Белая книга PROVER

1. Основные положения

Сервис PROVER предназначен для осуществления объективной и независимой верификации видеоконтента. PROVER представляет собой онлайн сервис, позволяющий получать подтверждение наступления событий, зафиксированных на видео, подтверждать подлинность создания видеоматериалов с применением технологии блокчейн. PROVER предназначен как для автономного использования, так и для встраивания в сторонние платформы и сервисы для разработки приложений на базе его функционала.

2. Краткий обзор индустрии

Начавшаяся с появлением iPhone 10 лет назад революция смартфонов и планшетных компьютеров, оказала существенное влияние на способы коммуникации пользователей в сети. Каждый в одночасье стал творцом и распространителем своего контента, вызвав, как следствие, его экспоненциальный рост. Это существенно изменило и продолжает менять многие отрасли экономики, запуская процессы оцифровки окружающего нас мира. Фото и видео контент активно используется не только в развлекательно-познавательных целях, но и для других нужд, в том числе, экономического и юридического характера - финансовых, страховых, медицинских и других услугах. В этой связи имеется серьезная потребность в независимом децентрализованном сервисе, объективно гарантирующем подлинность создаваемого видеоконтента и защиту его от возможного подлога и недобросовестного редактирования.

Решение подобной задачи способно дать серьезный толчок в развитии цифровой экономики. Полученная технология может быть задействована в тысячах проектов из десятков различных областей и сможет помочь сотням миллионов пользователей по всему миру.

3. Проблемы отрасли

Подлинность цифровых видеозаписей событий и фактов, имеющих коммерческую и правовую ценность, нередко вызывает сомнения, поскольку видеофайлы могут быть отредактированы с целью подделки, сняты с использованием виртуальной видеокамеры (эмулятора), также в них могут быть искусственно изменены атрибуты, такие, как дата видеосъемки.

Задача сервиса PROVER заключается в том, чтобы гарантировать подлинность записи видеофайла с привязкой к конкретному времени осуществления видеосъемки.

4. Существующие решения

4.1. Онлайн решения по защите контента

Подлинность контента и права на него (копирайт) на сегодняшний день в основном пытаются гарантировать на уровне платформ для хранения и демонстрации видео материалов. Примером такого решения является ***Google content ID***, который позволяет только подтвердить время загрузки видео на YouTube, на этом строится предположение о его оригинальности, основываясь на принципе презумпции авторства (человек считается автором, пока настоящий автор не оспорит данный факт).

Недостатком этого и подобных решений является то, что они не позволяют установить время реальной записи видео, его оригинальность и целостность. Кроме того, эти решения работают в рамках своих платформ - видеосервисов "только на YouTube", предоставляются вручную администраторами сервиса по заявлению и всегда зависят от усмотрения администраторов сервиса. То есть всегда есть место человеческому фактору - субъективности или просто ошибки. А с надоедливыми искателями правды разговор короткий - "Владельцы контента, неоднократно подававшие необоснованные жалобы, могут лишиться права использования системы Content ID и потерять статус партнера YouTube".

Также контент пытаются защитить водяными знаками и логотипами на экране видео, но это может помочь только в последующем подтверждении своего авторства создателям контента, но не защитит от его подделки. То есть в юридических и финансовых делах это совершенно неприменимо.

4.2. Блокчейн решения по защите и подтверждении подлинности контента

На сегодняшний день уже существуют различные сервисы электронного нотариата на блокчейне, позволяющие заверять факт существования “Proof of existence” и авторство “Proof of ownership” различных файлов, документов и другого цифрового контента:

- Block Notary - это сервис, который помогает создавать доказательства существования любого контента (фото, файлы, любые носители) с использованием сети TestNet3 или Bitcoin. Фронтэнд системы представляет собой мобильное приложение для iOS, которое регистрирует хэш документа в блокчейне.
- Emercoin DPO Antifake - это технология, функционирующая на базе платформы Emer, позволяет создать для предмета (продукта) уникальный цифровой паспорт, хранящийся в блокчейне, и предоставляет сервисы для управления этим паспортом. Технология ориентирована в основном на оффлайн сегмент, помогает зарегистрировать в системе индивидуальные реквизиты (VIN, IMEI идентификаторы) и обеспечивает защиту реальных товаров от подделок.

- Stampery - это технология, которая может заверять электронную почту или любые файлы с помощью блокчейна. Технология упрощает процесс заверения писем путем простой пересылки их по электронной почте на специально созданный для каждого клиента почтовый адрес. Юридические фирмы используют технологию Stampery для очень экономичного способа верификации документов.
- https://www.ascribe.io/ - сервис регистрации авторства, последующего контроля и распространнения цифрового контента. Позиционируется для цифровых произведений искусства. Предлагает зарегистрировать произведение, выставить его на продажу в защищенный маркетплэйс, а затем следить за его использованием (демонстрацией).
- https://letsnotar.me - легкий сервис, автоматически сохраняющий в блокчейне хэш загружаемых в него файлов, запускаемый на смартфоне. Позволяет сразу обращаться к камере, делать фотографии и видеозаписи, хэшируя их. Однако он не способен гарантировать, что видеозапись ведется с реальной, а не виртуальной камеры, а следовательно, он не защищает от подлога данных.

Все эти сервисы объединяет одно - они могут заверить хэш попадающего в их распоряжение файла, уже записанного ранее или якобы записываемого с камеры устройства, однако же, они совершенно не могут гарантировать его оригинальность, целостность и подлинность, не позволяют защититься от подлога и недобросовестного редактирования, поскольку не обладают соответствующей технологией верификации создаваемого видеоконтента. Технология PROVER позволяет гарантировать, что данный контент создавался в конкретное время и конкретном месте, с камеры конкретного устройства, гарантировать отсутствие признаков подлога и редактирования.

PROVER может стать функциональным дополнением, расширяющим возможности рассмотренных выше сервисов и уровень доверия к ним. Имея возможность 100% гарантировать подлинность создаваемого видеофайла, данные сервисы смогут, например, предоставлять услуги по нотариальному заверению именно подлинности видео заявлений и обращений.

На сегодняшний день в мире нам не удалось обнаружить информацию о проекте занимающимся верификацией видео на этапе его создания. Это не удивительно, ведь для решения этой задачи необходимо обладать уникальным набором компетенций как в области блокчейн систем, так и в разработке модулей обработки видеопотока (видеоанализа). Наша сила в нашей команде, т.к. мы сочетаем в себе все необходимые уникальные компетенции и обладаем большим опытом в этих редко пересекающихся областях.

5. Что такое PROVER

Сервис PROVER состоит из нескольких компонентов:

- Мобильное приложение, которое устанавливается на смартфон и запускается вместе с включением видеокамеры, либо же само инициирует запуск видеокамеры.

- Набор алгоритмов и реализующих их утилит для интеграции технологии PROVER в сторонние решения и сервисы.
- Смартконтракт PROOF (только для реализации на базе платформы Ethereum)..

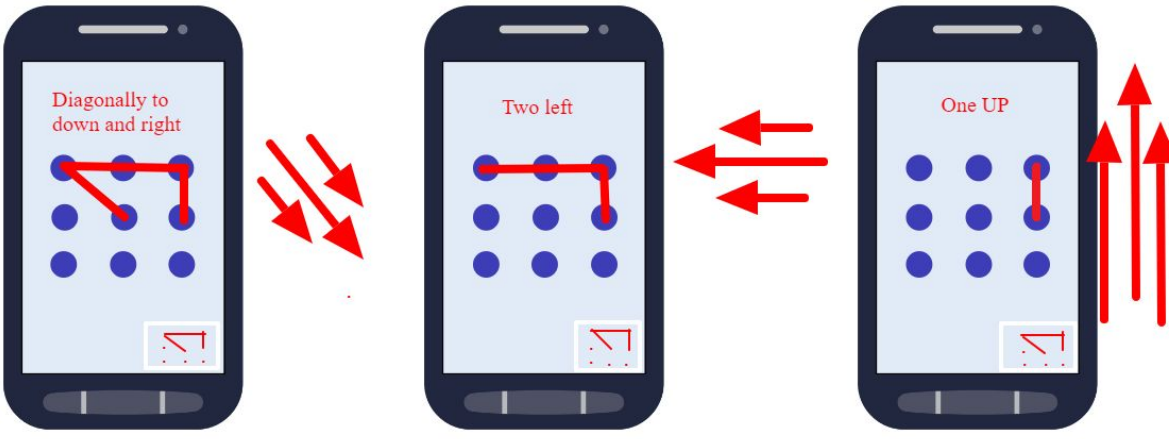
Основной научно-технической задачей, решение которой лежит в основе проекта, является верификация видеоматериалов, отснятых пользователем. В ходе верификации система подтверждает, что:

- съемка видеоматериала произведена реальной видеокамерой, встроенной в мобильное устройство, а не эмулировано виртуальной видеокамерой;
- видеоматериал является целостным, не редактированным, без склеек и вставок;
- съемка произведена в известный промежуток времени.

Все исходные тексты проекта будут размещены в общедоступном репозитории по адресу https://github.com/isvirin/prover.

5.1. Верификация с использованием swype кода

В основе верификации видео лежит алгоритм автоматического обнаружения swype кода в видеопотоке. В отличие от классического swype кода, который вводится движением пальца по сенсорному экрану, образуя непрерывную линию, соединяющую точки, изображенные на экране, в технологии PROVER ввод swype кода осуществляется перемещением смартфона с включенной камерой. Модуль видеоаналитики, который будет разработан на первом этапе, анализируя видеопоток с камеры смартфона, сможет обнаружить направление перемещения смартфона, а следовательно, построить виртуальную линию swype кода, которая должна в требуемой последовательности пересечь виртуальные окружности, расположенные на экране в виде трех строк по три окружности.



Иными словами, включив камеру смартфона, пользователь увидит на экране кнопку “Ввести swype код”, нажав на которую, пользователь увидит координатную сетку из 9 точек по 3 в ряд, swype код для ввода, сгенерированный в блокчейне, а также текущую точку, от которой начинается ввод swype кода. Для ввода swype кода пользователю необходимо перемещать в пространстве телефон, чтобы виртуальная линия swype кода прошла по требуемой траектории. После этого координатная сетка исчезает и появляется оповещение, что код введен.

Использование автоматического алгоритма распознавания swype кода позволит пользователю на этапе формирования видеозаписи быть уверенным в том, что впоследствии при подтверждении подлинности видеозаписи swype код будет распознан также.

5.2. Верификация с использованием датчиков

Для предотвращения подлога параллельно с видеопотоком будет осуществляться запись потока информации со всех датчиков, имеющихся в мобильном устройстве, с максимально возможной частотой - акселерометр, гироскоп, магнитометр, координаты GPS и др. Впоследствии будет разработан математический аппарат, который позволит по этой информации восстановить в мельчайших деталях движения мобильного устройства в руках пользователя и сопоставить эти движения с видеоконтентом.

Приведем описание математической модели определения положения в пространстве на основе трехосевого акселерометра и трехосевого магнитометра при условии перемещений с незначительными ускорениями.

Для определения положения тела в пространстве введем глобальную трехмерную декартову систему координат $OXYZ$, так чтобы ось OZ совпадала по направлению с направлением силовых линий гравитационного поля $\vec{g}$, а ось OY с совпадала со склонением вектора магнитного поля планеты $\vec{h}$.

Для описания положения датчика в глобальной системе координат (ГСК) введем локальную систему координат (ЛСК), оси которой будут совпадать с соответствующими осями датчиков ускорения и магнитного поля. Тогда положение ЛСК (датчиков) в ГСК можно описать четырьмя векторами:
$\vec{r_{\text{лск}}}$ – вектор смещения начала координат ЛСК относительно начала координат ГСК;
$\vec{i_{\text{лск}}}$, $\vec{j_{\text{лск}}}$, $\vec{k_{\text{лск}}}$ – направляющие векторы ортонормированного базиса ЛСК выраженные через направляющие векторы ортонормированного базиса ГСК.

Такое описание дает полную информацию об ориентации ЛСК в ГСК в координатной форме. Задача об определении угловой ориентации сводится к нахождению координат векторов $\vec{i_{\text{лск}}}$, $\vec{j_{\text{лск}}}$, $\vec{k_{\text{лск}}}$ в ГСК.

В связи с тем, что гравитационное поле более стабильно, чем магнитное, возьмем за основу датчик ускорений. Показания датчика ускорений представляют собой координаты вектора ускорения свободного падения, разложенные по осям ЛСК:
$\vec{a_{\text{лск}}} = \{a_x, a_y, a_z\}$

Нормализованный вектор $\vec{a_{\text{лск}}}$ есть ничто иное как вектор $\vec{k_{\text{гск}}}$, т. е. Задающий вектор оси OZ ГСК, выраженный через направляющие векторы ЛСК:
$\vec{k_{\text{гск}}} = \frac{\vec{a_{\text{лск}}}}{|\vec{a_{\text{лск}}}|} = \{k_x, k_y, k_z\}$

Показания датчика магнитного поля представляют собой координаты вектора магнитного поля, разложенные по осям ЛСК:
$\vec{m_{\text{лск}}} = \{m_x, m_y, m_z\}$

Вектор $\vec{m_{\text{лск}}}$ в общем случае может быть не параллелен вектору $\vec{k_{\text{гск}}}$, следовательно, нужно получить его нормальную компоненту:
$\vec{n_{\text{лск}}} = \vec{m_{\text{лск}}} - (\vec{m_{\text{лск}}} \cdot \vec{k_{\text{гск}}}) \cdot \vec{k_{\text{гск}}}$

Нормализованный вектор $\vec{n_{\text{лск}}}$ есть ничто иное как вектор $\vec{j_{\text{гск}}}$, т. е. Задающий вектор оси OY ГСК, выраженный через направляющие векторы ЛСК:
$\vec{j_{\text{гск}}} = \frac{\vec{n_{\text{лск}}}}{|\vec{n_{\text{лск}}}|} = \{j_x, j_y, j_z\}$

Задающий вектор оси OX ГСК, выраженный через направляющие векторы ЛСК найдем с использованием векторного произведения:
$\vec{i_{\text{гск}}} = \vec{j_{\text{гск}}} \times \vec{k_{\text{гск}}} = \{i_x, i_y, i_z\}$

Составим матрицу строки, которой представлены векторами $\vec{i_{\text{гск}}}$, $\vec{j_{\text{гск}}}$, $\vec{k_{\text{гск}}}$ затем транспонируем ее и разложим на векторы (также по строкам):
$|i_x\ i_y\ i_z\ j_x\ j_y\ j_z\ k_x\ k_y\ k_z\ |^T = |i_x\ j_x\ k_x\ i_y\ j_y\ k_y\ i_z\ j_z\ k_z\ |$

Таким образом, векторы:
$\vec{i_{\text{лск}}} = \{i_x, j_x, k_x\}$
$\vec{j_{\text{лск}}} = \{i_y, j_y, k_y\}$
$\vec{k_{\text{лск}}} = \{i_z, j_z, k_z\}$
определяют оси ЛСК в ГСК, иными словами, определяют ориентацию ЛСК в ГСК.

Выразим вектор $\vec{a_{\text{лск}}}$ в ГСК:
$\vec{a} = a_x \cdot \vec{i_{\text{лск}}} + a_y \cdot \vec{j_{\text{лск}}} + a_z \cdot \vec{k_{\text{лск}}} = \{a'_x, a'_y, a'_z\}$

Отсчеты вектора $\vec{a}$ заданы в квантах АЦП датчика ускорения, для дальнейших вычислений переведем их в систему СИ и запишем вектор мгновенных ускорений:
$\vec{a_{\text{физ}}} = \frac{range}{N} \cdot \vec{a}$,
где range – диапазон аналого-цифрового преобразователя датчика, а N – цена деления.

Для учета гравитационного поля следует уменьшить вертикальную компоненту на значение ускорения свободного падения:
$\vec{a_g} = \vec{a_{\text{физ}}} - \{0, 0, -g\}$

Перейдем от ускорения к скорости:
$\vec{v_g} = \int \vec{a_g} dt + \vec{v_0} \approx \sum \vec{a_g} \Delta T + \vec{v_0}$

Перейдем от скорости к координатам:
$\vec{r_g} = \int \vec{v_g} dt + \vec{r_0} \approx \sum \vec{r_g} \Delta T + \vec{r_0}$

Таким образом, предложенная математическая модель позволяет определить из показаний датчиков ориентацию датчика относительно глобальной системы координат, связанной с физическими особенностями планеты, определить путем двойного интегрирования линейное перемещение датчика в глобальной системе координат, а следовательно, по записанной последовательности измерений восстановить траекторию движения персонального мобильного устройства пользователя.

6. Как это работает

6.1. Общий алгоритм работы

Пользователь устанавливает мобильное приложение PROVER на свой смартфон, предоставляет права доступа к видеокамере, и впредь оно автоматически запускается вместе с видеокамерой устройства, работая в фоновом режиме.

При включении видеокамеры смартфон пользователя обращается через сеть Интернет к блокчейну для получения swype кода - проверочного задания, которое должен выполнить пользователь при осуществлении записи видеоданных. Ввод swype кода осуществляется путем движения смартфона с включенным режимом видеозаписи в пространстве виртуальных точек 3 на 3 точки по сгенерированной случайным образом (в блокчейне или на основе данных из блокчейна) траектории. При вводе swype кода пользователю отображаются подсказки в виде виртуальных точек 3 на 3 и распознанного движения смартфона. Для дополнительного подтверждения достоверности осуществляется синхронная с видео запись последовательности измерений с датчиков (акселерометр и магнитометр), встроенных в смартфон.

После завершения видеозаписи осуществляется вычисление хеша полученного видеофайла. Полученный хэш сохраняется в блокчейне.

Записанные видеоданные хранятся у пользователя на персональном устройстве или облачном диске и могут быть предъявлены на проверку подлинности при необходимости.

Таким образом, в блокчейне в том или ином виде оказывается доступной следующая информация:

- дата и время получения индивидуального swype кода;
- сгенерированный swype код;
- хэш файла видеоданных (сам файл видеоданных хранится у пользователя);
- дата и время загрузки хэша.

Для проверки подлинности конкретного видеофайла пользователя осуществляют вычисление хэша файла, для которого затем из блокчейна извлекают сохраненную информация или при ее отсутствии уведомляют о том, что такой хэш не был загружен. На основе полученной информации делают вывод о том, что файл с данным хешем

был создан не ранее времени получения индивидуального swype кода и не позднее времени загрузки хеша в блокчейн.

В ходе проверки подлинности осуществляется визуальный просмотр видеофайла на предмет непрерывности для исключения видеомонтажа и наличия на видеозаписи swype кода, выданного пользователю в качестве проверочного задания. После выполнения первого этапа проекта PROVER наличие swype кода на видеозаписи будет определяться автоматическим алгоритмом. После выполнения второго этапа проекта PROVER непрерывность видеозаписи будет определяться автоматическим алгоритмом. При наличии непрерывной видеозаписи, содержащей совпадающий со сформированным из блокчейна swype кодом, делается обоснованный вывод о том, что запись была сделана пользователем не ранее момента выдачи swype кода. При этом момент выдачи swype кода зафиксирован в блокчейне и недоступен для модификации пользователем.

После выполнения третьего этапа проекта PROVER будет доступен алгоритм восстановления траектории движения персонального мобильного устройства в руках пользователя в ходе осуществления видеозаписи и сравнение восстановленной траектории с записанными видеоданными. При совпадении видеозаписи и восстановленной траектории движения персонального мобильного устройства делается вывод об однозначном подтверждении того, что запись сделана с реальной камеры, встроенной в персональное мобильное устройство, а не эмулируемой.

6.2. Реализация для Ethereum

Реализация для платформы Ethereum осуществляется на базе смартконтракта PROOF. Мобильное приложение пользователя обращается через сеть Интернет к смартконтракту PROOF для получения swype кода. Смартконтракт PROOF фиксирует выданный одноразовый swype код и время его выдачи.

После завершения видеозаписи осуществляется вычисление хеша полученного видеофайла. Полученный хэш отправляется в смартконтракт PROOF, где происходит его сохранение.

Смарконтракт PROOF сохраняет следующую информацию:

- дата и время получения индивидуального swype кода;
- сгенерированный swype код;
- хэш файла видеоданных (сам файл видеоданных хранится у пользователя);
- дата и время загрузки хэша.

Для проверки подлинности конкретного видеофайла пользователя осуществляют вычисление его хэша, который отправляется смартконтракту PROOF. В ответ смартконтракт PROOF возвращает сохраненную для этого хеша информацию или уведомляет пользователя, что такой хэш не был загружен. На основе полученной информации можно сделать вывод о том, что файл с данным хешем был создан не ранее времени получения индивидуального swype кода и не позднее времени загрузки хеша в смартконтракт.

6.3. Реализация для Emercoin

Реализация для платформы Emercoin предполагает, что формирование swype кода осуществляется на основе хэша последнего (или любого другого) блока в цепочке, адреса клиента блокчейн сети и, возможно, другой дополнительной информации (например, IMEI идентификатора мобильного устройства пользователя). Псевдокод формирования swype кода может выглядеть следующим образом:

```
bytes32 mixedString = mix(userAddress, IMEI, blockHash);
bytes32 temp = sha3(mixedString);
swype = uint16(uint256(temp) % 65536);
```

После завершения видеозаписи осуществляется вычисление хеша полученного видеофайла. Полученный хэш сохраняется в блокчейне Emercoin вместе с указанием номера блока, на основе которого сформирован swype, а также IMEI мобильного устройства, которым снято видео, при этом транзакция должна быть выполнена с того же адреса, который был использован при вычислении swype кода.

Для проверки подлинности конкретного видеофайла пользователя осуществляют вычисление его хэша, по которому из блокчейна Emercoin извлекают сохраненную информацию. На основе полученной информации можно сделать вывод о том, что файл с данным хешем был создан не ранее времени добавления в блокчейн блока, на основе хэша которого сгенерирован swype код, и не позднее времени загрузки хеша видеофайла в блокчейн.

7. Где это может быть применено

Технология PROVER может использоваться самостоятельно, но особую ценность она приобретает из-за того, что может быть положена в основу большого количества прикладных решений, приложений и сервисов из самых разных областей.

7.1. Автострахование (КАСКО)

Прикладное решение “Доступное КАСКО” предназначено для страхования автомобилей, например, для экспресс страхования на определенный короткий срок (сутки, одна поездка и т.п.). Клиент осуществляет видеозапись состояния своего автомобиля для заключения договора страхования с целью предоставления заверенной системой PROVER, видеозаписи в качестве доказательства в случае наступления страхового случая. При наступлении страхового случая пользователь предъявляет в страховую компанию записанные видеоданные для проверки их подлинности.

Решение состоит из следующих компонентов:

- ***Сервер платформы (провайдера услуг)*** представляет собой сервис, который принимает запросы от Мобильных приложений клиентов, транслирует их на Сервер страховой компании, а также размещает информацию в блокчейне.

- ***Сервер страховой компании*** представляет собой сервис, осуществляющий непосредственно тарификацию страховых услуг, оказываемых пользователю. По сути, Сервер страховой компании считает, сколько денег осталось у клиента. Для верификации данных Сервер страховой компании может обратиться к Серверу платформы, а может - непосредственно к блокчейну.

- ***Мобильное приложение управления статусом страховки клиента*** позволяет включить и выключить страховку, указав при этом объем оказываемых услуг. Мобильное приложение взаимодействует только с Сервером платформы, который гарантирует клиенту размещение данных в блокчейне. Кодом доступа к услуге может являться номер страхового полиса, который покупается клиентом непосредственно в Страховой компании. При активации страховки пользователь должен записать на видео транспортное средство со всех сторон. Видеофайл пользователя хранится у него (на смартфоне, планшете или на личном сетевом диске типа Google Drive или Dropbox), а на Сервер платформы отправляется хеш, который размещается в блокчейне. При наступлении страхового случая пользователь направляет в Страховую компанию видеофайл, а Страховая компания проверяет его подлинность по хешу и времени записи через блокчейн.

Этот подход позволяет исключить мошенничество со страховкой, когда клиент в сговоре с сотрудником страховой компании или сотрудником правоохранительных органов страхует заранее поврежденный автомобиль, либо заявляет инсценированный страховой случай. В мировом масштабе это позволит страховым компаниям сэкономить миллиарды долларов.

Существенным для рассматриваемого применения является однозначное или с высокой степенью достоверности подтверждение того, что предоставленные видеоданные были записаны клиентом не ранее или не позднее известного зафиксированного промежутка времени, а запись произведена пользователем реальной камерой, встроенной в персональное мобильное устройство (смартфон, планшет, персональный компьютер, ноутбук), чтобы предотвратить возможность выдачи видеоданных, записанных пользователем заранее до наступления страхового случая.

Решение “Доступное КАСКО” будет представлять собой серверную платформу и мобильное приложение, позволяющие включать и выключать страховку в течение периода страхования, указывая при этом объем запрашиваемых услуг (опций страхования). Это позволит, например, не страховать автомобиль от угона, когда водитель находится за рулем, или вообще отключить страховку КАСКО, когда автомобиль поставлен на парковку в гараж. При этом установка дополнительных устройств в автомобиль не требуется.

Конкурентными преимуществами создаваемого продукта “Доступное КАСКО” являются:

- возможность снизить цену на КАСКО за счет гибкого управления страхованием в течение всего периода действия полиса, тем самым привлечь

автовладельцев разного уровня достатка, в т.ч. тех, которые не пользовались данным продуктом страхования вообще из-за его высокой стоимости;
- отсутствие необходимости установки на автомобиль дополнительных устройств;
- возможность владельцу страховки легко и быстро доказать наступление страхового случая путем отправки в Страховую компанию видеоматериалов, при этом подтверждение подлинности представленных видеоматериалов осуществляется с использованием технологии PROVER;
- возможность обеспечения доверенного взаимодействия между клиентом и страховой компанией, используя распределенный реестр на основе блокчейн, хранящий все транзакции управления опциями страхования, а следовательно, гарантом сохранности выполненных клиентом действий является не страховая компания, а миллионы пользователей блокчейн сети по всему миру!

7.2. Подтверждение авторства оригинального видео

Пользователь, ведя запись видео (с помощью камеры мобильного телефона или планшета), может еще и зарегистрировать при этом свое авторство и иметь возможность его подтвердить. Это может быть интересно как мобильным репортерам, внештатным корреспондентам (стрингерам), блогерам, так и спортсменам-экстремалам, путешественникам, музыкантам, композиторам и другим людям, занимающимся творчеством, а так же просто обычным пользователям социальных сетей и видео сервисов.

Монетизация предполагается на основе фримиум модели. Приложение будет предоставлять возможность зарегистрировать несколько видео бесплатно, далее сервис будет предлагать приобрести подписку на определенный срок или же предлагать приобретать пакеты для регистрации 50, 100, 500 видеофайлов.

7.3. Удаленный контроль пациентов

Технология PROVER может лечь в основу системы удаленного контроля пациентов, в частности, в области подтверждения приема лекарств. Для этого, пациент устанавливает приложение клиники или фармацевтической компании, куда встроена технология PROVER. У него есть "аптечка" и график приема лекарств. В нужное время на телефоне всплывает уведомление, и он, чтобы выпить таблетку, включает приложение и под запись достает таблетку из упаковки и пьет, заверяя подлинность с помощью swype кода.

Для клиники и страховой компании это может служить подтверждением, что больной получил должную терапию в полном объеме.

Для пациента - доказательством, что он надлежащим образом выполнил все предписания врачей, и лечение будет оплачено страховой компанией.

Для фармацевтической компании это может быть гарантией того, что весь курс их препарата будет пройден пациентом в полном объеме, что позволит существенно повысить объем продаж лекарственных средств, принося дополнительную прибыль.

7.4. Онлайн игры и квесты

В мобильное приложение игры встраивается технология PROVER, которая позволяет регистрировать наступление игровых событий, например обнаружение закладки тайников и кладов с призами, достижение какого-то уровня, географической точки и т.п.

7.5. Отчеты о проделанной работе

С технологией PROVER будет легко осуществима удаленная проверка производства работ (строительство, монтаж, уборка, патрулирование, курьерская доставка, выкладка товара в магазине, размещение наружней рекламы и т.п.). Видео беспристрастно фиксирует процесс или результат процесса - факт, а сервис PROVER объективно заверяет дату и время записи видео.

7.6. Фиксация нарушений ПДД и общественного порядка

Технология PROVER может применяться в области охраны общественного порядка. С ее помощью можно беспристрастно зафиксировать время, дату и место видео свидетельства для предоставления правоохранительным органам и обеспечить объективное доказательство подлинности. Данные могут быть использованы в качестве доказательства в ходе разбирательства, может быть заявлена техническая экспертиза, и эксперт сможет восстановить информацию о синхронизации и геопозиции и подтвердить ее достоверность.

7.7. Образовательные проекты

Для образовательных проектов технология PROVER может выполнить функцию по удаленной идентификации пользователей и верификации их действий внутри образовательной платформы. На сегодняшний день в мире набрали популярность Массовые Открытые Образовательные Курсы (MOOC), например Coursera, Udemy, Udacity, edX и другие. Уязвимым звеном данных сервисов является невозможность 100% подтверждения личности обучающегося и верификации полученных знаний. В настоящий момент на платформе Coursera после сдачи тестов требует сделать селфи, но этот подход совершенно не защищен от подлога. Можно интегрировать технологию PROVER в приложение для дистанционного обучения и вести видеозапись сдачи экзамена пользователем, чтобы убедиться, что экзамен сдает именно он - человек на видео, и что он не пользовался шпаргалками, не выходил из комнаты и т.п. Этот подход позволит выдавать именные сертификаты с фото и быть уверенными в том, что человек изображенный на фото на сертификате, является именно тем человеком, что сдавал экзамены. Протоколы экзаменов можно будет также хранить для подтверждения.

7.8. Защищенные видео заявления - подтвержденные видеосообщения юридического характера

Технология PROVER дает возможность защитить и подтвердить видеосообщения юридического характера без очного визита к каким-либо должностным лицам. Удаленное видео заявление о совершении какой-либо сделки, дача показаний, объяснение, отчет, интервью и т.п. станут объективными и легитимными

свидетельствами. Система подтверждает дату и место съемки, должностное лицо сличает видео с имеющимся в базе фото или видео и подтверждает, что это тот же самый человек. Например так могут работать удаленные приемные государственных, муниципальных и корпоративных должностных лиц, рассматривая записанные обращения граждан/клиентов.

8. Команда проекта

Команда проекта имеет более чем десятилетнюю историю совместной работы. Она реализовала целый ряд масштабных проектов в области IT и техники для здоровья, результатами которых стали интеллектуальные системы видеонаблюдения, продукты и сервисы для здоровья, известные в 64 странах мира. На данный момент команда проекта исповедует принципы открытости и децентрализации и вкладывает свои силы в развитие передовых технологий, связанных с блокчейн. Мы умеем делать качественные и востребованные продукты и верим, что можем принести пользу всему блокчейн сообществу, внеся проектом PROVER свой вклад в криптоэкономику.

8.1. Основной состав

Набильская Надежда, CEO, сооснователь

- В сфере информационной безопасности и технических средств охраны работает с 2010 года.
- Выпускница Президентской программы подготовки управленческих кадров.
- Руководство проектами: разработка программного обеспечения - 3, научно-исследовательские и опытно-конструкторские работы - 5, прикладные научные исследования - 2.

Рытиков Алексей, CTO

- Опыт разработки программного обеспечения в сфере технических средств охраны 10 лет.
- Участвовал в ряде опытно-конструкторских работ в качестве ключевого участника. Имеет опыт разработки сложных технических систем в области видеонаблюдения.

Воронин Вячеслав, ученый в области видеоанализа

- Доцент Донского государственного технического университета.
- Кандидат технических наук.
- Соавтор монографии “Метод и алгоритмы выделения полезного сигнала на фоне шумов при обработке дискретных сигналов”.
- Рецензент Международного журнала IEEE Transactions on Image Processing, Международной конференции International Symposium on Image and Signal Processing and Analysis (ISPA), Международной конференции International Symposium on Circuits and Systems (ISCAS).
- Лауреат именной премии главы администрации (губернатора) Ростовской области.
- Лауреат премии «Ползуновские гранты».

Супрун Виталий, разработчик мобильных приложений

- Опыт разработки мобильного программного обеспечения более 10 лет.
- Разработчик мобильного приложения ECG Dongle.

Юферова Елена, бизнес-консультант

- Опыт работы в консалтинговых компаниях в качестве руководителя направления кадрового консалтинга и компаниях реального сектора в качестве эксперта в области управления организацией и человеческими ресурсами более 25 лет.
- Соавтор справочника по управлению персоналом. Соавтор практического пособия для менеджеров «Лицом к лицу с будущим сотрудником» М, 2001 г.

8.2. Советники

Евгений Шумилов

- Евангелист блокчейн технологий.
- Основатель и генеральный директор Emercoin.

Олег Ховайко

- Эксперт по криптографии и финансам.
- Технический директор Emercoin.

Свирин Илья, сооснователь

- Кандидат технических наук.
- Технологический предприниматель.
- Основатель группы компаний “Нордавинд”.
- Разработчик технологий в области цифровых систем видеонаблюдения, персональной техники и сервисов для здоровья (в т.ч. всемирно известные Кардиофлешка “ECG Dongle” и сервис “КардиоОблако”).
- Автор многочисленных научных публикаций по вопросам информационной безопасности и теоретическим основам программирования.

9. Дорожная карта проекта

10. Руководство покупателя. Инвестирование

Чтобы финансировать разработку и функционирование системы PROVER, будет проводиться фаза сбора средств, известная как crowdsale. Проведение crowdsale будет осуществляться в рамках экосистемы Ethereum. Во время crowdsale люди могут приобретать по фиксированной ставке токены PROOF, которые:

- являются платежным средством при оплате услуг сервиса PROVER. На технологическом уровне прикладные сервисы (например, сервис автострахования) расплачиваются с PROVER токенами PROOF, а взаиморасчеты со своими пользователями ведут в любой валюте, например, в своих токенах или фиатных деньгах;
- предоставляют право на размещение запроса на финансирование проекта. Проекты могут быть направлены на развитие непосредственно инфраструктуры PROVER (разработка новых алгоритмов, создание реализаций под другие платформы), маркетинг и продвижение проекта PROVER, создание прикладных решений, направленных на развитие экосистемы PROVER (реализация прикладных сервисов, использующих PROVER в качестве инструмента подтверждения подлинности видеоматериалов). Объем запроса не может превышать количество имеющихся у заявителя токенов PROOF, умноженное на 1000, что позволяет учитывать степень вовлеченности заявителя в комьюнити проекта PROVER, а следовательно, его личную заинтересованность в развитии проекта;
- предоставляют право для участия в голосовании по заявленным на финансирование проектам. Сила голоса участника пропорциональна количеству имеющихся у него токенов PROOF.

Для обеспечения замкнутости экономической системы PROVER собранные смартконтрактом PROOF токены PROOF должны возвращаться в оборот:

- 10% собранных токенов PROOF, но не менее 1 токена за каждое подтвержденное видео, передаются команде проекта для поддержания функционирования децентрализованной системы;

- 90% собранных токенов PROOF остаются под управлением смартконтракта PROOF и распределяются по итогам голосования держателей токенов PROOF под проекты, инициированные как командой проекта, так и внешними исполнителями.

10.1. Pre-ICO

Смартконтракт PROOF осуществляет эмиссию токенов PROOF, количество которых в течение Pre-ICO ограничено максимальной собираемой суммой 300’000$, при достижении которой выпуск токенов прекращается. Токены продаются цене, фиксированной в долларах США, 125 PROOF = 1$, т.е. инвестор на этапе Pre-ICO имеет бонус 25%, по сравнению с последующим crowdsale. Покупка осуществляется путем перечисления эфира на адрес смартконтракта, а владельцем приобретенных токенов становится отправитель транзакции. **Будьте внимательны и помните, что не следует осуществлять оплату из несовместимых с контрактами ERC20 кошельков или со счета на криптовалютной бирже - это может привести к потере контроля над приобретенными Вами токенами.** Курс эфира к доллару США фиксируется в момент запуска Pre-ICO и остается неизменным в течение всего времени его проведения. Продолжительность проведения Pre-ICO составляет 14 дней с момента запуска. В течение Pre-ICO должны быть проданы все выпущенные токены, в противном случае все собранные средства возвращаются инвесторам, за вычетом комиссий за транзакции и стоимости газа.

Все собранные на Pre-ICO средства перечисляются команде и должны быть потрачены на проведение следующих работ:

- разработка прототипа мобильного приложения под Android, обеспечивающего взаимодействие со смартконтрактом PROOF;
- маркетинг и продвижение проекта, подготовка к проведению crowdsale;
- перевод white paper на китайский, немецкий, французский, итальянский и украинский языки.

10.2. Crowdsale

Смартконтракт PROOF осуществляет эмиссию токенов PROOF, количество которых в течение crowdsale ограничено максимальной собираемой суммой 5’200’000$, при достижении которой выпуск токенов прекращается. Токены продаются цене, фиксированной в долларах США, 100 PROOF = 1$. Покупка осуществляется путем перечисления эфира на адрес смартконтракта, а владельцем приобретенных токенов становится отправитель транзакции. **Будьте внимательны и помните, что не следует осуществлять оплату из несовместимых с контрактами ERC20 кошельков или со счета на криптовалютной бирже - это может привести к потере контроля над приобретенными Вами токенами.** Курс эфира к доллару США фиксируется в момент запуска crowdsale и остается неизменным в течение всего времени его проведения. После окончания crowdsale осуществляется единовременная дополнительная эмиссия, в ходе которой выпускается 28% общей эмиссии токенов, 19% из которых остаются за командой проекта, 1% предназначены для советников проекта, а 8% передается в управление первоначальным инвесторам. Продолжительность проведения crowdsale составляет 30 дней с момента запуска.

Для ранних инвесторов предусмотрена система скидок:

- 115 PROOF = 1$ при покупке в первый день crowdsale;
- 110 PROOF = 1$ при покупке в течение первой недели проведения crowdsale.

Средства, полученные в ходе crowdsale, предназначены для создания инфраструктуры PROVER и обеспечения ее функционирования. Выпуск токенов после этого не предусмотрен. Условием успешности crowdsale является сбор минимум 3’600’000$, не считая средств, собранных в ходе Pre-ICO. В противном случае все собранные средства возвращаются инвесторам, за вычетом комиссий за транзакции и стоимости газа.

Количество выполняемых этапов проекта зависит от количества собранных в ходе crowdsale средств. В результате каждого этапа создается функционально законченный продукт, а выполнение каждого следующего этапа добавляет ему дополнительные потребительские свойства.

10.3. Расходование инвестиционного капитала

Сразу после успешного crowdsale средства в объеме 1’500’000$ перечисляются смартконтрактом PROOF команде для выполнения первого этапа проекта. Остальные средства остаются под управлением смартконтракта PROOF.

По завершении каждого этапа проекта команда размещает результаты работы в открытом доступе на сайте проекта http://prover.io и на https://github.com/isvirin/prover, а также публикует план работ на следующий этап и требуемое финансирование. Все это выносится на голосование, которое длится 7 дней с момента его инициирования командой разработчиков.

Владельцы токенов PROOF осуществляют голосование, при этом сила голоса конкретного владельца пропорциональна количеству имеющихся у него токенов PROOF в общей эмиссии. Если положительно голосует простое большинство от общего числа проголосовавших токенов, работа по выполненному этапу считается принятой, а план работ на следующий заявленный этап утвержденным. При этом запрошенная сумма перечисляется с баланса смарт-контракта PROOF команде проекта PROVER.

10.4. Синергия токенов PROOF и HMQ

Проекты Humaniq и PROVER расширяют экосистемы друг друга. Так для PROVER проект Humaniq является прикладным решением. Взаиморасчеты между Humaniq в сторону PROVER осуществляются в токенах PROOF, при этом со своими клиентами Humaniq работает в собственных токенах HMQ.

Таким образом, рост стоимости токенов PROOF обеспечивается за счет того, что проект Humaniq расширяет экосистему PROVER, создавая дополнительную востребованность токенов PROOF на рынке при их единовременно фиксированной эмиссии.

Рост стоимости токенов HMQ обеспечивается за счет того, что проект PROVER расширяет функциональность системы Humaniq, обеспечивая дополнительную ценность Humaniq для клиентов, которая дополнительно предоставляется в рамках неизменной эмиссии токенов HMQ.

11. Руководство пользователя. API смартконтракта PROOF

Смартконтракт PROOF разработан с использованием экосистемы Ethereum и осуществляет эмиссию токенов PROOF с использованием стандарта Token ERC20. Спецификация ERC20 расширена для обеспечения возможности голосования и реализации общей бизнес-логики проекта PROVER.

11.1. Голосование за результаты работы команды проекта

Для владельцев токенов PROOF предусмотрена возможность голосования за прием выполненных командой проекта работ по завершенному этапу, а также утверждение плана дальнейших работ и выделение финансирования на очередной этап. Голосование запускается командой проекта путем вызова функции ***startVoting(uint weiReqFund)*** смартконтракта PROOF и длится 7 дней. В качестве параметра функция получает объем требуемого финансирования для выполнения следующего этапа проекта. При начале голосования испускается событие ***VotingStarted(uint weiReqFund)***, которое содержит информацию о запрошенном на следующий этап финансировании.

Для получения информации о текущем голосовании может быть вызвана константная функция ***votingInfo()***, которая возвращает информацию о запрошенном на следующий этап финансировании и времени до окончания голосования. Если голосование не проводится или уже завершено, то функция возвращает нулевые значения.

Функция голосования ***vote(bool inSupport)*** доступна только держателям токенов PROOF и может быть вызвана каждым держателем только один раз в ходе одного голосования. Подсчет голосов осуществляется только в момент завершения голосования, чтобы исключить многократный учет одних и тех же токенов при голосовании, если в ходе голосования была осуществлена их передача. При каждом успешном вызове функции испускается событие ***Voted(address indexed voter, bool inSupport)***, которое содержит адрес проголосовавшего держателя токенов PROOF и его голос.

Голосование завершается командой проекта путем вызова функции ***finishVoting()*** по истечении 7 дней голосования. Функция осуществляет подсчет голосов, учитывая фактическое количество токенов, которыми владеют все держатели, вызвавшие функцию ***vote(bool inSupport)*** в ходе голосования. Вес голоса каждого держателя токенов PROOF определяется количеством имеющихся у него токенов PROOF. Голосование осуществляется простым большинством. При завершении голосования испускается событие ***VotingFinished(bool inSupport)***, которое содержит информацию об исходе голосования. При положительном исходе голосования запрошенные средства в эфирах перечисляются со счета смартконтракта PROOF на адрес команды проекта.

11.2. Размещение доказательства

Размещение доказательства в смартконтракте PROOF осуществляется в два этапа. На первом этапе пользователь вызывает функцию ***swypeCode()***, которая случайным образом генерирует swype код и возвращает его пользователю. Кроме того, функция осуществляет запоминание сгенерированного swype кода, а также время его выдачи.

После того, как пользователь осуществил запись видеоматериала (с учетом выданного swype кода), он вызывает функцию ***setHash(uint16 swype, bytes32 hash)***, которая в качестве параметров получает выданный ранее swype код и хэш записанного видеофайла. Функция осуществляет запоминание переданного хеша в ассоциативном массиве, где для каждого элемента сохраняются swype код, время выдачи swype кода, время загрузки хеша и адрес владельца видеоматериалов. Именно вызов данной функции требует платы за осуществление услуги сервиса в соответствии с тарификацией, зависящей от количества токенов PROOF у пользователя.

11.3. Проверка доказательства

Для проверки доказательства пользователь может обратиться к публичному контейнеру ***mapping (bytes32 => Video) public videos***, из которого по хешу видеофайла могут быть получены swype код, время выдачи swype кода, время загрузки хеша и адрес владельца видеоматериалов. В случае нахождения в контейнере хеша видеофайла, можно утверждать, что, если в видеопотоке присутствует сохраненный в смартконтракте PROOF swype-код, данный видефайл был записан не ранее зафикисрованного времени выдачи swype-кода и не позднее времени загрузки хеша.

Проверка наличия в видеопотоке swype-кода осуществляется внешними алгоритмами, которые будут разработаны на первом этапе выполнения проекта. Реализация данных алгоритмов будет доступна в виде исходных кодов на https://github.com/isvirin/prover.

11.4. Миграция токенов

В случае выявления критических ошибок в бизнес-логике смартконтракта PROOF или разработке новых технических решений, существенно улучшающих бизнес-логику, команда проекта PROVER оставляет за собой право замены смартконтракта на новый. При этом, чтобы соблюсти интересы держателей токенов предоставляется механизм безопасной миграции токенов.

11.5. Финансирование проектов, инициированных комьюнити

Любой держатель токенов PROOF, заинтересованный в развитии или продвижении проекта PROVER, имеет право на размещение запроса на финансирование. Для размещения запроса держатель токенов PROOF вызывает функцию ***deployProject(uint proofReqFund, string urlInfo)***, которая получает в качестве параметров запрашиваемый объем финансирования в токенах PROOF, а также URL, по которому размещено публичное описание проекта. Следует учитывать, что объем запроса не может превышать количество имеющихся у заявителя токенов PROOF,

умноженное на 1000. Если держателем токенов PROOF уже размещен запрос на финансирование и он находится на голосовании, то размещение еще одного запроса не допускается. Новый запрос может быть размещен сразу после завершения голосования по уже размещенному запросу, независимо от результатов голосования.

Если размещение запроса произошло успешно, то испускается событие ***Deployed(address projectOwner, uint proofReqFund, string urlInfo)*** и начинается процедура голосования, которая длится 7 дней.

Для получения информации о размещенном запросе на голосование может быть вызвана константная функция ***projectInfo(address projectOwner)***, которая возвращает информацию о запрошенном финансировании, URL, по которому размещено публичное описание проекта и время до окончания голосования. Если голосование не проводится или уже завершено, то функция возвращает нулевые значения.

Функция голосования ***vote(address projectOwner, bool inSupport)*** доступна только держателям токенов PROOF и может быть вызвана каждым держателем только один раз в ходе голосования по каждому запросу. Подсчет голосов осуществляется только в момент завершения голосования, чтобы исключить многократный учет одних и тех же токенов при голосовании, если в ходе голосования была осуществлена их передача. При каждом успешном вызове функции испускается событие ***Voted(address projectOwner, address voter, bool inSupport)***.

Голосование завершается путем вызова функции ***finishVoting(address projectOwner)*** по истечении 7 дней голосования. Функция осуществляет подсчет голосов, учитывая фактическое количество токенов, которыми владеют все держатели, вызвавшие функцию ***vote(address projectOwner, bool inSupport)*** в ходе голосования. Вес голоса каждого держателя токенов PROOF определяется количеством имеющихся у него токенов PROOF. Голосование осуществляется простым большинством. При завершении голосования испускается событие ***VotingFinished(address projectOwner, bool inSupport)***, которое содержит информацию об исходе голосования. При положительном исходе голосования запрошенные средства в токенах PROOF перечисляются с баланса смартконтракта PROOF на баланс заявителя.

12. Заключение

Первые телефоны с видеокамерами появились примерно в 2002-м году, а полноценная эпоха смартфонов с появлением iPhone пришла только 10 лет назад. До 2005 года в мире не существовало сервиса YouTube! Первая версия кода биткойна возникла в 2008, а первый кошелек Сатоши Накамото создал в начале 2009 года. Можете ли вы сегодня представить свою жизнь без них?

Количество пользователей смартфонов в мире составляет около 4 млрд человек, а к 2020 году их будет более 6 млрд. И все эти люди будут общаться в мессенджерах, постить контент, в том числе и видео контент в различные социальные сети.

Все они неизбежно будут вовлечены в новую экономику, будут приобретать товары и услуги онлайн, с помощью своих устройств, в том числе и с помощью крипто валют.

Большинство существующих сегодня оффлайн услуг будут так или иначе вынесены в онлайн и масштабированы на миллиарды пользователей по всему миру. Существенно тормозит этот процесс сейчас лишь страх перед мошенниками и требования к безопасности.

Наша технология способна поставить надежный заслон на пути мошенников и широко открыть двери в онлайн и криптоэкономику для целых отраслей банковской и страховой, юридической и прочих консервативных сфер.

Нельзя также не отметить и глобальную пользу нашего проекта для развития блокчейн комьюнити. Наша система может являться драйвером популяризации технологии блокчейн и криптовалют среди широких слоев населения нашей планеты, приведя в блокчейн миллионы новых пользователей со всего мира, увеличив общий объем и востребованность блокчейн экономики в целом!

13. Литература

1. https://en.wikipedia.org/wiki/Swype
2. https://en.wikipedia.org/wiki/Insurance_fraud
3. By the numbers: insurance fraud statistics
4. Google content ID
5. Ethereum homepage
6. Ethereum Request for Comments (ERC) 20
7. Solidity homepage
8. How To Learn Solidity: The Ultimate Ethereum Coding Guide
9. BlockChain Technology Beyond Bitcoin

14. Смартконтракт PROOF

```
/*
This file is part of the PROOF Contract.

The PROOF Contract is free software: you can redistribute it and/or
modify it under the terms of the GNU lesser General Public License as published
by the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

The PROOF Contract is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU lesser General Public License for more details.

You should have received a copy of the GNU lesser General Public License
along with the PROOF Contract. If not, see <http://www.gnu.org/licenses/>.
*/

pragma solidity ^0.4.0;
```

```
contract owned {

    address public owner;

    function owned() payable {
        owner = msg.sender;
    }

    modifier onlyOwner {
        require(owner == msg.sender);
        _;
    }

    function changeOwner(address _owner) onlyOwner public {
        require(_owner != 0);
        owner = _owner;
    }
}

contract Crowdsale is owned {

    uint256 public totalSupply;
    mapping (address => uint256) public balanceOf;

    uint    public etherPrice;
    address public crowdsaleOwner;
    uint    public totalLimitUSD;
    uint    public minimalSuccessUSD;
    uint    public collectedUSD;

    enum State { Disabled, PreICO, CompletePreICO, Crowdsale, Enabled, Migration }
    event NewState(State state);
    State   public state = State.Disabled;
    uint    public crowdsaleStartTime;
    uint    public crowdsaleFinishTime;

    modifier enabledState {
        require(state == State.Enabled);
        _;
    }

    struct Investor {
        uint256 amountTokens;
        uint    amountETH;
    }
    mapping (address => Investor) public investors;
    mapping (uint => address)     public investorsIter;
```

```
uint                    public numberOfInvestors;

function () payable {
    require(state == State.PreICO || state == State.Crowdsale);
    uint256 tokensPerUSD = 0;
    if (state == State.PreICO) {
        tokensPerUSD = 125;
    } else if (state == State.Crowdsale) {
        if (now < crowdsaleStartTime + 1 days) {
            tokensPerUSD = 115;
        } else if (now < crowdsaleStartTime + 1 weeks) {
            tokensPerUSD = 110;
        } else {
            tokensPerUSD = 100;
        }
    }
    if (tokensPerUSD > 0) {
        uint valueETH = msg.value;
        uint valueUSD = valueETH * etherPrice / 1000000000000000000;
        if (collectedUSD + valueUSD > totalLimitUSD) { // don't need so much ether
            valueUSD = totalLimitUSD - collectedUSD;
            valueETH = valueUSD * 1000000000000000000 / etherPrice;
            msg.sender.transfer(msg.value - valueETH);
            collectedUSD = totalLimitUSD; // to be sure!
        }
        uint256 tokens = tokensPerUSD * valueUSD;
        require(balanceOf[msg.sender] + tokens > balanceOf[msg.sender]); // overflow
        require(tokens > 0);

        Investor memory inv = investors[msg.sender];
        if (inv.amountETH == 0) { // new investor
            investorsIter[numberOfInvestors++] = msg.sender;
        }
        inv.amountTokens += tokens;
        inv.amountETH += valueETH;
        investors[msg.sender] = inv;
        totalSupply += tokens;
        collectedUSD += valueUSD;
    }
}

function startTokensSale(address _crowdsaleOwner, uint _etherPrice) public onlyOwner {
    require(state == State.Disabled || state == State.CompletePreICO);
    crowdsaleStartTime = now;
    crowdsaleOwner = _crowdsaleOwner;
    etherPrice = _etherPrice;
    delete numberOfInvestors;
```

```
    delete collectedUSD;
    if (state == State.Disabled) {
        crowdsaleFinishTime = now + 14 days;
        state = State.PreICO;
        totalLimitUSD = 300000;
        minimalSuccessUSD = 300000;
    } else {
        crowdsaleFinishTime = now + 30 days;
        state = State.Crowdsale;
        totalLimitUSD = 5200000;
        minimalSuccessUSD = 3600000;
    }
    NewState(state);
}

function timeToFinishTokensSale() public constant returns(uint t) {
    require(state == State.PreICO || state == State.Crowdsale);
    if (now > crowdsaleFinishTime) {
        t = 0;
    } else {
        t = crowdsaleFinishTime - now;
    }
}

function finishTokensSale(uint _investorsToProcess) public onlyOwner {
    require(state == State.PreICO || state == State.Crowdsale);
    require(now >= crowdsaleFinishTime || collectedUSD >= minimalSuccessUSD);
    if (collectedUSD < minimalSuccessUSD) {
        // Investors can get their ether calling withdrawBack() function
        while (_investorsToProcess > 0 && numberOfInvestors > 0) {
            address addr = investorsIter[--numberOfInvestors];
            Investor memory inv = investors[addr];
            balanceOf[addr] -= inv.amountTokens;
            totalSupply -= inv.amountTokens;
            --_investorsToProcess;
            delete investorsIter[numberOfInvestors];
        }
        if (numberOfInvestors > 0) {
            return;
        }
        if (state == State.PreICO) {
            state = State.Disabled;
        } else {
            state = State.CompletePreICO;
        }
    } else {
        while (_investorsToProcess > 0 && numberOfInvestors > 0) {
```

```
            --numberOfInvestors;
            --_investorsToProcess;
            delete investors[investorsIter[numberOfInvestors]];
            delete investorsIter[numberOfInvestors];
        }
        if (numberOfInvestors > 0) {
            return;
        }
        if (state == State.PreICO) {
            if (!crowdsaleOwner.send(this.balance)) throw;
            state = State.CompletePreICO;
        } else {
            if (!crowdsaleOwner.send(1500000 * 1000000000000000000 / etherPrice)) throw;
            // Create additional tokens for owner (28% of complete totalSupply)
            balanceOf[msg.sender] = totalSupply * 28 / 72;
            totalSupply += totalSupply * 28 / 72;
            state = State.Enabled;
        }
    }
    NewState(state);
}

// This function must be called by token holder in case of crowdsale failed
function withdrawBack() public {
    require(state == State.Disabled || state == State.CompletePreICO);
    uint value = investors[msg.sender].amountETH;
    if (value > 0) {
        delete investors[msg.sender];
        msg.sender.transfer(value);
    }
}
}

contract Token is Crowdsale {

    string  public standard    = 'Token 0.1';
    string  public name        = 'PROOF';
    string  public symbol      = "PF";
    uint8   public decimals    = 0;

    modifier onlyTokenHolders {
        require(balanceOf[msg.sender] != 0);
        _;
    }

    mapping (address => mapping (address => uint256)) public allowed;
```

```
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);

    function Token() payable Crowdsale() {}

    function transfer(address _to, uint256 _value) public enabledState {
        require(balanceOf[msg.sender] >= _value);
        require(balanceOf[_to] + _value >= balanceOf[_to]); // overflow
        balanceOf[msg.sender] -= _value;
        balanceOf[_to] += _value;
        Transfer(msg.sender, _to, _value);
    }

    function transferFrom(address _from, address _to, uint256 _value) public {
        require(balanceOf[_from] >= _value);
        require(balanceOf[_to] + _value >= balanceOf[_to]); // overflow
        require(allowed[_from][msg.sender] >= _value);
        balanceOf[_from] -= _value;
        balanceOf[_to] += _value;
        allowed[_from][msg.sender] -= _value;
        Transfer(_from, _to, _value);
    }

    function approve(address _spender, uint256 _value) public enabledState {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
    }

    function allowance(address _owner, address _spender) public constant enabledState
        returns (uint256 remaining) {
        return allowed[_owner][_spender];
    }
}

contract MigrationAgent {
    function migrateFrom(address _from, uint256 _value);
}

contract TokenMigration is Token {

    address public migrationAgent;
    uint256 public totalMigrated;

    event Migrate(address indexed from, address indexed to, uint256 value);

    function TokenMigration() payable Token() {}
```

```
    // Migrate _value of tokens to the new token contract
    function migrate(uint256 _value) external {
        require(state == State.Migration);
        require(migrationAgent != 0);
        require(_value != 0);
        require(_value <= balanceOf[msg.sender]);
        balanceOf[msg.sender] -= _value;
        totalSupply -= _value;
        totalMigrated += _value;
        MigrationAgent(migrationAgent).migrateFrom(msg.sender, _value);
        Migrate(msg.sender, migrationAgent, _value);
    }

    function setMigrationAgent(address _agent) external onlyOwner {
        require(migrationAgent == 0);
        migrationAgent = _agent;
        state = State.Migration;
    }
}

contract ProofTeamVote is TokenMigration {

    function ProofTeamVote() payable TokenMigration() {}

    event VotingStarted(uint weiReqFund);
    event Voted(address indexed voter, bool inSupport);
    event VotingFinished(bool inSupport);

    struct Vote {
        bool inSupport;
        bool voted;
    }

    uint public weiReqFund;
    uint public votingDeadline;
    uint public numberOfVotes;
    uint public yea;
    uint public nay;
    mapping (address => Vote) public votes;
    mapping (uint => address) public votesIter;

    function startVoting(uint _weiReqFund) public enabledState onlyOwner {
        require(weiReqFund == 0 && _weiReqFund > 0 && _weiReqFund <= this.balance);
        weiReqFund = _weiReqFund;
        votingDeadline = now + 7 days;
        VotingStarted(_weiReqFund);
    }
```

```
function votingInfo() public constant enabledState
    returns(uint _weiReqFund, uint _timeToFinish) {
    _weiReqFund = weiReqFund;
    if (votingDeadline <= now) {
        _timeToFinish = 0;
    } else {
        _timeToFinish = votingDeadline - now;
    }
}

function vote(bool _inSupport) public onlyTokenHolders enabledState
    returns (uint voteId) {
    require(votes[msg.sender].voted != true);
    require(votingDeadline > now);
    voteId = numberOfVotes++;
    votesIter[voteId] = msg.sender;
    votes[msg.sender] = Vote({inSupport: _inSupport, voted: true});
    Voted(msg.sender, _inSupport);
    return voteId;
}

function finishVoting(uint _votesToProcess) public enabledState onlyOwner
    returns (bool _inSupport) {
    require(now >= votingDeadline && weiReqFund <= this.balance);

    while (_votesToProcess > 0 && numberOfVotes > 0) {
        address voter = votesIter[--numberOfVotes];
        Vote memory v = votes[voter];
        uint voteWeight = balanceOf[voter];
        if (v.inSupport) {
            yea += voteWeight;
        } else {
            nay += voteWeight;
        }
        delete votes[voter];
        delete votesIter[numberOfVotes];
        --_votesToProcess;
    }
    if (numberOfVotes > 0) {
        _inSupport = false;
        return;
    }

    _inSupport = (yea > nay);

    if (_inSupport) {
```

```
            if (migrationAgent == 0) {
                if (!owner.send(weiReqFund)) throw;
            } else {
                if (!migrationAgent.send(weiReqFund)) throw;
            }
        }

        VotingFinished(_inSupport);
        delete weiReqFund;
        delete votingDeadline;
        delete numberOfVotes;
        delete yea;
        delete nay;
    }
}

contract ProofPublicVote is ProofTeamVote {

    function ProofPublicVote() payable ProofTeamVote() {}

    event Deployed(address indexed projectOwner, uint proofReqFund, string urlInfo);
    event Voted(address indexed projectOwner, address indexed voter, bool inSupport);
    event VotingFinished(address indexed projectOwner, bool inSupport);

    struct Project {
        uint   proofReqFund;
        string urlInfo;
        uint   votingDeadline;
        uint   numberOfVotes;
        uint   yea;
        uint   nay;
        mapping (address => Vote) votes;
        mapping (uint => address) votesIter;
    }
    mapping (address => Project) public projects;

    function deployProject(uint _proofReqFund, string _urlInfo) public onlyTokenHolders
enabledState {
        require(_proofReqFund > 0 && _proofReqFund <= balanceOf[this]);
        require(_proofReqFund <= balanceOf[msg.sender] * 1000);
        require(projects[msg.sender].proofReqFund == 0);
        projects[msg.sender].proofReqFund = _proofReqFund;
        projects[msg.sender].urlInfo = _urlInfo;
        projects[msg.sender].votingDeadline = now + 7 days;
        Deployed(msg.sender, _proofReqFund, _urlInfo);
    }
```

```
    function projectInfo(address _projectOwner) enabledState public
        returns(uint _proofReqFund, string _urlInfo, uint _timeToFinish) {
        _proofReqFund = projects[_projectOwner].proofReqFund;
        _urlInfo = projects[_projectOwner].urlInfo;
        if (projects[_projectOwner].votingDeadline <= now) {
            _timeToFinish = 0;
        } else {
            _timeToFinish = projects[_projectOwner].votingDeadline - now;
        }
    }

    function vote(address _projectOwner, bool _inSupport) public onlyTokenHolders
enabledState
        returns (uint voteId) {
        Project storage p = projects[_projectOwner];
        require(p.votes[msg.sender].voted != true);
        require(p.votingDeadline > now);
        voteId = p.numberOfVotes++;
        p.votesIter[voteId] = msg.sender;
        p.votes[msg.sender] = Vote({inSupport: _inSupport, voted: true});
        projects[_projectOwner] = p;
        Voted(_projectOwner, msg.sender, _inSupport);
        return voteId;
    }

    function finishVoting(address _projectOwner, uint _votesToProcess) public enabledState
        returns (bool _inSupport) {
        Project storage p = projects[_projectOwner];
        require(now >= p.votingDeadline && p.proofReqFund <= balanceOf[this]);

        while (_votesToProcess > 0 && p.numberOfVotes > 0) {
            address voter = p.votesIter[--p.numberOfVotes];
            Vote memory v = p.votes[voter];
            uint voteWeight = balanceOf[voter];
            if (v.inSupport) {
                p.yea += voteWeight;
            } else {
                p.nay += voteWeight;
            }
            delete p.votesIter[p.numberOfVotes];
            delete p.votes[voter];
            --_votesToProcess;
        }
        if (p.numberOfVotes > 0) {
            projects[_projectOwner] = p;
            _inSupport = false;
            return;
```

```
        }

        _inSupport = (p.yea > p.nay);

        if (_inSupport) {
            transfer(_projectOwner, p.proofReqFund);
        }

        VotingFinished(_projectOwner, _inSupport);
        delete projects[_projectOwner];
    }
}

contract Proof is ProofPublicVote {

    struct Swype {
        uint16  swype;
        uint    timestampSwype;
    }

    struct Video {
        uint16  swype;
        uint    timestampSwype;
        uint    timestampHash;
        address owner;
    }

    mapping (address => Swype) public swypes;
    mapping (bytes32 => Video) public videos;

    uint priceInTokens;
    uint teamFee;

    function Proof() payable ProofPublicVote() {}

    function setPrice(uint _priceInTokens) public onlyOwner {
        require(_priceInTokens >= 2);
        teamFee = _priceInTokens / 10;
        if (teamFee == 0) {
            teamFee = 1;
        }
        priceInTokens = _priceInTokens - teamFee;
    }

    function swypeCode() public enabledState returns (uint16 _swype) {
        bytes32 blockHash = block.blockhash(block.number - 1);
        bytes32 shaTemp = sha3(msg.sender, blockHash);
```

```
        _swype = uint16(uint256(shaTemp) % 65536);
        swypes[msg.sender] = Swype({swype: _swype, timestampSwype: now});
    }

    function setHash(uint16 _swype, bytes32 _hash) public enabledState {
        require(balanceOf[msg.sender] >= priceInTokens);
        require(swypes[msg.sender].timestampSwype != 0);
        require(swypes[msg.sender].swype == _swype);
        transfer(owner, teamFee);
        transfer(this, priceInTokens);
        videos[_hash] = Video({swype: _swype,
timestampSwype:swypes[msg.sender].timestampSwype,
            timestampHash: now, owner: msg.sender});
        delete swypes[msg.sender];
    }
}
```