СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ

УДК 004.056.5

П.И. СТЕЦЕНКО, Г.З. ХАЛИМОВ, д-р техн. наук

АНАЛИЗ БЕЗОПАСНОСТИ СИСТЕМЫ СМАРТ-КОНТРАКТОВ С СОХРАНЕНИЕМ КОНФИДЕНЦИАЛЬНОСТИ

Введение

Построенные над децентрализованными криптовалютами системы смарт-контрактов на основе технологии Blockchain позволяют сторонам при отсутствии взаимного доверия безопасно совершать сделки друг с другом, не привлекая третьей доверенной стороны. В случае договорных нарушений или преждевременного завершения выполнения условий смарт-контракта технология Blockchain гарантирует, что другие, не нарушающие условий, стороны получат соразмерное вознаграждение [1].

Несмотря на все преимущества технологии Blockchain и смарт-контрактов нынешняя форма этих технологий не обладает транзакционной конфиденциальностью. Вся последовательность действий, предпринятых в смарт-контракте, распространяется по сети и записывается в регистре Blockchain [2]. Таким образом, разработка децентрализованной системы смарт-контрактов с сохранением конфиденциальности на основе технологии Blockchain является крайне актуальной проблемой.

В данной работе проведен анализ безопасности системы смарт-контрактов с сохранением конфиденциальности на основе технологии Blockchain и представлена формальная модель такой системы смарт-контрактов. Модель совместима с концепцией универсальной компонуемости [3]. В разд. 1 приводится общее описание структуры системы смарт-контрактов с сохранением конфиденциальности на основе технологии Blockchain. В разд. 2 представлены шаблоны проектирования основных функциональностей системы. В разд. 3 приводится анализ безопасности системы смарт-контрактов с сохранением конфиденциальности на основе технологии Blockchain и описываются основные требования к безопасности смарт-контракта.

1. Определение структуры системы смарт-контрактов с сохранением конфиденциальности на основе технологии Blockchain

Определение 1 [4]**.** Смарт-контракт – это компьютерная программа с участием цифровых активов и двух или более сторон, где некоторые или все стороны кладут свои активы в данную программу и эти активы автоматически перераспределяются между сторонами в соответствии с формулой, основанной на определенных данных, которые не известны во время запуска контракта.

Определение 2 [5]**.** Blockchain – это хронологическая база данных транзакций, хранимая сетью компьютеров, распределенная между всеми узлами и синхронизирующаяся между всеми пользователями, реализована в виде последовательности блоков транзакций, каждый из которых содержит хэш предыдущего блока, связывая этот блок в качестве единственного предшественника (прообраза).

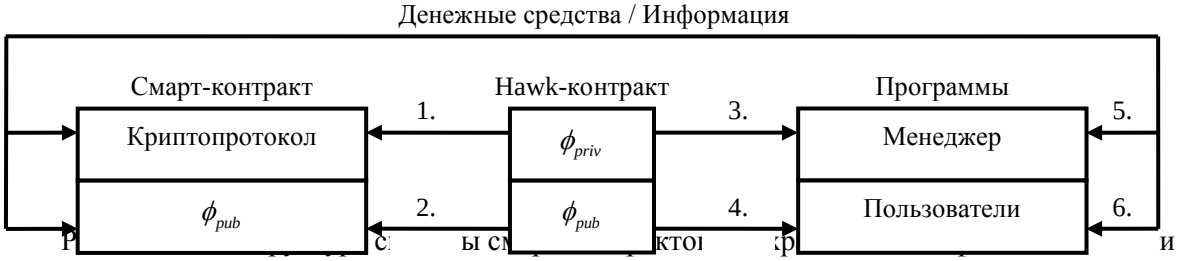
Определение 3. Смарт-контракт с сохранением конфиденциальности – это смарт-контракт, хранящий все пользовательские данные, кроме открытых условий, в зашифрованном виде на регистре Blockchain и не позволяющий установить истинные личности сторон, берущих в нем участие.

Рассмотренная система смарт-контрактов базируется на компиляторе Hawk, который автоматически компилирует программу с криптографическим протоколом между Blockchain и пользователями [4]. Структура системы смарт-контрактов с сохранением конфиденциальности на основе технологии Blockchain представлена на рис. 1.

Программа смарт-контракта состоит из двух частей:

- программа личного контракта, обозначаемая ϕ_{priv}, принимает входные данные сторон, а также денежные единицы (например, ставки в аукционе). Программа ϕ_{priv} выполняет вычисления, чтобы определить распределение выплат среди сторон. Например, на аукционе ставка победителя идет к продавцу, а чужие ставки возвращаются. Личный контракт ϕ_{priv} призван защитить данные участников и обмен денег;

- программа открытого контракта, которая обозначается ϕ_{pub}, не затрагивает личные данные или средства.



на основе технологии Blockchain

Система смарт-контрактов с сохранением конфиденциальности определяется следующими взаимодействиями:

1) компилятор Hawk компилирует программу личного контракта ϕ_{priv} в криптографический протокол, включающий в себя следующие части: логику протокола, которая должна выполняться с помощью Blockchain, и логику, которая будет выполняться сторонами контракта [4];

2) компилятор Hawk компилирует программу открытого контракта ϕ_{pub} на выполнение непосредственно на открытом регистре Blockchain.

Открытая часть смарт-контракта ϕ_{pub} передается пользователю. Пользователь осуществляет контроль над выполнением смарт-контракта на регистре Blockchain, может передавать денежные средства в смарт-контракт и получать его внутренние состояния.

Система смарт-контрактов с сохранением конфиденциальности на основе компилятора Hawk предполагает специальную договорную сторону – менеджера.

Менеджер имеет доступ к данным, находящимся в ϕ_{priv} для эффективного выполнения криптопротокола. Менеджер занимается вычислением распределения платежей, предусмотренных смарт-контрактом, и ему доступны внутренние состояния смарт-контракта.

Менеджер не может повлиять на исход договора. Если менеджер прерывает выполнение протокола, он может быть оштрафован, и пользователи получают вознаграждение в пропорциональном размере.

Менеджер не должен быть доверенным для поддержания безопасности или конфиденциальности базовой валюты, что предотвращает атаки «двойной траты», «инфляции», «деанонимизации».

Если несколько экземпляров контракта выполняются одновременно, каждый экземпляр контракта может устанавливать отдельного менеджера и последствия скомпрометированного менеджера ограничиваются одним экземпляром.

Роль менеджера может быть реализована при помощи использования доверенного вычислительного оборудования, такого как Intel SGX, или заменена многопользовательским вычислением [5, 6].

2. Шаблоны проектирования основных функциональностей системы смарт контрактов

Формализация Blockchain-модели опирается на структурные шаблоны проектирования. Структурные шаблоны проектирования реализуют модульность принципа проектирования. Структурные шаблоны проектирования обрабатывают множество общих деталей, таких как таймеры, псевдонимы и т.п.

Шаблоны реализуют набор общих функций (таймер, регистр, псевдонимы), которые применимы ко всем функциональным возможностям и контрактам в описываемой модели выполнения смарт-контрактов. Система обозначений разбивается на модули так, что общие детали не повторяются при описании идеальных функциональных возможностей и программ контрактов.

Предлагаемая модель опирается на три основные базовые функциональности: отдельная функциональность смарт-контракта Ω, идеальная функциональность Ψ, которая включает в себя функции, на которых базируется модель в целом, – это время, псевдонимы, открытый регистр и т.д.; и функциональность пользовательских протоколов Π, необходимая для осуществления модульного подхода в отношении этого сегмента модели.

Шаблон проектирования функциональности контракта $\Omega(C)$ охватывает программу смарт-контракта, обозначенную C, и реализует функциональные возможности контракта. Шаблон проектирования функциональности контракта формально приведен в табл. 1. Он реализует стандартные функции: таймер, глобальный регистр, денежные переводы между сторонами.

Таблица 1

Шаблон проектирования функциональности контракта $\Omega(C)$

Init: при инициализации выполняются следующие действия:

структура данных регистра $ledger[\overline{P}]$ содержит баланс счета стороны $\overline{P}$; отправить весь баланс $ledger$ злоумышленнику A;

устанавливается текущее время $T := 0$;

устанавливается очередь приема $rqueue := 0$;

далее запускается процедура инициализации для программы C;

внутреннее состояние программы C отправляется злоумышленнику A.

Tick: получив «tick»-сообщение от честной стороны, если функциональность собрала «tick»-подтверждения от всех честных сторон с момента последнего «tick», то:

подтверждается перестановка $perm$, осуществленная злоумышленником, для $rqueue$, чтобы переставить сообщения, полученные в предыдущем раунде;

следует вызов процедуры таймера $Timer$ программы C;

переупорядоченные сообщения пропускаются в программу C для обработки;

устанавливается $rqueue := 0$;

устанавливается $T := T + 1$.

Другие активации:

- *Аутентифицированный прием:* при получении сообщения $(authentificated, m)$ от стороны P:

отправить (m, P) злоумышленнику A;

поставить в очередь сообщение путем вызова $rqueue.add(m, P)$.

- *Прием под псевдонимом:* при получении сообщения вида $(pseudonymous, m, \overline{P}, \sigma)$ от любой стороны:

отправить $(m, \overline{P}, \sigma)$ злоумышленнику A;

интерпретировать $\sigma := (nonce, \sigma')$ и $Verify(\overline{P}.spk, (nonce, T, \overline{P}.epk, m), \sigma') = 1$;

если сообщение $(pseudonymous, m, \overline{P}, \sigma)$ не было получено ранее в этом раунде, ставится в очередь сообщение путем вызова $rqueue.add(m, \overline{P})$.

Продолжение табл. 1

- *Анонимный прием:* при приеме сообщения $(anonymous, m)$ от стороны P : отправить m злоумышленнику A ; если m не было получено ранее в этом раунде, ставится в очередь сообщение путем вызова $rqueue.add(m)$. **Перестановка (Permute)**: после получения $(\text{permute}, \text{perm})$ от злоумышленника A , записать $perm$.
Предоставление (Expose): при получении $exposestate$ от стороны P , вернуть внутреннее состояние функциональности стороне P . Следует отметить, что это также означает, что сторона может запросить функциональность для текущего времени T . **Операции регистра** (внутренняя активация): **Перевод (Transfer):** при приеме $(transfer, amount, \overline{P}_r)$ от некоторого псевдонима $\overline{P}_s$: утверждается, что $ledger[\overline{P}_s] \geq amount$ $ledger[\overline{P}_s] := ledger[\overline{P}_s] - amount$ $ledger[\overline{P}_r] := ledger[\overline{P}_r] + amount$.

Замечание 1.

1. Шаблон проектирования функциональности контракта $\Omega(C)$ обеспечивает корректность, но не конфиденциальность – предоставляет внутреннее состояние контракта любой стороне, делающей запрос.

2. Шаблон проектирования функциональности контракта $\Omega(C)$ позволяет сделать функциональность регулируемой по времени обработки: сообщения, полученные в одном раунде и поставленные в очередь, могут обрабатываются в следующем раунде. Порядок обработки этих операций определяется майнером, который вырабатывает следующий блок цепочки Blockchain.

3. Шаблон проектирования функциональности контракта $\Omega(C)$ фиксирует стремление злоумышленника. Предполагается, что злоумышленник может сначала увидеть все сообщения, отправленные контракту честными сторонами, а затем принять решение о своих сообщениях для этого раунда, а также о порядке, в котором контракт должен обрабатывать сообщения в следующем раунде. Такое моделирование стремления злоумышленника необходимо для предотвращения атак гибкости транзакций.

Шаблон проектирования идеальной функциональности $\Psi(idealP)$ (идеальная функциональность) включает идеальную программу, обозначаемую $idealP$. Шаблон определяет стандартные функции, такие как время, псевдонимы, открытый регистр и денежные переводы между сторонами. Шаблон проектирования идеальной функциональности Ψ формально представлен в табл. 2.

Таблица 2

Шаблон проектирования идеальной функциональности $\Psi(idealP)$

Init: при инициализации выполняются следующие действия: Время. Устанавливается текущее время $T := 0$. Устанавливается очередь приема $rqueue := 0$. Псевдонимы. Устанавливаются $nyms := \{(P_1, P_1)...(P_N, P_N)\}$, то есть, изначально истинная личность каждого участника записывается в качестве псевдонима по умолчанию для участника смарт-контракта. Регистр. Структура регистра $ledger[P]$ хранит баланс каждой личности $P \in \{P_1, ..., P_N\}$.

Продолжение табл. 2

Перед тем, как генерируются любые новые псевдонимы, только подлинные личности пополняют свой баланс. Массив $ledger[]$ отправляется идеальному злоумышленнику S.

idealP.init – запускается процедура инициализации для программы $idealP$.

Tick: получив «tick»-сообщение от честной стороны P, уведомляется S о $(tick, P)$. Если функциональность собрала «tick»-подтверждения от всех честных сторон с момента последнего «tick», то:

следует вызов процедуры таймера Timer программы $idealP$;

далее применяется перестановка, осуществленная злоумышленником $perm$ к $rqueue$ для изменения порядка обработки сообщений, полученных в предыдущем раунде;

для каждого $(m, \overline{P}) \in rqueue$ в переставленном порядке вызываются отсроченные действия (на сером фоне), которые определены идеальной программой $idealP$ в точке активации – «после приема сообщения m от псевдонима P ». Следует отметить, что программа $idealP$ взаимодействует с псевдонимами вместо идентификаторов сторон; устанавливается $rqueue := 0$;

пусть $T := T + 1$.

Другие активации. При приеме сообщения вида (m, P) от участника P:

положим, что $(\overline{P}, P) \in nyms$;

следует вызов немедленных действий, определенных идеальной программой $idealP$ в точке активации «после приема сообщения m от псевдонима $\overline{P}$ »;

сообщение ставится в очередь путем вызова $rqueue.add(m, \overline{P})$.

Перестановка (Permute): после получения $(\text{permute}, \text{perm})$ от злоумышленника S, записывается $perm$.

Получение времени (GetTime): при получении $gettime$ от участника P, уведомляется злоумышленник S о $(gettime, P)$ и возвращается значение текущего времени T участнику P.

Генерация псевдонима (GenNym): при получении $gennym$ от честной стороны P: уведомляется злоумышленник S о $gennym$. Ожидается, пока S не отреагирует новым псевдонимом $\overline{P}$ таким, что $\overline{P} \notin nyms$. Теперь пусть $nyms := nyms \cup \{(P, \overline{P})\}$ и отправить $\overline{P}$ стороне P. При получении $(gennym, \overline{P})$ от скомпрометированной стороны P: если $\overline{P} \notin nyms$, пусть $\overline{P} := nyms \cup \{(P, \overline{P})\}$.

Операции регистра (внутренняя активация):

Перевод (Transfer): при приеме $(transfer, amount, \overline{P}_r)$ от некоторого псевдонима $\overline{P}_s$:

довести $(transfer, amount, \overline{P}_r, \overline{P}_s)$ до идеального злоумышленника S;

следующие операции выполняются с отсроченной активацией:

положим, что $ledger[\overline{P}_s] \geq amount$;

$ledger[\overline{P}_s] := ledger[\overline{P}_s] - amount$;

$ledger[\overline{P}_r] := ledger[\overline{P}_r] + amount$,

где $\overline{P}_s$, $\overline{P}_r$ могут псевдонимами настоящих личностей. Следует отметить, что личность каждой из сторон является псевдонимом по умолчанию для стороны.

Замечание 2.

1. Данный шаблон проектирования предназначен для формального описания основных функций регистра при выполнении смарт-контракта. Отражает формальное описание функций времени, псевдонимности, открытого регистра и денежных переводов между сторонами.

2. Шаблон проектирования идеальной функциональности Ψ позволяет указать два типа точек активации для идеальной программы – точку моментальной активации и замедленной (отсроченной) активации. Точки активации вызываются на получателя сообщений. Моментальные активации обрабатываются немедленно, в то время как отсроченные активации собираются и обрабатываются партиями в следующем раунде.

3. Ψ позволяет идеальному злоумышленнику S указывать порядок $perm$, в котором сообщения должны быть обработаны в следующем раунде.

Шаблон проектирования протокола Π позволяет реализовать модульный подход в представлении пользовательских протоколов, представлен в табл. 3.

Таблица 3

Шаблон проектирования протокола $\Pi(prot)$

Относительно псевдонима:

Генерация псевдонима (GenNym): после получения входа $gennym$ из среды ε, генерируется $(epk, esk) \leftarrow Keygen_{enc}(1^{\lambda})$ и $(spk, ssk) \leftarrow Keygen_{sign}(1^{\lambda})$;

вызывается $payload := prot.GenNym(1^{\lambda}, (epk, spk))$;

хранится $nyms := nyms \cup \{(epk, spk, payload)\}$ и выход $(epk, spk, payload)$ в качестве нового псевдонима.

Отправка (Send): после получения внутреннего вызова $(send, m, \overline{P})$:

если $\overline{P} == P$ – отправить $(authentificated, m)$ $\Omega(C)$ (это аутентифицированная отправка);

в противном случае (псевдонимная отправка):

пусть псевдоним $\overline{P}$ записан в $nyms$;

запрашивается текущее время T от $\Omega(C)$;

вычисляется $\sigma' := Sign(ssk, (nonce, T, epk, m))$, где *ssk* – записанный секретный ключ подписи (secret signing key) соответствующий $\overline{P}$, *nonce* – сгенерированная случайная строка, а *epk* – записанный открытый ключ шифрования (public encryption key), соответствующий $\overline{P}$;

пусть $\sigma := (nonce, \sigma')$;

отправить $(pseudonymous, m, \overline{P}, \sigma)$ $\Omega(C)$.

Анонимная отправка (AnonSend): после получения внутреннего вызова $(anonsend, m, \overline{P})$ отправить $(anonymous, m)$ $\Omega(C)$.

Таймер и переводы регистра (ledger transfers):

Перевод (Transfer): после получения $(transfer, \$amount, \overline{P}_r, \overline{P})$ от среды ε:

пусть $\overline{P}$ – ранее сгенерированный псевдоним;

отправить $(transfer, \$amount, \overline{P}_r)$ $\Omega(C)$ как псевдоним $\overline{P}$.

Tick: получив «tick»-сообщение от среды ε, переслать сообщение $\Omega(C)$.

Другие активации:

Осуществление деятельности в качестве псевдонима: после получения любого входа вида $(m, \overline{P})$ от среды ε:

пусть $\overline{P}$ – ранее сгенерированный псевдоним;

передать $(m, \overline{P})$ локальной программе участника для обработки.

Другие: при приеме любого другого входа от ε или любого другого сообщения от участника – передать вход/сообщение локальной программе участника для обработки.

Замечание 3.

1. Шаблон проектирования протокола Π предназначен для формализации пользовательских протоколов и включает в себя основные этапы работы пользователя под псевдонимом.

2. Пользовательские протоколы взаимодействуют с программой смарт-контракта, описанном в шаблоне проектирования $\Omega(C)$.

3. Проектирование локальных пользовательских программ в соответствии с шаблонами проектирования функциональности $\Psi(\square)$ и $\Omega(\square)$ позволяет получить функциональные возможности с четко определенными формальными значениями. Программы также могут быть спроектированы с помощью шаблона проектирования протокола Π , чтобы получить полный протокол с формальными значениями.

3. Анализ безопасности системы смарт-контрактов с сохранением конфиденциальности

Система смарт-контрактов гарантирует следующие требования безопасности для сторон в смарт-контракте [4]:

- входная независимая конфиденциальность – ни один пользователь не видит ставки других пользователей, прежде чем совершит свои собственные. Ставки пользователей не зависят от других предложений, и система гарантирует входную независимую конфиденциальность даже в отношении нарушителя менеджера;

- последующая конфиденциальность – до тех пор, пока менеджер не раскрывает информацию, ставки пользователей являются скрытыми друг от друга (и от общественности) даже после выполнения смарт-контракта;

- финансовая справедливость – если сторона или менеджер аукциона досрочно прерывают выполнение контракта, то сторона прерывания должна быть финансово оштрафована, а остальные стороны получают пропорциональную компенсацию. В дополнение к этому данная модель позволяет определять дополнительные правила в рамках смарт-контракта, которые управляют финансовой справедливостью;

- защита от недобросовестного менеджера – модель гарантирует безопасность в отношении менеджера-злоумышленника. Помимо прерываний, нечестный менеджер не может повлиять на исход аукциона и на перераспределение денег, даже тогда, когда он сталкивается с множеством пользователей.

Представленная система смарт-контрактов с сохранением конфиденциальности позволяет фиксировать намерения злоумышленника. Моделирование намерений злоумышленника очень важно, так как оно отражает класс известных атак опережения, которые используют гибкость транзакций [7, 8].

Определение 4. Атака гибкости транзакций определяется тем, что злоумышленник создаёт ложный идентификатор для пользовательской транзакции.

Идентификатор транзакции формируется хэшированием информации в транзакции. Цифровая подпись пользователя является один из элементов хэш транзакции и подтверждает, что пользователь является отправителем транзакции. Злоумышленник изменяет подпись пользователя и все виды сделок, циркулирующие в сети, имеют два совершенно разных идентификатора транзакции. Так как конкретная сделка может быть подтверждена только один раз, то только один из идентификаторов транзакции будет добавлен в цепочку блоков, в то время как другой будет игнорироваться. Таким образом, пользователь лишается своих денежных средств.

Шаблоны проектирования основных функциональностей системы смарт-контрактов описывают намерения злоумышленника. Можно спроектировать идеальные функциональные возможности, которые исключают подобные атаки опережения. В частности, это могут быть усовершенствованные алгоритмы проверки цифровой подписи пользователя на этапе, предшествующем отправке хэшированной транзакции по сети.

Система смарт-контрактов основывается на технологии Blockchain. Копия регистра хранится на каждом компьютере сети и они синхронизируются, чтобы проверить, что имеют одну общую копию базы данных. Это обеспечивает исключительную степень отказоустойчивости. Если некоторое количество компьютеров прервет свою работу, общая база данных может быть воссоздана в полном объеме, так как одна и та же копия регистра хранится многими компьютерами.

Выводы

1. Представленная система смарт-контрактов с сохранением конфиденциальности на основе технологии Blockchain предлагает одновременно транзакционную конфиденциальность и программируемость в децентрализованной криптографической валютной среде.

2. Представленная система смарт-контрактов основывается на компиляторе Hawk и не хранит данные о финансовых операциях в открытом виде в регистре Blockchain. Компилятор автоматически генерирует эффективный криптографический протокол взаимодействия сторон контракта с Blockchain. Криптографический протокол взаимодействия сторон строится на неинтерактивных доказательствах с нулевым разглашением. Систему смарт-контрактов с сохранением конфиденциальности можно применить к различным вариантам взаимодействия пользователей с Blockchain.

3. Шаблоны проектирования функциональностей системы смарт-контрактов позволяют исключить атаки опережения за счет фиксации намерений злоумышленника. Например, это могут быть усовершенствованные алгоритмы проверки цифровой подписи пользователя на этапе, предшествующем отправке хэшированной транзакции по сети.

Список литературы: 1. *Szabo N*. The Idea of Smart Contracts / Nick Szabo. [Electronic resource]. – Режим доступа: http://szabo.best.net/smart_contracts_idea.html – 1997. 2. *Kumaresan R., Bentov I.* How to Use Bitcoin to Incentivize Correct Computations / R. Kumaresan, I. Bentov. – CCS. – 2014. –12 p. 3.*Canetti R.* Universally Composable Security: A New Paradigm for Cryptographic Protocols / R. Canetti. – FOCS. – 2013. – 87 p. 4. *Buterin V.* DAOs, DACs, DAs and More: An Incomplete Terminology Guide // ETHEREUM BLOG. [Electronic resource]. – Режим доступа: https://blog.ethereum.org/2014/05/06/daos-dacs-das-andmore-an-incomplete-terminology-guide/. – 2014. 5. *Blockchain.* BITCOIN FOUNDATION. [Electronic resource]. – Режим доступа: https://en.bitcoin.it/wiki/Block_chain. 4. Hawk: Privacy-Preserving Blockchain and Smart Contracts. [Electronic resource]. – Режим доступа: http://oblivm.com/hawk/index.html. 5. *Bogdanov D*. Sharemind: A Framework for Fast Privacy-Preserving Computations / D. Bogdanov, S. Laur, J. Willemson. – ESORICS. – 2008. – 15 p. 6.*Garay J. A.* The Bitcoin Backbone Protocol: Analysis and Applications / J. A. Garay, A. Kiayias, and N. Leonardos. – Eurocrypt. – 2016. – 37 p. 7.*Ben-Sasson E.* Zerocash: Decentralized anonymous payments from Bitcoin / E. Ben-Sasson, A. Chiesa, C. Garman, et al. // IEEE Symposium on Security and Privacy. – 2014. – 16 p. 8.*Decker C., Wattenhofer R.* Bitcoin Transaction Malleability and MtGox / Christian Decker, Roger Wattenhofer. – Computer Security ESORICS. – Springer. – 2014. – p. 313-326.

Харьковский национальный
университет радиоэлектроники

Поступила в редколлегию 11.04.2016