CHAINPACT LITEPAPER

Q4/2022

CONTENTS

EXECUTIVE SUMMARY

In modern society, contracts are everywhere. They form the backbone of all business processes, be it a sales agreement, an employment contract, the terms of a product warranty or the simple implied contract of paying for a cab ride. Some of these are written down, some are customary and thus implied, and yet others are remnants of past oral promises.

The contract law in most jurisdictions requires parties to honour the terms of a concluded contract, and is aligned with certain doctrines and principles. These vary across geographies and jurisdictions but have many broad similarities. Some common elements of contracts include an offer, a consideration, acceptance within capacity and legality or enforceability of the terms, depending on their interpretation and awareness.

Although the backing of a court of law boosts trust and efficiencies of businesses, it only comes into active use in case of a breach. What ensues is a string of costly affairs, ridden with uncertainties and ambiguities. Most contracts, however, rely on simpler pathways of dispute resolution. Most people usually rely on mutual trust, standards and a game-theoretic way of dealing. That is, while a legal contract is a good to have, it's not usually the first mode of conduct in day to day transactions.

Codifying the terms and conditions of a contract into a smart contract (which is not technically the same thing) is a more efficient and secure manner of ensuring they are upheld. The power of decentralisation and security (thanks to cryptography) gives smart contracts on blockchains the ability to boost trust in a trustless setting. Essentially, smart contracts are just codes that everyone can see but not evade. While at first glance, it may seem a bit off-putting, we are essentially expecting code to honour contracts in all walks of life, given the digital transformation. The standard expectation that our digital platforms work without issues, and that the companies behind building them, do so in good faith, already stacks contracts following "code is law".

ChainPact brings together most of the benefits of written tangible contracts, for certain business objectives, and the power of smart contracts to let anyone make an agreement that cannot, technically, be breached. The idea is not to convert a spoken language into computer code, but to identify common business workflows, make smart contracts with easy to use interfaces and have a platform to connect them all. While this approach of creating opinionated contracts may not suit 100% of the use cases, our target is only to fulfil most of them, whence the behaviour of the contracts is well understood and simple.

CONTRACTS AND BUSINESS WORKFLOWS

Special Contracts are rare

Contracts for most use cases are very standard. A template for a common house sale agreement for a certain jurisdiction can be downloaded from the internet. It should contain most clauses, like who the parties are, the description of the house and price, the agreed upon warranty, quality, mode of delivery etc. This should be enough for almost everyone. The need for a completely different draft of the contract with very different conditions will be a special case. The template itself should be customizable enough to tailor any specific requirements too.

However, an extraordinary situation will demand some extraordinary contractual clauses, when the ones implied aren't enough. Such situations are rare, and can only be tackled with individual attention, by someone with good subject knowledge of all things to consider.

Legally enforceable contracts

A legal contract is understood to be one between two parties, that involves a promise or obligation to do or to transfer something, in exchange for something else. The contract law, while varying across jurisdictions, governs the rights and the obligations of contracts, even allowing parties to seek judicial remedies in case of damages.

A contract or its terms can be expressly written or implied. Implied terms come into existence due to regulations and fair expectations for common dealings. For instance, a patient may be expected to pay the doctor after examination is done - an implied contract.

A written contract comes into force when the parties have signed it, while being in capacity to do so, and not under undue influence or "duress". They should alsohave the correct understanding of the clauses given in the contracts. Performance of the obligations per the clauses should not require a law or public policy to be violated. Upon breach of a term, the court can award remedies to the affected party, depending on the nature and extent of losses.

Different types of remedies for breach of contract include "damages" which can be compensatory (for actual or projected losses), liquidated (agreed upon compensations as per clauses), nominal or exemplary. Further, consequential damages would provide for lost opportunity cost as a result of a breach, in addition to compensation, if all the parties were already aware of such possibilities.

For legal validity, all elements of the contract have to be in order. For instance, in some jurisdictions, a contract doesn't stand without a consideration (the price in exchange of promise). Further, while some jurisdictions can allow a simple exchange of words or conduct a valid contract, others may require specific procedures to be followed and agencies be involved for certain kinds of contract to come into effect.

Code is Law

With increasing reliance on digital systems: softwares and platforms, the fulfilment of key obligations depend directly or indirectly on the data and execution of these systems. For instance, it may be required that a garden must be covered if the weather forecast indicates expected rains. In this scenario the accuracy of the forecast is important, but not a necessary condition for the fulfilment of the obligation. In a way the forecasting system itself is directly influencing the performance of this obligation.

Similarly, if a plane's automated system indicates that a certain control must change for safety reasons, pilots may be required to let that happen, manually or automatically. This can be seen as a contract within the code of the automated system which is expected to behave in a certain way, based on its design. But this is not a real contract with a machine, but rather the functioning of it, having come as a result of many other business contracts on the way.

When we want automated software to function according to previously agreed terms or common knowledge of how they function, we are essentially relying on the parties making and controlling their execution. This can be stopped or altered by them, unless there is a contractual arrangement directly with them to not do so. This, however, isn't a problem, if the party using or being affected by the software has reasonable control over it. It may be noted here that such a discussion is only relevant for a software and not a hardware, as a hardware is not self-functioning through sophisticated conditions (without any software).

Reasonably controlled physical possession of a hardware's interface (steering wheel of a car, for example) is enough for it to be deemed to have no further contractual obligations to meet.

It turns out that properties of numbers (or maths and physics, in general) cannot be cheated, irrespective of how smart a software is, or who writes and controls it. Blockchains rely on this to secure the data and execution of smart contracts. Basically, a smart contract is just a piece of code floating around in the

blockchain on all connected nodes. It is executed on them, and the result itself is then tied to cryptographic numbers that are easy to verify but hard to alter.

The cryptography poses a techno-economic barrier against undue changes to the blockchain, and the decentralised, pseudonymous nature of it ensures that its access remains open. That has a big downside to it: vulnerabilities. An attacker could exploit a known or discovered vulnerability to unduly extract value out of a smart contract (profiting, in the process). Since, once a smart contract goes to the blockchain, it is immutable, there is no choice but to sustain the damage. One can't simply change something from the backend to directly fix it, although there are strategies to mitigate such situations with smart contracts.

Business with Blockchain

Smart contracts are arguably the most efficient and dependable solutions for business processes, even with the costs that come with it. But that is only if it is used in the correct manner. It is important to treat smart contracts as stores of core business logic alone. This should include critical operations like transfer of funds, managing balances and dues and storage of very crucial pieces of numerical data.

One may imagine that using a smart contract for a certain workflow is entering into a contract with other parties (represented through their wallets) where conditions and obligations are enforced through the immutability of the logic. It is always known in advance what may happen through the course of a smart contract's lifecycle. So, parties are aware beforehand what the implications of their actions are, and there can be (theoretically) no alternate outcomes.

One may conduct most auxiliary things, such as verbal communications, off-chain (not on the smart contracts) to save on gas fees. The smart contract can then be used to deal with critical components of fulfilment of the "promise", which can itself be coded in.

For instance, if a certain digital asset costs X, that asset can be released by the smart contract immediately after a payment of X to it, with no possible alternate outcomes. We may be too accustomed to such a transaction on our Web 2.0 applications such as e-commerce, but we must remember that such applications can be manually intervened and orders "cancelled" or retracted. Assets and objects in the real world will, of course, be needed to be mapped to the smart contract ecosystem before it can be utilised directly, giving rise to the concept of "tokens" or "Non-Fungible Tokens"

Open participation contracts

An open cause seeking widespread participation can start with a promise and have a flow of value, actions and/or incentives. As an example, consider an individual calling for a tree plantation drive at a remote place. This may require physical participation and some small cash collection for booking the resources (like transportation, tools etc.) needed for this operation. The contract (implied) is that the individual should direct any contribution to the said cause.

Such contracts, implied or express, can become difficult to manage, since it's hard to ensure parties act in line with the arrangement. It is also not easy to chase down the participants or the organiser, should the objectives or terms of the contracts fail. In most cases, recovering damages are never what people are after, but the fulfilment of the operation itself.

Blockchains, on the other hand, thrive in trustless settings. Rather than expecting parties to act in a certain way, the smart contract acts according to the code, if the parties choose to subscribe to that. Funds can be directed, points can be calculated and data integrity shall remain, no matter what.

INTRODUCING CHAINPACT

What does it do?

ChainPact lets anyone create an agreement on a blockchain, without any intermediaries. Unlike traditional agreements though, ChainPact depends on conditions coded into smart contracts for execution of actions and obligations. The entire pact revolves around fulfilment of conditions on the data about the pact stored in the smart contract. This makes pacts inflexible and direct, as against the ambiguous nature of more traditional verbal agreements.

Features

- Creating a pact with a third party that involves the workflow of a gig work or an employment contract
- Creating a blockchain-based DAO voting proposal
- Creating a fund-pooling or fund raising campaign based on blockchain
- Managing in-app transactions and data locally rather than server-side
- Support for mobile browsers and wallets

ABOUT BLOCKCHAINS

Blockchains are decentralised, distributed networks of data that guarantee some interesting properties, through cryptography. These are:

- **Immutable and tamper-evident:** the data that goes to the blockchain once, and is accepted by the network (which it is, if it is correct), it cannot be changed. By that we mean that a massive computational effort and cost would be required to make a calculation to produce a slightly different version of the blockchain, thanks to cryptographic hashes.
- **Access and ownership oriented:** Assets and data are owned by individuals through their cryptographic wallets. Making changes to them like modifying values, sending coins etc. can only be done by the owner. This is ensured with the use of digital signatures. Also the system, by design, cannot discriminate between wallet addresses, as against authoritative censors in traditional systems. A transaction will get through as long as it is mathematically correct.
- **No single point of failure:** Most blockchains rely on consensus of data between thousands of nodes, ensuring that a few rogue or unavailable nodes cannot bring down the entire network.

Every node on the blockchain has a copy of the whole database (the blockchain, or just the “chain”). All changes to this data is made through transactions, which are requests made to one of these nodes by an external device - a wallet or account. A transaction carries its data and the digital signature of the wallet that initiates it. If the wallet has access to the action this transaction is trying to carry out, it will be successful!

Wallets and Accounts

The blockchain works around public-key cryptography. An account is associated with an address - a mapping of the public key of a certain private (secret) key.

In simple terms, a (pseudo) random number is generated as a secret by an individual and stored in the user application called wallet. This secret is used to generate the address, which is used to refer to this wallet (associated with the secret). As long as the secret is not shared with anyone else, the account can only be operated by the wallet’s owner.

The wallet holder asserts identity using a digital signature: a cryptographic proof that a piece of data is being signed (or a transaction is being authorised) by the holder of the secret key associated with this account.

An individual can generate as many accounts as they wish, without any trace or association with their previously generated accounts. However, since transactions can be traced, if a new account is used to transact with a smart contract or another account, an association can be inferred. Hence, accounts are pseudonymous and not anonymous.

In the ethereum-like ecosystems, the contracts themselves are accounts too, with balances, but without any associated secret key. The wallet account only operates in accordance with smart contract logic.

About Smart contracts

Smart contracts are like blocks of code which reside on the blockchain and are executed by the nodes to change the state of the blockchain. This functionality in blockchain platforms like ethereum allows people to carry out more sophisticated data modifications and workflows, apart from mere payments alone (as in Bitcoin). The smart contract contains immutable code sent to the blockchain through a transaction and is assigned an address, similar to wallet address, but without any private key. The functionality in the smart contract can then be accessed by sending transactions to this smart contract address.

For instance, assume there is a smart contract code that has been set with the condition that only A and B can withdraw a certain amount of money locked in the smart contract. If C tries to withdraw it, the transaction will be rejected. The code for this is clearly visible to the public, just like all other data, and all nodes in the blockchain will run this transaction through the smart contract code. This eliminates the concerns around the secretive and controlled nature of the application logic in traditional web applications.

For instance, assume there is a smart contract code that has been set with the condition that only A and B can withdraw a certain amount of money locked in the smart contract. If C tries to withdraw it, the transaction will be rejected. The code for this is clearly visible to the public, just like all other data, and all nodes in the blockchain will run this transaction through the smart contract code. This eliminates the concerns around the secretive and controlled nature of the application logic in traditional web applications.

PACTS

In the ChainPact ecosystem, each unit of agreement (between one or more parties), represented through independent or shared smart contract logic, is called a pact. Thus, a pact is a collection of data related to a certain pre-agreed workflow, combined with pre-coded smart contract functionalities that act on the data, in accordance with the workflow.

In the realm of blockchains, especially in the popular smart contract oriented ones like ethereum, writing and storing data is a very expensive operation. There is an intentional high gas price associated with this, as replication and storage of data by every node in the network is an overall expensive event. Thus, it is wise to codify common business logic rather than writing them as clauses in a spoken language. Now, many users share the same core of logic (which codify clauses) but have differing data.

This should be done alongside a minimal amount of data recording. For instance, if the agreement is to get paid amount X on a date DT, then only X and DT should be recorded and the rest should be codified.

At this point, one way wonder the following:

- Can we convert plain human-readable language to code?
- Should we store a PDF document on the blockchain and use that instead?

An important distinction between a human taking an action, and a smart contract doing it, is automation. If a certain condition could be automated into a smart contract, it becomes a trusted way to execute it, without any other dependencies If X amount is meant for account A, it can only be withdrawn by account A (using their wallet), without needing anyone sitting at any counter verifying any other documents.

THE PACTS

It's easy to see that unless the circumstances are dramatically different, most use cases can rely on a standard set of codified clauses for standard agreements. The use of some parameterized conditions can give additional required flexibility to the pacts deployed as smart contracts.

Additionally, if someone tries to make a new smart contract every time, as a conversion from verbal language, it may contain errors or vulnerabilities. Smart contracts are unchangeable (fundamentally, but there are ways to make them upgradeable). And that means the vulnerabilities also remain alongside well-meaning code which can be exploited. Hence it is more convenient, wiser even, to have one smart contract code being shared, used, tested, fixed and distributed for a certain workflow. So, we create only a few varieties of pacts, opinionated and tested, to fit the needs of most people rather than everybody.

SIMPLE GIG PACT

The simple gig pact is meant to be used by freelancers and employers to create agreements for a gig or an employment setup.

Motivation

Freelancing often involves posting up gig titles on an aggregator platform, where customers would place orders and communicate their requirements. Once the gig is complete the payment is released to the freelancer, or any disputes sorted out by the platform moderators. While this is an okay model in terms of connecting the buyer and seller in the gig market, the payment, arbitration and the fees are anything but ideal. Following is a list of the concerns:

- High commissions: Platforms could charge anywhere between 10% and 30% out of the payment. At times, such commissions are levied on both the buyer and the seller
- Payment delays: The payment isn't cleared right away on these platforms. They tend to give a long window consisting of up to a few weeks before actually letting the payment go through
- Cross-border payments: Aside from integration of platforms like Paypal, a lot of the cross-border payments get stuck with services like wire transfer, or for various geo-political restrictions. Besides another set of charges and commissions related to currency exchange, often very indirect and meaningless ones, are imposed by the various actors in the chain of these payment clearances.
- Ambiguities of terms: Various components of the dispute resolution depend on manual processes, which follow the platform's terms and conditions which are often ambiguous. Moreover, the terms often favour one party based on strategies like "who brings more business", rather than a just a system
- Access to services can be restricted: A centralised platform can close their business any day, and we have seen this happen way more times than we would like to. Further, they could restrict an account, block it, cite nonsensical regulatory issues or not act according to their own terms or by-laws. There isn't much a retail user can do about that, as it is notoriously difficult to impose laws in far away jurisdictions.

Apart from the Gig market, the remote-work culture is on the rise in the post-2020 job market. It is expected that a lot of employees will be remote

workers, contracted with written agreements that may not actually be valid.

Broadly speaking following could be concerns of a remote worker or the employer:

- What if one party doesn't pay or doesn't hold their side of the bargain?
- How to enforce a PDF agreement onto a jurisdiction far away from their own?
- How to build trust, as a small company, to be able to have quality talent on board, given the uncertainties of remote-work situations?
- How to effectively manage multiple employees from a diverse set of countries, including compliances and payments?

Further, the case of crypto-payments alone isn't very favourable to centralised platforms, given the regulatory scenario. Even if a platform does enable fiat to crypto conversions, due to various technical and other issues, it stops working.

Creating a simple gig pact with ChainPact

In order to get started with the simple gig pact there are only 2 pre-requisites for both the parties

- A wallet - compatible with EVM like Metamask, there are many others as well
- Some balance for transactions - This would depend on the platform the gig pact needs to be deployed to. With L2 solution like Matic, it's much lesser than something like Ethereum mainnet

Either of the parties: referred to as "Employer" and "Employee" respectively could create the pact specifying following initial details in the Create Pact form:

- Employer wallet address
- Employee Wallet address
- Pay amount in native currency
- Pay interval in days

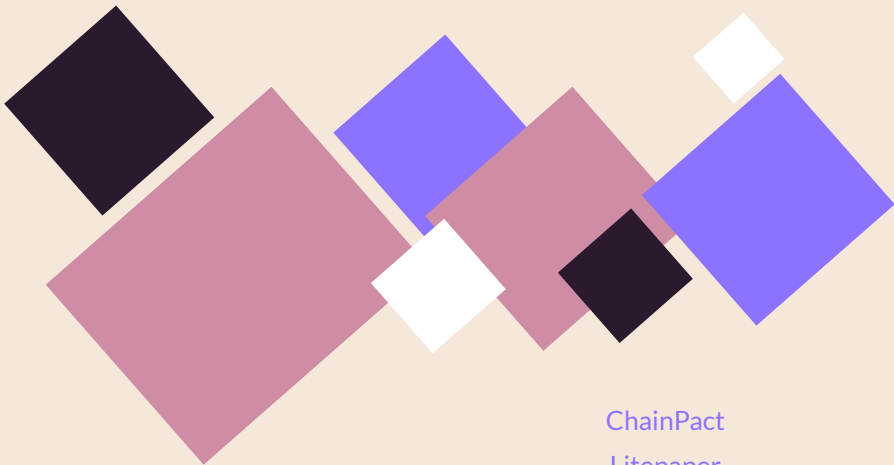
A value equivalent to the pay amount needs to be locked to the pact along with deploy transaction, this is to ensure payment security for the employee.
After this the pact is deployed as a separate smart contract on to the blockchain. And can be interacted with for further obligations or actions. The pact then needs to be "Signed" by both the employer and employee and "Started" for a timer on the pay interval to be started.

Actions on the Simple Gig Pact

Once the pact is deployed and signed, there are many actions to be taken, as a part of the general workflow, these actions include:

- Approve pay (by employer)
- Withdraw amount (by employee)
- Pause Pact
- Terminate or Resign
- Raise dispute, add and accept arbitrators
- Send Full and Final Settlement

In terms of locking the pay amount, the time the pact is “started” is considered a reference point from where the pro-rata pay amount is calculated. The locked amount minus the pro-rata pay amount is refunded to the employer on Terminate. And the remaining locked amount can only be withdrawn by the employer when the employee sends their Full and Final. This way, none of the parties benefit from the amount staying locked. And in case of disputes a third party wallet can mark the dispute as resolved if they are invited as an arbitrator by one of the parties and the other accepts it.



PROPOSAL PACT

A lot of today's social interaction involves a micro-blogging format, but without a secure flow for direct payments or fundraising. For instance, if I were to say "I will plant 1000 trees" on a micro-blog post, how do people directly support this cause from there? The purpose of a proposal pact is to be able to write simple words into a smart contract to form a pact.

A proposal pact represents a social contract embedded within a smart contract that enables a user to:

- Write an immutable text to the blockchain and lock some funds against it, like a valued promise
- Put a time lock on to funds against a proposal pact
- Let users pitch in more funds to the pact (that is to the cause represented by the text on the pact)
- Add participants who can get involved in the pact - they can pitch in, vote, disburse payments after voting etc.
- Conduct voting with set beneficiaries for its two outcomes - motion passes with most participants giving a YES vote, motion fails with most NO votes
- Discuss on the prospects and the proposal of the original pact with a connected social media thread (like twitter or Disqus)

The voting and participants part is kept optional, and someone could just put up a whole bunch of text without voting enabled. That, then, becomes an immutable text agreement or a proof of statement from a person (represented by an wallet account address)

How to make a proposal pact?

Go to proposal pacts click on "Create a proposal pact". The screen presents you with a form to enter the details of the pact.
For the most basic type of it, the only thing that is mandatory is the title of the pact. Enter the title and click on Deploy. Allow the transaction on the Metamask prompt and done!
After this a new Pact ID is assigned to the pact, which is a unique identifier used to refer to this pact both on the frontend and the smart contract.
In addition, following is the list of data that can be stored against a pact id, along with their significance:

- **Pact Description:** Stored in the smart contract as a string value, as an extension of the pact title. The subject matter of the proposal should go in this section, and any external links are previewed on the frontend
- **Initial value:** Some native value sent by the creator of the pact, as a contribution to the pact. This is also stored as such on to the smart contract
- **Lock Till (timestamp):** Till what date-time to lock all the funds of the pact. This signifies that funds stored against this pact id cannot be withdrawn till this date
- **Add participants:** Participants are wallet addresses with the authority to vote on the pact. Participants can be added even without enabling voting.
- **Enable voting:** To enable voting, one must add the beneficiaries for the 2 vote outcomes:
    - Yes Vote beneficiaries: The wallet addresses to send all funds to for YES votes > NO votes
    - Beneficiaries for NO vote: Wallet addresses or refund flag, if NO votes >= YES votes

Apart from this, the frontend for proposal pact also has Disqus comments and twitter threads enabled, but these things are not stored on the blockchain. The pact id and current contract address per chain is used as the identifier for these.

PROPOSAL PACT - USE CASES

Blockchain based crowdfunding

Crowdfunding is a great way to raise funds directly from the public for projects or ideas that a population segment is likely to be interested in. The challenges of crowdfunding that proposal pact address are the following:

- Single point of failure: A centralised third party is in charge of overlooking the administration of funds, a potentially dangerous situation. That is because for various financial, legal and operational reasons the funds may be withheld by them, after which an effective recovery may be near impossible.
- Red tape: Pre-processing, verification, access-restrictions are going to be there in most centralised crowdfunding platforms. They have to, after all, conduct their business. If I were to crowdfund for a cause incompatible with their business goals, I may not be allowed to participate. Or I may have to wait for a list of procedures to be conducted by them before I am allowed to raise any meaningful amount of money
- Cross border payments: A big challenge in any use case. If a crowdfunded campaign went for a million dollars, how do you effectively transfer this amount without having a large chunk of it eaten away in commissions.
- Platform commissions: Often more than 10%
- Crypto compatibility: For various regulatory or operational reasons, platforms shy away from direct crypto payments

How it works:

- Create a proposal pact with short summary of your proposal and external links
- Add potential investors/voters to the participant list
- (optional) Set minimum contribution for voting rights
- Start voting window and collect votes
- Use “Conclude Voting” option after the voting time is over

This mechanism can be extended to voting for all kinds of proposals including that of a DAO or casual organisations or groups.

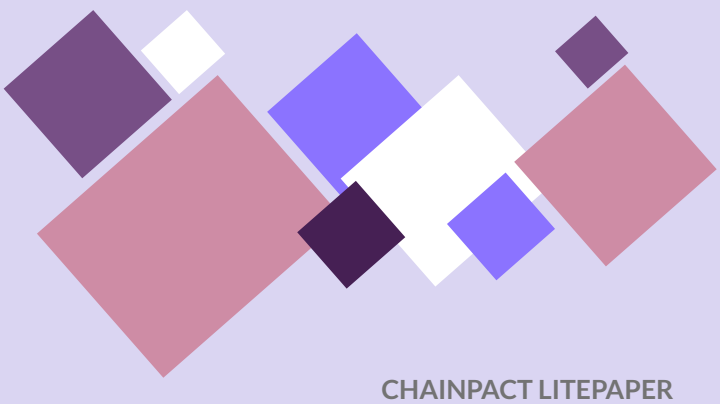
Fund pooling

A proposal pact can also be used to create a fund pool with an objective. Suppose a group of 5 individuals want to pool money to be given out to a party X for a

certain cause. One of the individuals could create a proposal pact, summarising the proposal, add the 5 individuals as participants and X as the YES beneficiary. Further the option “Refund for NO” could be selected, allowing the group to withdraw their contribution when the motion fails.

Casual Use cases

There are plenty of casual use cases for proposal pact as well. Thousands of posts are made on social media with objectives needing financial backing. Besides, if there is one place a post can’t be deleted and will stay there forever is the blockchain. If someone wished so, they could just make a plain word statement on to the blockchain, persisting it through the test of time.



GOVERNANCE AND SUSTENANCE

At the time of writing this, ChainPact was born out of the author's idea of a world with lesser barriers to efficient contracting. Such a thing will not sustain, nor be of value to a multitude of people without their direct participation. That's why ChainPact shall remain open to developer and shareholder participation throughout its journey.

Aside from that, we aim to enable a continuous feedback loop for progress and development of the application and ecosystem around it. This is also to capture the usual expectations and customs around business workflows to design better pacts to accommodate them.

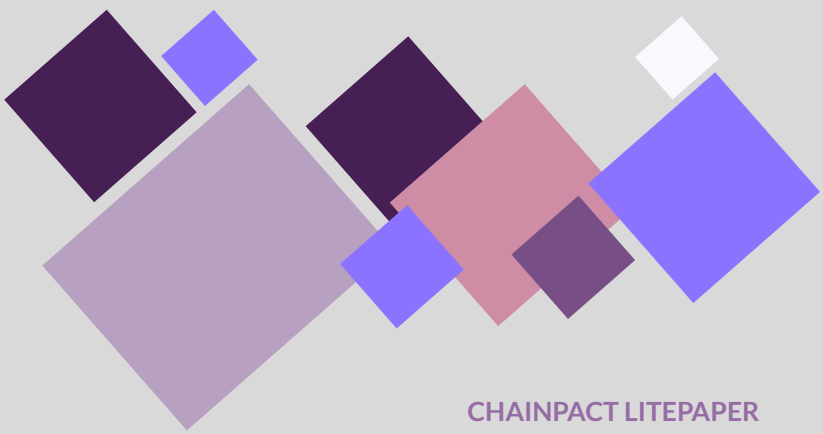
All of this requires a sustainable financing model and developer funding. There are a few ideas around that:

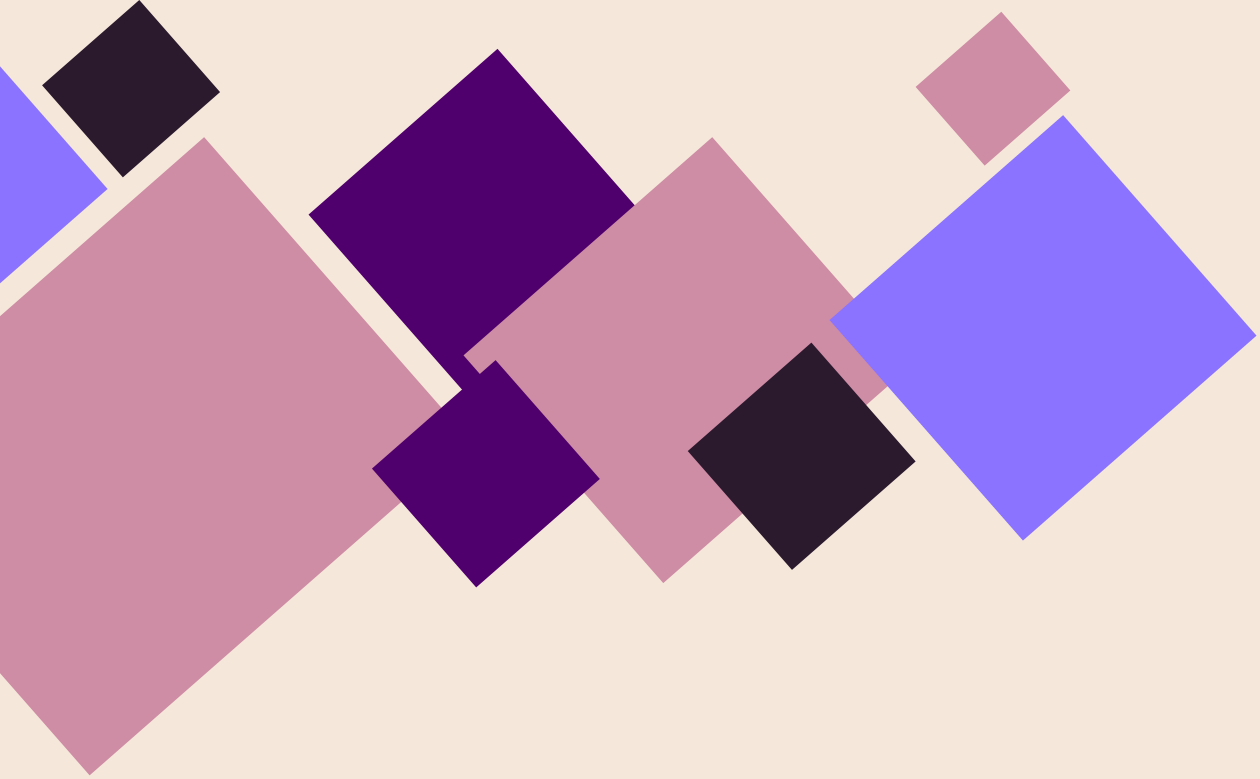
- Direct Charges - Costs recovered directly from the users in the following ways:
  - Transaction cost: Running a subnet or a new blockchain network for ChainPact can incentivise it through transaction costs on the network. A new subnet on Avalanche will also drastically reduce the transaction costs to the smart contracts themselves
  - Premium offerings: Certain premium on-chain offerings like reserving cool usernames or group names can further help bring in some cash
  - Small commissions: If the above methods fail, it may also be wise to keep a nominal direct commission on the pact amount
- Insurance: The purpose of insurance is to give remedies when disasters happen, so as to not let it hamper business as usual. ChainPact will introduce direct security insurance as a part of offerings, to channel the funds into smart contract security R&D
- Commercial collaborations
- Token sale

DAO (for future only)

The purpose of a DAO is to have a group of users working towards a common goal, governed by on-chain logic, complemented with discussions and voting. In future, ChainPact's DAO will start with the original team itself, before picking up investors and other interested individuals. The proposed allocation for the associated token will be roughly:

- 40% Developer allocation
- 20% Pre-Sale
- 20% Marketing and Promotions
- 20% Treasury





FUTURE

At the time of writing this paper ChainPact supports two pacts: one for a Gig, and one generic Proposal pact with voting. In the foreseeable future we will be writing pacts for more workflows, give users a better integration with traditional contracting methods like pdf signing, work with local law experts to explore the extent of hybrid legal contracts, and integrate points from the feedback received on these.

Apart from this we shall soon commence our efforts on providing enterprise tools for better managing pacts with features that enable more efficient workflow with a large number of pacts, notifications, integrations with other platforms, backups, infrastructure services around the pacts. This shall be in line with our vision to enable businesses to use assured, intuitive, and robust contracting flows.

All of this, of course, while navigating through the challenges of ensuring smart contract security, managing infrastructure and attracting open-source contributions.