин-код,
-7 – ошибка записи атома,
-8 – ошибка контрольного чтения атома,
-9 – ошибка обновления случайного числа.

Модуль вычисления Power

Cpower

Формат использования

Cpower Time

Модуль возвращает число бит, которое необходимо для обеспечения трудоемкости в Time минут.

Все модули генерируют одноименные названию модуля текстовые журналы.

ПОНЯТИЕ СМАРТ-КОНТРАКТА

Первые идеи смарт-контрактов были предложены в 1996 г. Ником Сабо [2]. Практические реализации стали возможными, благодаря появлению в 2008 г. технологии блокчейн. Некоторые принципы смарт-контрактов были заложены в протоколе первой блокчейн-валюты (криптовалюты) *Bitcoin*, однако они не были реализованы в клиентском программном обеспечении, не обладали полнотой по Тьюрингу из соображений безопасности и не использовались на практике. С развитием технологии блокчейн, стали высказываться идеи, что поверх протокола биткойна могут быть созданы различные протоколы более высокого уровня, включая полноценные смарт-контракты, по аналогии с тем как поверх TCP/IP существует множество протоколов прикладного уровня.

Смарт-контракты впервые стали применяться на практике в проекте *Ethereum* (рус. Эфириум). Идея создания проекта появилась в 2013 г. В тот момент основатель журнала «*Bitcoin Magazine*» В. Бутерин пришёл к выводу, что биткойн плохо подходит в качестве базового протокола, поскольку изначально не был спроектирован под данную задачу, и написал в одной из своих статей об идее создания такого протокола с нуля.

Будем полагать, что смарт-контракт – исполняемый код, у которого зафиксирована целостность, помещенный в атом-экзекутор и оперирующий с атомами и звеньями блокчейна. При этом результат работы смарт-контракта всегда помещается в новый атом или звено блокчейна. В противном случае идеология неизменности звеньев блокчейна будет нарушена.

ПРИМЕР ФОРМИРОВАНИЯ И РАБОТЫ СМАРТ-КОНТРАКТА, ПОСТРОЕННОГО ИЗ ЭЛЕМЕНТАРНЫХ МОДУЛЕЙ

Полагаем, что пользователь nlp100 («налогоплательщик-100») имеет кошелек с криптовалютой (для оплаты налогов) cnal100, на котором имеется 500 крипторублей.

Для тестового создания кошелька используем смарт-контракт:

genatomx nlp100_1 privet1 500 cnal100_u

genatomx nlp100_3 privet3 500 cnal100_b.

В реальной платежной системе для создания кошелька будет необходимо выполнить согласованные с уполномоченным банком или оператором криптовалюты действия по идентификации пользователя, созданию кошелька и зачислению криптовалюты на него.

Пользователь *fns* (налоговая служба) выставил «налогоплательщику-100» налоговый платеж в 50 крипторублей в адрес банка *bank*, который оформил в виде смартконтракта *fns*100_b.*bat*.

Первично пользователь заводит свои контейнеры закрытого ключа по работе с кошельком nlp100_1 для получения смартконтрактов и извещений от налоговой службы nlp100_2 и для обмена с банком nlp100_3 при помощи модуля *inituser*.

Смартконтракт представляет собой последовательный вызов методов создания атомов блокчейна *genatomx*, которые списывают с кошелька налогоплательщика-100 50 крипторублей и зачисляют их на счет банка, формируя атомы a1 и a2:

genatomx nlp100_1 %1 -50 a1

genatomx nlp100_3 %2 +50 a2.

Необходимо заметить, что для работы с кошельком налогоплательщика требуется предъявление пароля или приватного ключа, что указано аргументом %1 (пароль или ключ вводится пользователем), соответственно, второй пароль или приватный ключ нужен для зачисления средств банку.

Налоговая служба заводит контейнер для обмена с «налогоплательщиком-100» – *fns*100, банк заводит контейнер для обмена с «налогоплательщиком-100» – *bank*100, банк для направления квитанций налоговой создает контейнер *bfns*.

Для создания контейнеров используется следующий смартконтракт:

inituser nlp100_1 privet1

inituser nlp100_2 privet2

inituser nlp100_3 privet3

inituser fns100 privet2

inituser bank100 privet3

inituser bfns privet4.

Пароли или методы ввода приватного ключа используются для доступа пользователя к своему кошельку – *privet*1 – приватный ключ, неизвестный никому, кроме пользователя, *privet*2 – для обмена с налоговой службой, *privet*3 – для обмена пользователя с банком и *privet*4 – для обмена банка с налоговой службой.

Налоговая служба формирует атом a0, в котором находится зашифрованный текст смартконтракта *fns*100_b.*bat*:

genatomx nlp100_2 *privet*2 *fns*100_b.*bat* a0.

Налогоплательщик экстрагирует из атома смартконтракт:

excx nlp100_2 *privet*2 a0 *user*100.*bat*

и выполняет контракт *user*100.*bat*, вводя вместо %1 и %2 пароли или приватные ключи для доступа к своему кошельку и для передачи транзакции в банк.

Контракт создает звенья блокчейна a1 и a2, в которых находятся измененное состояние кошелька налогоплательщика (уменьшение на 50 крипторублей) и распоряжение по зачислению необходимой суммы на кошелек банка.

Банк экстрагирует из атома распоряжение налогоплательщика по счета

excx nlp100_3 privet3 a2 b1.

Банк зачисляет деньги на свой кошелек (криптосчет) и высылает подтверждение налогоплательщику путем формирования очередного атома

genatomx nlp100_3 privet3 zachisleno=50 a3.

На стороне налогоплательщика автоматически списывается с кошелька зачисленная в банк сумма:

genatomx nlp100_1 privet1 450 cnal100_u

genatomx nlp100_3 privet3 450 cnal100_b.

Новая копия кошелька создается со значением 450=500-50 по уведомлению банка.

Банк имеет возможность проверить в *cnal*100_b, что сумма в кошельке изменилась:

excx nlp100_3 privet3 cnal100_b cnal100_t2.

На стороне клиента (налогоплательщика) операции с кошельком выполняются только при балансе средств – когда в кошельке клиента и банка одинаковые суммы.

Баланс проверяется при помощи следующего смартконтракта:

excx nlp100_1 privet1 cnal100_u cnal100_t1

excx nlp100_3 privet3 cnal100_b cnal100_t2.

Последняя операция – формирование атома-подтверждения a4:

genatomx bfns privet4 prinjato=nlp100=50 a4.

Налоговая служба экстрагирует атом a4

excx bfns privet4 a4 b3

и получает строку подтверждения оплаты налога.

Для реального использования необходимо ввести также подтверждение банковского платежа в казначейство.

Для демонстрации системы голосования можно выполнить следующий смартконтракт:

genatomx nlp100_1 privet1 Ivan_Ivanov g1

genatomx nlp100_2 privet2 Petr_Petrov g2

genatomy Ivan_Ivanov g3 19

genatomy Petr_Petrov g4 20

excx nlp100_1 privet1 g1 golos11

excx nlp100_2 privet2 g2 golos21

excy g3 golos12

excy g4 golos22.

Для удобства используются уже созданные контейнеры.

Сначала формируются голоса за *Ivan_Ivanov* и *Petr_Petrov*, которые сформированы и доступны только проголосовавшим пользователям, а затем – доступны всем, но с раскрытием в заданное время, причем для *Petr_Petrov* время раскрытия в среднем в два раза больше. Затем голоса раскрываются в соответствующие файлы и сравниваются.

Журнал функции *excy* показывает примерное различие времен экстрагирования голосов – приблизительно 12 секунд и 31 секунда при одинаковом проценте перебора (около 30):

00:39:01 05.02.2018:Start excy

00:39:01 05.02.2018:Success Protect Function

00:39:01 05.02.2018:Ok Test Random
00:39:13 05.02.2018:RealCicle = 157705.000000
[Full cicles = 524288.000000] 30.079842 percent of full cicles
Full time = 11.875000 sec
00:39:13 05.02.2018:Ok AtomFileY
00:39:13 05.02.2018:Start excy
00:39:13 05.02.2018:Success Protect Function
00:39:13 05.02.2018:Ok Test Random
00:39:44 05.02.2018:RealCicle = 410876.000000
[Full cicles = 1048576.000000] 39.184189 percent of full cicles
Full time = 30.861000 sec
00:39:44 05.02.2018:Ok AtomFileY.

ВЫВОДЫ

Предлагаемая технология позволяет формировать практически весь необходимый спектр атомов блокчейна и выполнять операции с ними, используя технологию смарт-контрактов. Криптографические операции с атомами целесообразно реализовывать на основе отечественных криптоалгоритмов для обеспечения свойств доверенности и возможности последующей сертификации и аттестации разработанных решений у государственных регуляторов.

СПИСОК ЛИТЕРАТУРЫ

1. Биктимиров М.Р., Домашев А.В., Черкашин П.А., Щербаков А.Ю. Блокчейн: универсальная структура и требования // Научно-техническая информация. Сер. 2. – 2017. – № 11. – С. 1-4; Biktimirov M.R., Domashev A.V., Cherkashin P.A., Shcherbakov A.Yu. Blockchain Technology: Universal Structure and Requirements // Automatic Documentation and Mathematical Linguistics. – 2017. – Vol. 51, № 6. – p. 235 -238.
2. Ник Сабо. Умные контракты (Четвертая революция стоимости) // Компьютерра. – 1998. – № 38. – С. 12-19.

Материал поступил в редакцию 19.02.18

Сведения об авторе

ЩЕРБАКОВ Андрей Юрьевич – доктор технических наук, профессор, главный научный сотрудник Федерального исследовательского центра «Информатика и управление» РАН, профессор МИЭМ НИУ ВШЭ, член Ученого совета ВИНИТИ РАН, ведущий специалист по информационной безопасности Технопарка «Техпромбизнес», консультант по криптографии фирмы «Лаборатория Касперского», Москва
e-mail: x509@ras.ru