Teambrella: A Peer-to-Peer Coverage System

Version 2.2

Alex Paperno
alex@teambrella.com

Vlad Kravchuk
vlad@teambrella.com

Eugene Porubaev
eugene@teambrella.com

Abstract

There is a conflict of interest between insurance companies and policyholders. We propose a way to solve this issue by implementing a system that provides peer-to-peer coverage. Peers have mutual control on most aspects of their coverage, such as risk evaluation and processing of payments. The mutual control is implemented via a voting mechanism that can be delegated to proxies. To ease the burden of payments between peers, we use a blockchain as a mean of providing coverage and payment of reimbursements.

Introduction

Every insurance company pools funds from policyholders. The more money left in the pool, the higher is the company's profit. In other words, it is profitable for the company to deny even valid claims. As a result, insurance companies and policyholders have conflicting interests.

This is reflected in various bad-faith practices of the companies, such as unreasonable delays, denial of payment, etc. Though some jurisdictions try to mitigate this by adopting the tort of insurance bad faith, the general public has a very negative perception of the industry. This, in turn, leads to a high amount of frauds committed by otherwise law-abiding persons. Surveys show that nearly half the public would do nothing if they became aware of an instance of insurance fraud [4] and 24 percent even find such fraud acceptable [3].

Anti-fraud measures are costly and hostile. The end result is unfair treatment, bureaucracy, lack of transparency and prices that are much higher than customers' expectations. For auto insurance in the USA, a typical loss ratio is about 60 percent [1] and adjustment expenses make up another 10 percent of premiums [2]. Only half of the premiums are paid back as reimbursements.

Both unfair prices and bad-faith practices can be significantly mitigated by implementing a peer-to-peer (P2P) coverage organization. Peers control via voting each and every decision and are free to instantly delegate their votes to other peers, creating chains of trust. As long as there is no central money-handling party, every reimbursement payment to a peer is, in fact, a payment of premiums from other peers. Though the total amount of premiums is not fixed, peers have full control over their spending, which can be two times lower on average.

While for every claim there is still a payee and payer(s), it is in every paying peer's interest to do what they believe to be fair. By doing so, the payer sets the standards of treatment that would be applied to them in case of an incident. This practically removes conflict of interest and significantly reduces burdens otherwise put on policyholders.

The framework for the P2P organization includes a decision-making layer and a payment layer. The payment layer is based on blockchain technology. The decision-making layer consists of a server(s) used as a medium for communication and voting.

Teams

Teams are self-governing user communities. A team consists of peers (teammates) that collectively manage all coverage functions, such as:

- setting of coverage rules (i.e., what is covered, what documents need to be submitted, etc.)
- signing of new members
- appraisal of claims and approval of reimbursements
- payment of reimbursements

Teams can be created on the basis of likeness of peers or covered objects:

- kind of covered object (e.g., car, house, health)
- kind of covered incident (e.g., collision, damage to 3rd party)
- social or professional affinity (e.g., World of Warcraft players, McDonald's workers or army veterans)
- home/work location (e.g., living in same town, working in same office building)

Any person can create a team and define its initial set of rules. Coverage is activated once a minimal number of peers join the team (two by default) and fund their distributed wallets.

Distributed Wallets

Teammates make reimbursement payments from cryptocurrency wallets they control. Private keys to those wallets are stored on respective client systems only and are never transmitted outside.

Upon first launch, a trusted client application (e.g. an open-source one) creates a key pair (private key and corresponding public key). The pair is the main key pair for the new teammate's distributed wallet. This ensures that no coins can be spent from this wallet without their consent. The wallet is additionally controlled by N out of M other cosigners selected by the server. The teammate is allowed to withdraw their coins with the consent of other teammates, i.e., insuring that all outstanding premiums can be paid out. When the teammate is selected as a cosigner of another teammate's wallet, the cosigning is carried out with the same key. The key is also used for signing requests to the server. Finally, it can be used in case of server-less team disbanding.

For UTXO blockchains, such as Bitcoin or Bitcoin Cash, the wallets are P2SH ones, with a redeem script using $1 + N$ out of M signatures. N and M are selected so that the corresponding redeem transactions are relatively short, yet a sufficiently high level of security is provided for storing coins in such wallets. For teams with at least 9 active members, $N = 3$ and $M = 8$. An attacker would need to gain control of 4 out of 9 systems at the same time. (See also: Attack Vector Analysis).

For Ethereum blockchain, the wallets are deployed as Ethereum contracts.

The server(s) automatically adjusts lists of cosigners by creating new addresses and initiating transfers of coins to them. This way it handles cases of peers joining or leaving teams. Wallet

owners and cosigners control this process in a way similar to payment of reimbursements: each change of the address by the server(s) must be approved by both the wallet owner and its cosigners. (See also: Payment of Reimbursements).

Risk Coefficient

Each newly joining teammate negotiates an *insurable value* with the team. The insurable value is usually the price of the insured object or the maximum loss that can be incurred. For some kinds of coverage (e.g., health coverage) the insurable value can be the same for all teammates and be defined by the team rules. The insurable value limits the maximum amount of expenses that can be declared in a single claim by a teammate.

A teammate's *expense expectation* is an estimated average value of expenses related to a potential incident involving the teammate. We define the relation between expense expectation and insurable value as expense expectation ratio:

$$EER_{teammate} = \frac{ExpenseExpectation_{teammate}}{InsurableValue_{teammate}}$$

Teammates and their insured objects differ in probabilities of an incident, insurable values, and expense expectations. To compensate these differences and make coverage fair for each teammate, we introduce a *risk coefficient*:

$$Risk_{teammate} = \frac{PI_{teammate}}{PI_{average}} \times \frac{EER_{teammate}}{EER_{average}}$$, where

$PI_{teammate}$, $PI_{average}$ are probabilities of an incident for the teammate and average within the team, respectively, for a same period of time and

$EER_{average}$ is the average EER within the team.

Other things being equal, the higher a teammate's risk coefficient, the more premiums they would need to pay.

> Example 1:
>
> A team insures against collision damage. An average car is not a new one with a value of $15,000 and an expense expectation of $3,000. This makes $EER_{average}$ = 0.2. Peter joins the team with a brand new car with insurable value of $30,000. Despite the higher value of the car, the cost of repair after a potential non-total incident could be the same as for the team's average car. Taking into account the possibility of a total incident, Peter's expense expectation is $4,500, $EER_{teammate}$ = 0.15. Besides, Peter's probability of getting into an accident could also be a bit lower: $PI_{teammate}/PI_{average}$ = 0.8. Based on that, his risk coefficient could be as low as 0.6.

A truly fair value of a teammate's risk coefficient is practically impossible to determine due to inability to take into account every factor that affects the probability of an incident. A new teammate's risk coefficient is initially set via voting by other teammates. When the peer joins the

team, he accepts that the risk coefficient offered by the team can be considered as fair. (See also: Addition of New Members to Team). The coefficient can be changed later according to the team rules. For example, it may be decreased automatically after one year if the teammate has paid enough premiums but has not submitted a claim during the year.

For standard types of coverage, the central server(s) estimates a default risk coefficient for each newly joining person. The estimation is based on data provided by the applicant, publicly known statistics, and/or team rules.

Peer-To-Peer Coverage

Each teammate is both a provider and a consumer of an coverage service for every member of the team, including himself. The fairness of such P2P coverage can be achieved by arranging a symmetric relationship within each pair of teammates. This implies that teammate A's P2P coverage of teammate B matches teammate B's P2P coverage of teammate A with respect to the corresponding risk coefficients:

$$P2PCoverage_{A,B} \times Risk_A = P2PCoverage_{B,A} \times Risk_B \qquad \text{(i)}$$

$P2PCoverage_{A,B}$ is the maximum amount that teammate A can be reimbursed by teammate B for a single claim. A teammate's total amount of P2P coverages by all team members is the teammate's *coverage limit*:

$$CoverageLimit_A = \sum_{B \in Team} P2PCoverage_{A,B} \qquad \text{(ii)}$$

The coverage limit is the maximum reimbursement that a team is able to pay to a teammate for a single claim. The coverage limit differs for each team member and is automatically adjusted over time.

> Example 2:
>
> Alice and John are in a team that insures against collision damage. Alice's car is valued at $30,000 and John's one is at $10,000. The team is a big one, so they got full coverage for their cars.
>
> Both Alice and John have the same risk coefficient of 0.8. Alice's P2P coverage of John is 100$, so John's P2P coverage of Alice is the same, i.e., $100.
>
> If Alice crashes her car, then she would be reimbursed $30,000. From that amount, she would get $100 from John and $29,900 from other teammates. If John's car is destroyed, then he would be reimbursed $10,000 ($100 from Alice and $9,900 from the rest of the team).
>
> Example 3:

> Simon's risk coefficient is 1.0 and Peter's is 0.5.
> Therefore, if Simon's P2P coverage of Peter is $100, then Peter's coverage of John is $50.

Methods of calculation of P2P coverage are out of the scope of this paper. However, it must be mentioned that each teammate's P2P coverage of other teammates is limited by:

a. Amount of coins in the teammate's distributed wallet

   A teammate's liability never exceeds the amount of coins they control.

b. A preset premium payment limit

   It is not possible for a teammate to predict the exact total amount of their premium payments during a time frame. The way to control spending is by setting a fixed limit for each single premium payment.

   Alternatively or additionally, the team may install a rule to limit premium payments to some part of each teammate's wallet. This way it is guaranteed that the team would have some funds to cover future incidents after a single large claim is reimbursed.

c. Other teammates' P2P coverage values

   According to the fairness rule (i), the maximum that a teammate can be made liable to pay to another teammate is bound by the other party's P2P coverage of the teammate.

   > Example 4:
   >
   > Adam and Gregory are in a same team and they have the same risk coefficients. Gregory doesn't want to pay a fortune if a teammate has in incident, so he sets a fixed limit of $50 per single premium payment. Now, even though Adam is ok to pay $200 for an incident of a teammate, he will not pay more than $50 to Gregory.

d. The teammate's insurable value

   If a teammate's P2P coverage values warrant the coverage limit equal to the insurable value, then these cannot be increased any further.

Coverage Ratio

When a teammate's coverage limit is smaller than their insurable value, they are only partially insured. In the case of an incident, their expenses are to be reimbursed proportionally to the *coverage ratio*:

$$CoverageRatio_{teammate} = \frac{CoverageLimit_{teammate}}{InsurableValue_{teammate}}$$

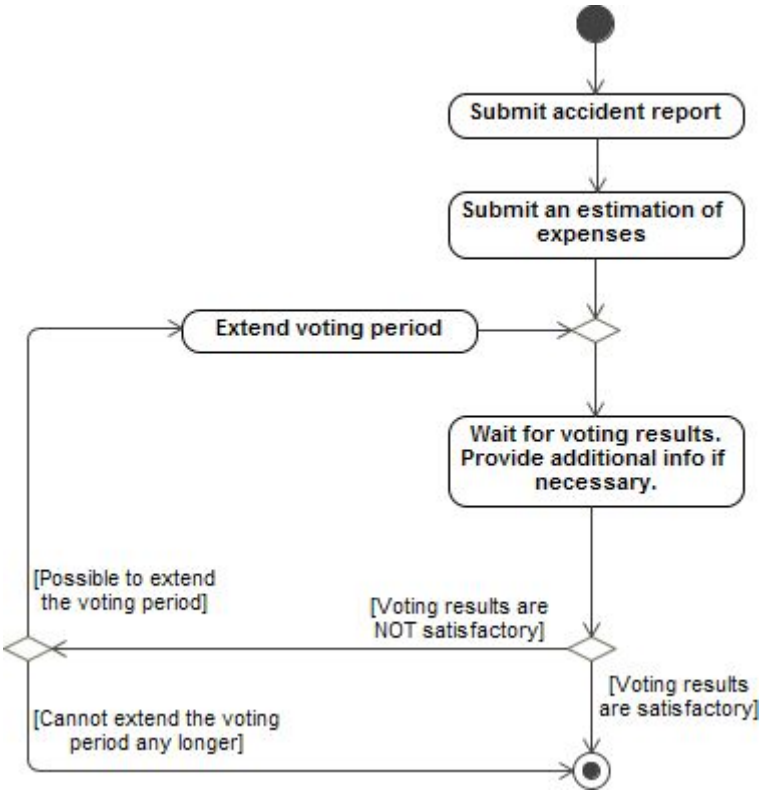
> Example 5:
>
> Carol's insurable value is $10,000. The team is not a big one, so her coverage limit is only $5,000, i.e. the coverage ratio is 50%. In the case of an insured incident with real expenses of $2,000, the maximum amount they can be reimbursed would be $1,000 (i.e., $2,000 * 50%).

A teammate's coverage ratio is visible in the user interface at all times. This allows the teammate to grasp what reimbursement they can expect in the case of an incident.

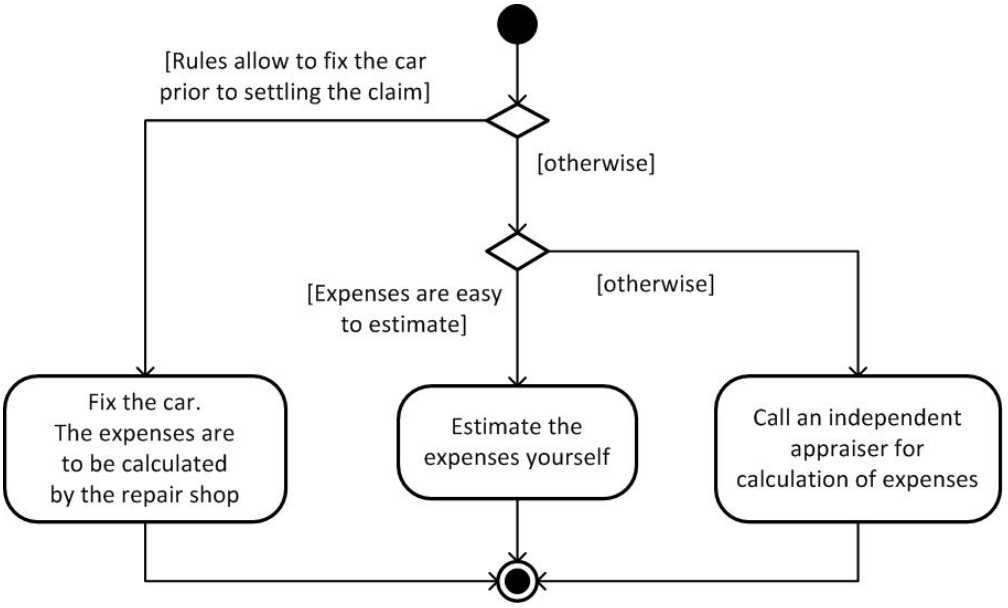
Claim Submission

When an insured incident occurs to a teammate, they need to notify other team members by submitting a claim. They discuss the claim and the claim submitter provides information regarding the accident and the claim. Current voting results and a list of votes cast are available to all team members at any moment while the voting is on. The voting automatically finishes in several days. A possible outcome is that the claim submitter has not persuaded other teammates of claim validity by that time and voting results are not positive for them. In this case, the claimant can extend the voting period if it is allowed by the team rules.

The workflow for a claiming teammate is presented below:



The process of estimation of expenses differs for various covered objects, coverage types, and team rules. For simple cases, such as the loss of a fancy smartphone, it could be just the price of a new one. For auto collision coverage claims it could be as follows:



It may later turn out that some substantial expenses are not included in the total by the claim submitter, e.g., due to a hidden damage. In such cases, a follow-up claim can be submitted if it is in compliance with the team rules. Total claimed expenses for a single claim cannot exceed the teammate's insurable value.

Voting on Reimbursement

The claim submitter provides estimation of expenses that limits the maximum reimbursement. If the team set a deductible via the team rules or the claim submitter's coverage ratio is not 100%, then the maximum reimbursement is automatically reduced:

$$MaxReimbursement_{claim} = (ExpensesClaimed_{claim} - Deductible) \times CoverageRatio_{claimer}$$

The team may grant any amount from 0% to 100% of the maximum reimbursement. The exact amount of reimbursement is determined by voting:

$$Reimbursement_{claim} = MaxReimbursement_{claim} \times VotingResult_{claim}$$

The weight of a teammate's vote is proportional to the total amount of their premium payments for several previous months. The voting result is calculated as the median value of votes, i.e., the total weight of votes cast for reimbursing more is the same as the total weight of votes cast for reimbursing less. Use of the median value lowers the effect of possible tactical voting in comparison with the mean value.

> Example 7:

> Simon's team votes for reimbursement of a claim. Simon believes that it would be fair to pay 80%, but the current voting result is only 61%. He decides to change his vote to 100%, but as long as his vote stays on the same side of 61%, the voting result does not change.

Each teammate's premium payment is calculated during the voting so that any teammate can see what his personal payment would be:

$$PP_{teammate,\,claim} = Reimbursement_{claim} \times \frac{P2PCoverage_{claimer,\,teammate}}{CoverageLimit_{claimer}}$$

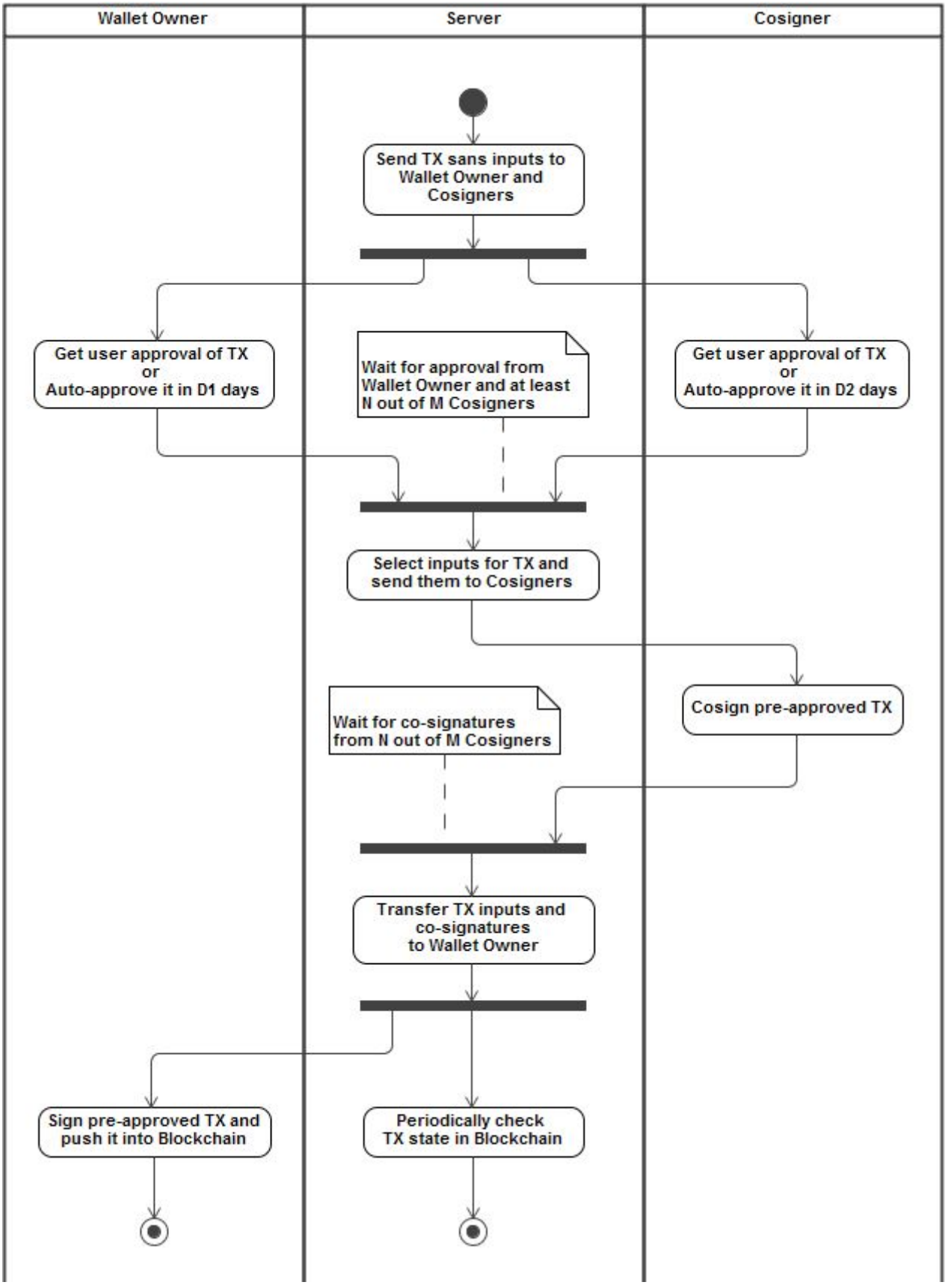
Payments and Withdrawal

Upon closing of voting the server(s) prepares a set of blockchain transactions (for Bitcoin/Bitcoin Cash) or contract messages (for Ethereum) from team members' distributed wallets to the claim submitter's pay-to address. Additional transactions/messages may be created for compensations for voting.

To reduce the amount of blockchain network fees, team rules may allow the server(s) to combine premium payments of small value. In this case some teammates do not take part in the payment and others pay more. The server(s) stores the difference between the fair share payment and the real payment. This difference is taken into account on the next payout so that it never exceeds a preset amount, e.g., $10.

A wallet owner may initiate a withdrawal transaction. The server(s) approves the transaction if the remaining funds on the wallet would be sufficient to pay the teammate's premium payments for all currently unsettled claims.

For UXTO blockchains, processing of each transaction is carried out as follows:


Wallet Owner
Server
Cosigner
Send TX sans inputs to Wallet Owner and Cosigners
Get user approval of TX or Auto-approve it in D1 days
Wait for approval from Wallet Owner and at least N out of M Cosigners
Get user approval of TX or Auto-approve it in D2 days
Select inputs for TX and send them to Cosigners
Cosign pre-approved TX
Wait for co-signatures from N out of M Cosigners
Transfer TX inputs and co-signatures to Wallet Owner
Sign pre-approved TX and push it into Blockchain
Periodically check TX state in Blockchain

The server(s) transmit the outputs of the transaction to the wallet owner and each cosigner of the wallet.

The client software on the wallet owner's and the cosigners' systems allows users to check the transaction's outputs and approve the transaction. If the transaction looks suspicious, then a user may disapprove it. If the user neither approves nor disapproves the transaction within several days (D1 = 3, D2 = 3 by default), then the client software auto-approves the transaction and notifies the server.

The client software maintains a list of known pay-to addresses. Payment to new addresses may be delayed for additional several days (D1 = 7, D2 = 7). (See also: Attack Vector Analysis).

When the transaction is approved, the server identifies a set of UTXOs to use as inputs for the transaction. It then passes this set to all the cosigners. Those cosigners that already approved the transaction cosign it without additional confirmations from users and transmit the co-signatures to the server.

The server further transmits the co-signatures along with the set of inputs to the wallet owner's client software. Here the transactions gets fully signed and is published in the blockchain via a pushtx service (https://blockchain.info/pushtx, https://btc.blockr.io/tx/push, etc.).

The server(s) checks the state of the transaction. If after seven days since being cosigned the transaction is still not published in the blockchain, then the owner of the distributed wallet may be penalized according to the team rules. To process transactions in a timely manner, each teammate's client software should be online at least once every 2–3 days.

For Ethereum blockchain, the procedure is essentially the same. The only difference is that there are no steps of identifying the set of UTXOs to use as inputs and providing these inputs to the wallet owner.

Addition of New Members to Team

A team may decide whether an invitation is required for joining the team. It may also determine if recently joined members are allowed to send out invitations.

A person wanting to join the team needs to submit an application. The application should contain all the information required by the team rules. For example, for auto collision coverage, it may be required to submit a description and photos of the car, information about drivers, previous accidents, etc. The application must include insurable value if it is not defined in the team rules.

Upon receiving an application, the server(s) initiates voting on the new member's risk coefficient. Similar to voting on reimbursement, a median value is selected, I.e. the result is determined in such a way that the total weight of votes cast for risk coefficients lower than the selected one is the same as for greater ones. During the voting period, teammates may require additional information from the applicant. The final value of the risk coefficient is determined after the voting is over (seven days by default). After the applicant approves the risk coefficient he becomes a new team member.

Team Rules

All aspects of interactions within a team are regulated by *team rules*. Rules are set by team members and enacted/maintained by the server(s). Until accepting a second team member to the team, the creator of the team may alter the rules in any way. In a team of several members, each change to the rules is a result of voting.

Team rules consist of predefined as well as custom rules suggested by the team members. For each predefined rule there is a parameter that can be modified. For a custom rule a modification can be either new text or deletion of the rule.

Voting is carried out automatically on an hourly basis. For a parameter that has several options (e.g., on/off, yes/no) the new value can be set if it is supported by the majority of teammates' votes. This also applies to all custom rules. New values of numeric parameters are determined as median values (i.e., the total weight of votes for lower values and for larger values are the same).

Changes to custom rules and parameters with several options go into effect after several days (seven by default) if no additional changes has been made during that period. For each numeric parameter a running average with a window of several days (seven by default) is used as the parameter's effective value.

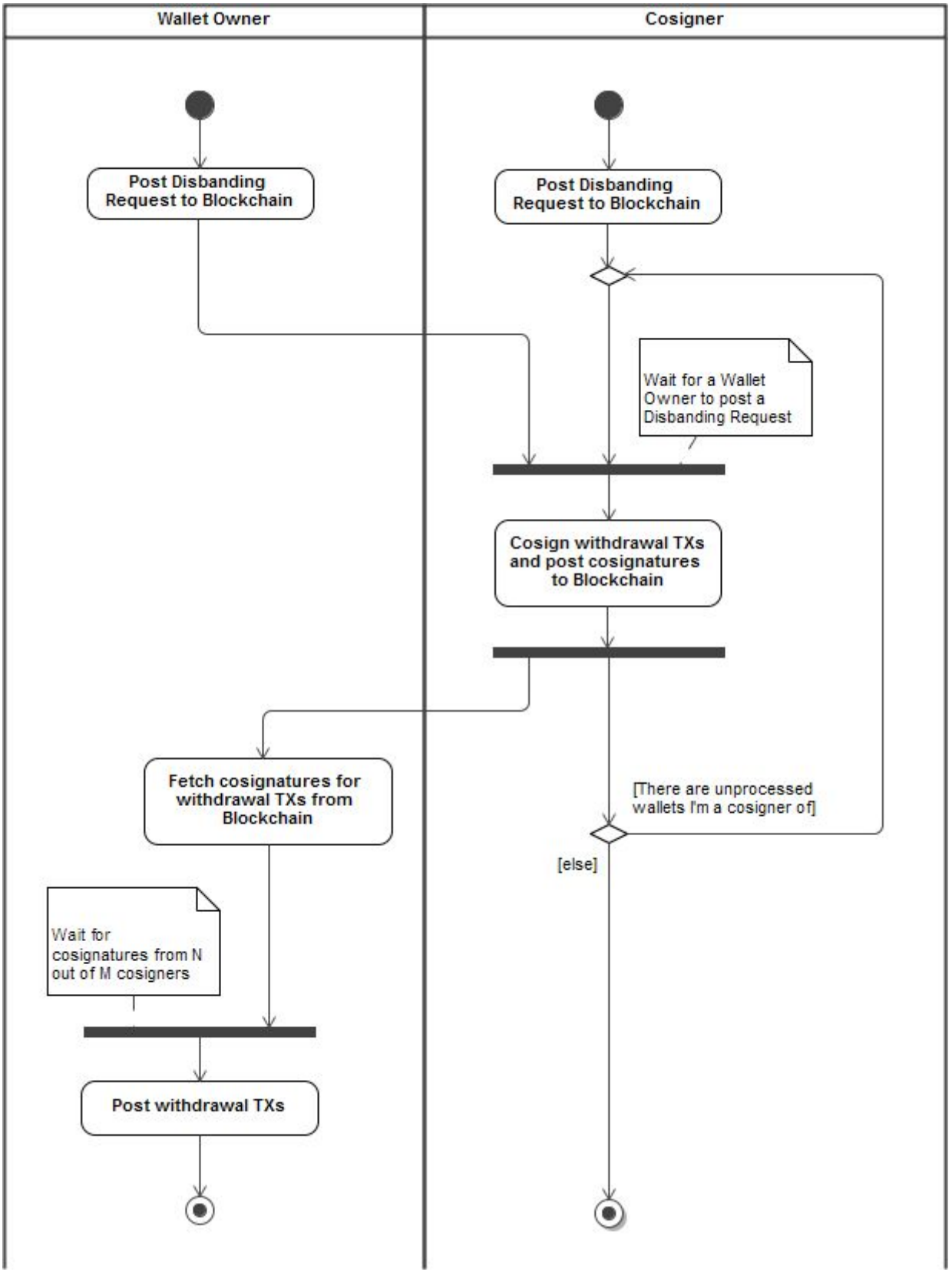
Team Disbanding

Members of a team may collectively transfer funds from their distributed wallets with no interaction with the server(s) at any time. If the server(s) became compromised, then team members would launch the client software in a special "disband" mode. To speed up the process, they can coordinate it via a social network or forum.

Initially, the client software publishes a disbanding request to the blockchain. Once the request is published, the client software stops processing any requests from the server related to the team being disbanded.

For the case of Ethereum network, the request is posted as a set of special disband messages to the Ethereum wallets that the client is a cosigner or an owner of. When a wallet receives at least *N* out of *M* (e.g., 3 out of 8) disband messages from its co-signatures as well as a message from the wallet owner, it releases the funds to the address specified by the disband message of the wallet owner.

In UTXO blockchains, the request is posted as a payloaded transaction from the teammate's personal wallet (the one with a P2PKH address derived from the teammate's pubkey [5]). The payload contains a pay-to address specified by the user. Subsequent data exchange uses payloaded transactions in the same way.



Withdrawal transactions are ones that transfer coins from the distributed wallets to the pay-to addresses of the respective wallets' owners. It might be that the teammate is not a cosigner of any other teammate's wallet. In these cases, the client software just waits for co-signatures for withdrawal transactions to be published in the blockchain by the cosigners. Once it fetches *N* out

of *M* (e.g., 3 out of 8) co-signatures, it fully signs the withdrawal transactions and pushes them to the blockchain.

If the teammate is a cosigner of some wallets, then the client software also awaits disbanding requests from each cosigned wallet's owner. Once a request is received, it cosigns respective withdrawal transactions and publishes the co-signatures in the blockchain.

Attack Vectors

Threat actor: compromised/fraudulent server

Threat	**Prevention**
Block access to teammates' funds: a) shut down server; b) don't process legitimate requests.	Teammates may collectively transfer coins from distributed wallets. (See: Team Disbanding.)
Unauthorized move of coins from a teammate's wallet: create a transaction spending from the wallet and send it for approval to teammates.	The attacker would have to hope that the fraudulent transaction is unnoticed until it gets auto-approved. The auto-approval interval is seven days by default for a new pay-to address and three days for a known address. The probability of being unnoticed decays exponentially with growing of the number of wallets under attack.

Threat actor: fraudulent teammate

Threat	**Prevention**
Try to take advantage of other teammates: In small teams, an attacker might want to enjoy the coverage until a major accident is to be reimbursed. Then they would want to quit the system and get their money back instead of paying the premium.	The attacker would need to persuade 3 out of 8 cosigners to help him create an out-of-system withdrawal transaction. Such cosigners would be flagged and might have funds on their distributed wallets suspended by their teammates.
Make a false claim.	Teammates are not limited with a standard set of verification procedures and may require any amount of additional proofs. Fraud against peers is also much less tolerated than against "the system": 10 percent of Americans agree that "insurance fraud doesn't hurt anyone" [6].

Refuse to sign (block) a legitimate TX. Optionally: blackmail the wallet owner.	It would take 6 out of 8 people to conspire to do this. In most cases teammates are non-anonymous, e.g. with known car plates (for auto coverage) and FB logins. Anyone who tries this anyway, would be at risk of losing control of their own wallet (cosigned by other teammates).

Threat actor: non-cooperating teammate

Threat	**Prevention**
Refuse to run the client software, e.g. after stopping of using the service (block legitimate transactions).	If a cosigner stops running client software for a long time, then the system schedules move of coins to another wallet, removing this person from the cosigners list.

Threat actor: compromised teammate

Threat	**Prevention**
Unauthorized move of coins from the wallet: create a transaction spending from the wallet.	A distributed wallet is controlled by the wallet's main key + 3 out of 8 cosigning keys stored on other teammate's systems. An attacker would also need to take control of three different cosigning teammate systems.
Unauthorized move of coins from the wallet: a) create a withdrawal request; b) make a false claim.	The real wallet owner can disapprove malicious transactions. They also may use a master key / social network login to block access from compromised client software.
Prevent legitimate use of the wallet by its real owner: Disapprove legitimate transactions.	The real wallet owner may use a master key / social network login to block access from compromised client software.

Proxy Voting

A teammate does not have to formulate an opinion on each and every decision; that is, they can delegate their vote to other teammates through a *proxy voting list*. By default, a newly joining teammate's list contains her inviter. The list is sorted: if another teammate from the top of the list casts a vote, then the votes of teammates below are no longer considered. Anytime during the voting period a teammate can cast their vote and cancel the vote cast by a proxy on the teammate's behalf.

In small teams, insurable incidents happen rarely and premium payments per an incident are relatively large. The motivation to vote in such teams can be substantial. However, in a larger team, most teammates may find that it does not justify spending time on frequent voting on issues with almost no financial involvement.

To make voting more appealing, the team pays a sum equivalent to 20% of reimbursement amount as compensation to those who do cast a vote:

$$TotalVoteComp_{claim} = Reimbursement_{claim} \times 20\%$$

Accordingly, each teammate's individual contribution to the voting compensation is proportional to the teammate's premium payment:

$$PC_{T,claim} = PP_{T,claim} \times 20\%$$

Each voting teammate's compensation reflects the number of teammates that trusted him or her to be their proxy voter:

$$VoteComp_{T,\,claim} = \sum_{A \in Repr_{T,claim}} PC_{A,claim} + \frac{Weight_{T,claim}}{WeightSum_{claim}} \times \sum_{B \in NoRepr_{claim}} PC_{B,claim}$$

,

where

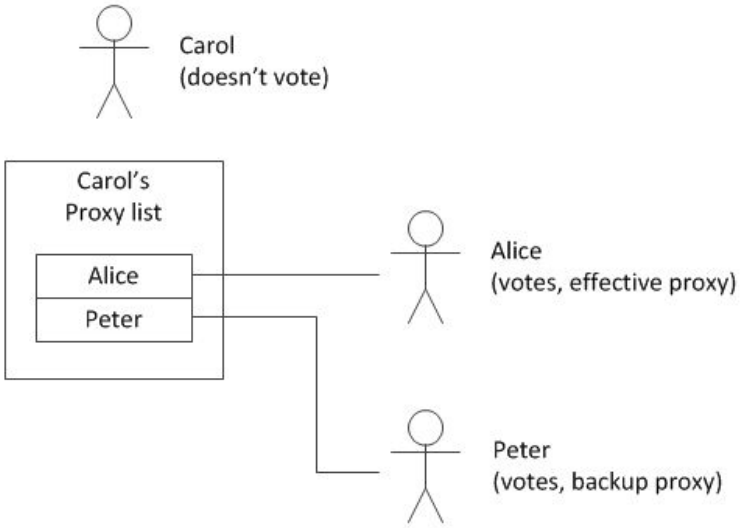
$Repr_{T,claim}$ is the set including the teammate and those having teammate T as the effective proxy during the voting for the claim,

$NoRepr_{claim}$ is the set including all the teammate that neither voted nor had a voting proxy,

$Weight_{T,claim}$ is the sum of vote weights of all teammates from $Repr_{T,claim}$, and

$WeightSum_{claim}$ is the sum of vote weights of all voting teammates.

Thus, non-voting teammates pay an additional 20% to proxies who vote on their behalf. On the other hand, voting teammates don't make any additional payments and may be compensated for their time and expertise.

> Example 6:
>
> Alice votes as a proxy for Carol. After the voting is finished and the claim is reimbursed, Alice is paid proportional to the sum of their premium payments.
>
> 
>

Teammates that are trusted by many others in big teams may be paid enough to do this job at least part-time. Fairness and correctness of their votes directly affect the number of people who trust them and, in turn, their income from being a proxy. This induces competition and control of among such pro voters: Any event of an unfairness or a dishonesty of a rival voter would be promptly reported to the society. In turn, the presence of such control makes the voters do their best.

Most of the work that proxy voters do is similar that of appraisers, fraud detection analysts, and other insurance company representatives. They are motivated to obtain licenses for insurance-related software (fraud-detection, appraisal, etc.). Naturally, a significant proportion of proxy voters can be freelance insurance professionals or employees of insurance companies interested in new markets. Proxy voters may have no covered objects at all and, therefore, can be invited to a team only to perform the voting job.

Conclusion

We have proposed a system for peer-to-peer coverage. Users of the system can create or join teams, where each member is a peer. Each peer is both a provider and a consumer of the coverage service and each premium payment is, in fact, a part reimbursement of a claim. Peers collectively administer all functions of their team by voting. To make the voting process efficient we propose chainable proxy voting, with voters being compensated for their time. To enforce premium payments we introduced distributed Ethereum wallets that prevent spending not sanctioned by other peers. Distributed wallets are not server-controlled and can be transferred from if the server(s) is compromised.

References

[1] Insurance Information Institute, “Auto Insurance. Costs and expenditures”, ”http://www.iii.org/fact-statistic/auto-insurance, 2014.

[2] Aaron D. Hall, “No-Fault Insurance in Minnesota”, http://thompsonhall.com/no-fault-insurance-in-minnesota/

[3] Insurance Information Institute, “Insurance Fraud”, http://www.iii.org/issue-update/insurance-fraud, 2015.

[4] Tony Baldock, “Insurance Fraud”, http://aic.gov.au/documents/6/0/E/%7B60ED996C-79A5-4D7A-AD75-FF073B7FF3BA%7Dti66.pdf , 1997.

[5] Andreas M. Antonopoulos, “Mastering Bitcoin”, https://www.bitcoinbook.info/, 2014.

[6] Insurance Research Council, “Insurance Fraud, A Public View, 2013 Edition”, http://www.insurance-research.org/sites/default/files/downloads/PressReleaseDraftMarch12.pdf, 2013.