Tonchain: the future of pensions?

Authors
Luuk van Benthem (l)
Loes Frehen (mr)
Joris Cramwinckel (ml)
Rik de Kort (r)[1]

Introduction

Many readers of this magazine will have heard of blockchain as an interesting technology with great potential. But what is its potential? Which financial services can be disrupted by using blockchain-based technology? In this article we will discuss an innovative experiment that focused on the application of blockchain for pension contracts.

In today's pension world, a pension fund invests participants' premiums, provides them benefit payments, and very importantly, *defines the way risks are shared*. Participation requires a certain minimal level of trust in the institution that offers the services (unless participation is mandatory). Blockchain has the potential to provide solutions to problems related to trust that may arise in the following situations:

- In many countries, trust in pension institutions is not at a satisfactory level. Diverse causes include opaque solidarity, unfair solidarity, lack of choice, or (fear of) fraud.
- Integrity within pension institutions is enforced via regulation. This has a price, because complex processes are in place to minimize the risk of fraud and human error.

In October 2017, a team consisting of members from APG, Ortec Finance, and PGGM initiated a project called Tonchain to answer the following question:

"Is it possible to build an autonomous pension contract with risk-sharing in the blockchain?"

- Autonomous: no intermediary or institution necessary to provide services.
- Pension contract: investment of premium payments resulting in a benefit payment from retirement age until death.
- Risk-sharing: participants share financial risks to improve (smoothe) the benefit payment.
- Blockchain: a technological tool to enhance autonomy by replacing trust in counterparties by trust in protocols.

This article will look first in more depth at the research question. Second, it will consider the dynamics of the Tonchain contracts. Next, the challenges (both technical and actuarial) that were encountered during the project will be discussed. The article will finish with an overview of how we will proceed.

The Question

Autonomy

With ongoing automation of processes in the financial sector, one could wonder whether at some point in time it will be possible to automate the entire pension chain. Especially for risk-sharing

pension contracts, this is not yet the case. Risk-sharing involves continuous decisions taken by the boards of pension funds. Decisions are translated into systems that require human validation. Our hypothesis is that by defining a pension contract with transparent risk-sharing and by using blockchain technology, autonomy could be made possible.

Is it possible to build an autonomous pension contract with risk sharing in the blockchain?

Pension contract

What is the definition of a pension contract? A pension is a way to ensure income after one retires, up until the moment of death. This can be done by saving and investing money throughout one's working life. There are numerous options to arrange the way invested capital is converted into a benefit payment. Is the benefit payment based on an agreed ambition where the premium payment and investment returns finance that ambition (Defined Benefit)? Or is the benefit payment based on the agreed premium payments that are invested (Defined Contribution)? Are there buffers to smooth benefit payments over time?

While these are interesting questions, transparency is a typical characteristic of blockchain, whereas sharing risk via a buffer in DB contracts is not very transparent. Additionally, simple dynamics are preferred because of technological limitations. These considerations led the team to develop an individual lifetime spanning DC contract without buffers. Because it's lifetime spanning, it allows participants to take investment risk during the retirement phase.

Risk sharing

Pension contracts naturally carry risks due to the long horizon they are exposed to. Taking investment risk is necessary to provide inflation compensated benefit payments. In the Tonchain contract, investment risk is borne by the individual. However, the risk of the investments is matched with the risk appetite of the participant. Interest rate risk is also borne by the individual, but reduced by means of lifecycle investment patterns.

When pensions are considered as just an investment account, it may be the case that the participants' lifespan will be longer than expected and that the income may run out, which can lead to a very poor standard of living at an advanced age. Risks associated with mortality are typically suited to be mitigated by sharing them amongst participants, using the law of large numbers.

The classic implementation of this risk sharing is a solidarity group: the participants who get the short end of the stick are supported by those who get the long end. That is, those participants who do not consume all their available income during their retirement (due to dying), support those who do consume all, or too much of it (due to longevity). The Tonchain contract implements this type of risk-sharing through the Tontine principle. Hence the name.

Blockchain

The project aimed to develop a smart pension contract on a public blockchain. A public blockchain provides maximal transparency to the participants, and a smart contract is a piece of computer code that is hosted and executed by such a blockchain. Ethereum was chosen for this project as the platform to develop on, in the Solidity programming language.

Nice-to-haves

Additional specifications were that the contract should be suitable for voluntary subscription (no assumptions about obligations to participate), provide the participant with a set of choices (e.g. investment risk to take, pensionable age). The contract should provide risk sharing among a heterogeneous group with respect to life expectancy. Because the focus was on investigating the pension contract of the future, there was no effort made to comply with current (Dutch) regulations.

Tontine Principle

History

First devised in the 17th century by a banker named Lorenzo de Tonti, the Tontine principle has been applied in a retirement context for years. It works as follows:

- During their working life, participants deposit money into a Tontine account. This account remains their own, but is part of the Tontine pool.
- After retirement, participants withdraw funds from their account. Here, the money can still be invested.
- Once a participant dies, ownership of their account is not transferred to their heirs, but instead the remaining account balance is divided among the other participants in the Tontine pool. In this way, those who live longer than expected are subsidized by those who don't.

The Tontine principle provides a flexible way to share risk. The plan is very individualized, with money having a clear owner during the lifespan of a participant. There is room for investing funds during both the working and retirement phases, with the participant "calling the shots" on those decisions.

Dynamics in Tonchain

The Tontine principle is implemented in Tonchain as follows: every participant has a separate account into which they pay premiums and receive investment returns. Starting at retirement age, every year a monthly benefit payment is determined, by dividing the available capital by a participant-specific annuity factor. This annuity factor depends on life expectancy and current interest rates. When a participant deceases, their remaining account balance is distributed between the surviving participants. The distribution key depends on survival probabilities and current account balance.

In contrast to the original Tontine, the pool is not closed but allows new participants to enter at any time.

The remaining funds of a deceased participant are divided amongst those who remain, based on their capital and mortality rate

Making the Tontine fair

There is a challenge to tackle when different members of the group (even at the same age) have different life expectancies: how to make mortality- and longevity risk sharing with the Tontine principle fair? That is, how is it ensured that no one has an ex ante advantage? Indeed, if anyone *does* have an ex ante advantage, there also has to be someone with an ex ante disadvantage. Since there are no assumptions related to an obligation to participate, the ones with an ex ante disadvantage are simply not going to join the pool (provided they are aware of the disadvantage).

A second point is the question of security. The underlying goal of a pension is to provide a steady retirement income. How can it be ensured that participants will have a steady income during their lifespan?

The Tontine principle is old, and in recent years has received a significant amount of attention in the literature. As such, the question of fairness has been given some thought; it boils down to finding a solution to a set of non-linear equations involving the amount each participant has on their account and their survival probabilities. In a pool of 25,000 participants, there are 25,000 equations with 25,000 unknowns. Sabin (2011) devised an algorithm that produces a satisfactory solution. However, as we will describe later, current block-chain technology is not feasible for implementing this algorithm. Instead, an approximation method is used, which will be discussed later.

Comparison

Comparing our implementation of the Tontine principle to common pension schemes, the former has favourable characteristics.

In a defined benefit (DB) scheme, the participants pay premiums into a central pool and obtain benefit payments from that central pool. The amounts are determined by the institution operating the pool. The institution also determines how the money is invested. In short, the participants share all kinds of risks (investment, longevity, and mortality) with very little control or transparency.

In a classic defined contribution (DC) scheme, the participants pay premiums which are deposited into their own account, and use this to buy an annuity for benefit payments. Here the participant has more control over the investment, but in return doesn't share the investment risk with other participants. Buying an annuity again is a way to share the longevity and mortality risks. However, there is a significant amount of interest rate risk: if the interest rate is low, the annuity payments will also be low. Since classic DC schemes force a participant to buy an annuity at retirement age, interest rate risk is strongly concentrated at this moment. Furthermore, the participant cannot invest after retirement and therefore is expected to have lower benefit payments.

Using the Tontine principle, there are individual accounts, where participants share mortality risk peer-to-peer. *Mortality risk* is uncertainty about the moment of death. Additionally, because there is no single buy-in moment, the interest rate risk is not concentrated. Finally, there is the possibility of investing capital during retirement, leading to higher expected benefit payments.

Technological Challenges

The innovations that blockchain presents provided the impetus for starting this project. Hence the natural technological environment for Tonchain was the blockchain.

Out of the many different platforms, the most stable, mature, and well-tested is Ethereum. Though still in its infancy, Ethereum offers an almost-complete programming language (it has all the features, but there are hard caps on memory and computation time). Being the first fully programmable blockchain, Ethereum has by far the largest development community. The result of this is a wide range of tools and tutorials that make it easier to develop for Ethereum. Being new to blockchain technology, the team gladly made use of these.

Security versus ease-of-use

The most striking feature of development for block-chain is the tension between secure programming and "easy" programming.

For example, in order to execute a benefit payment, one does not send money to a participant directly. Rather, the money is made available for the participant to pick up. The reason for this is that blindly sending money can lead to program execution, potentially triggering unintended transactions, whereas making money available does not. A second example is the limited computation resources available to a "smart contract" (program on the blockchain). There is a cost associated with code execution, called "gas", and a hard cap on the amount of gas one is allowed to use in a calculation.

Another issue encountered is that Tonchain needs to do some actuarial calculations on the blockchain, but doing this became difficult once it became apparent that Solidity (currently the only real programming language for Ethereum) only supports integers. The reason for this is consistency: once numbers in a general sense (floats and doubles) are allowed, all kinds of error possibilities involved with rounding are introduced. This leads to different network participants getting different results from the same computations, which results in stagnation of the network because participants don't agree on a state.

This issue affected the project directly, because the algorithm for fair mortality gains in the Tontine contract requires numbers generally, not just integers (it contains a square root). While workarounds are possible, the cost of computation and hard caps made this highly infeasible. This meant the team had to go back to the actuarial drawing board.

Risk Premium Method

To share the remaining capital of deceased participants in a fair way, a method based on mortality risk premia was used. The remaining funds of a deceased participant are divided amongst those who survive. For a surviving participant this is based on his mortality rate relative to other participants and their capital relative to total capital. The higher their capital and the lower their survival probability (both compared to other surviving participants) the higher their share in the capital of the deceased participants

The risk premium method is not a completely fair method like the algorithm of Sabin (2011) but it is accurate except for a few rare cases. For example when account balances between participants differ to an extreme extent and the total number of participants is limited. The distribution key in the risk premium method is easily computable from account balances and survival probabilities, without having to have full support for decimal numbers.

Public versus private

Even with the risk premium algorithm, the limits set by the Ethereum network were still quite stifling. In a development environment, developers work on a so-called "private" blockchain. This is a blockchain to which only a few participants have access. It allows the operator(s) to fully control all aspects of the blockchain, including the hard limits normally set by the network. Integrity can still be guaranteed by having the different parties on the network independently verify transactions. Using a private chain, the team was able to create a fully functional Tonchain contract.

The question is not: "how does one prevent errors?", but rather "what does one do in the event of error?"

It seems obvious: simply use a private blockchain in combination with the risk premium algorithm to launch a Tonchain service. But blockchain is all about trust: on a public chain participants know knows their transactions will be carried out as intended, because the people operating the network (a subset of the participants known as miners) are incentivized to do so. Since there are many miners, it becomes very hard for one miner or group of miners to disrupt the network. If one is to participate in a private chain network, one has to trust the few operators of the network to do their job correctly. However, lack of this kind of trust was one of the principal reasons for starting this project. Additionally, we expect public blockchain technology to develop rapidly, and we think that in a couple of years it will be mature enough to host a fully autonomous pension contract.

Immutability

Speaking of trust, now is a good time to talk about immutability. Once program code is deployed to the blockchain, it is there forever. This is what generates the trust in smart contracts: once they're up, no one can change them. But that leads to another issue: what if the smart contract contains an error?

Clearly, with something as important as pensions, errors are disastrous. So one checks, tests, audits, rechecks, and generally does everything in ones power to minimize the probability of error. But minimizing is all one can do: it is simply impossible to eliminate all errors. Examples from mature technologies abound: recently, we've seen vulnerabilities in the standard WiFi-communication protocol (WPA2 KRACK), and in virtually all computer processor units (Meltdown and Spectre). So the question is not: "how does one prevent errors?", but rather "what does one do in the event of an error?"

This means that a blockchain program needs failsafes and backdoors. Common mechanisms include "mother contracts", which point towards the most current program (providing update functionality). Another is dead man's switches, which terminate the contract in an orderly manner if the owner does not regularly give signs of life to keep the contract running. And now the earlier issue re-arises, in a public setting: who is trusted to maintain these failsafes and backdoors? The answer to that question (at least for now) is: we don't know. One interesting idea is having a contract that is not just open-source for auditing, but that is developed open-source as well, i.e. anyone can provide a new version of the contract, and the solidarity group decides by using a voting mechanism if and how they want to move to the new contract.

Where to go from here?

During the three months of the experiment, the team developed a working demo on a private Ethereum-based blockchain. Even in this private environment, the contract handles just a few thousand participants at a time; enough for a stable Tontine pool, but not sufficient to cover large populations. Furthermore, because the focus was on risk-sharing, assumptions were made about to other relevant services. For example:

- Asset management services on blockchain are available. In reality, propositions are currently under development and don't match the quality of non-blockchain propositions. A classical asset management service is not preferred good solution, because this results in a trust leak where capital is handed over without agreements being arranged in public smart contracts, making them unnecessarily opaque to the participant.
- Identity confirmation is possible. In reality, on public blockchain platforms there is not yet enough information available to validate a person's identity and get updates on life events such as death and marriage.

This leaves the following questions unanswered:

- How does the contract ensure participants are really who they say they are?
- How does the contract obtain data about who has died?
- How can one make an autonomous contract sustainable with respect to unexpected developments in the future?
- How does the contract or the blockchain guarantee the participants' money is being invested according to their directions?
- What do regulators think about this kind of pension plan? How is the contract made compliant with rules and regulations?
- Where do the survival probabilities come from? Who decides those?
- What interest rate is used for the benefit payments?
- What is the procedure when participants want to leave the product? What penalty do they have to pay?

To conclude, how do we answer the question that our experiment began with?

> *"Is it possible to build an autonomous pension contract with risk sharing in the blockchain?"*

The experiment showed that a pension contract with risk-sharing can potentially be hosted on a public blockchain. Even though autonomy of the contract is attractive due to potential for cost reduction and a reduced need for trust, autonomy is also tricky due to things like immutability. After all, a pension platform should remain available far into the future, and the world might change significantly during its lifespan.

Currently, available public blockchain platforms have many disadvantages, and blockchain services need to be further developed before they can be used for pension solutions. More thought needs to be given to the maintenance problem: how can we deal with technical updates while maintaining the integrity of the pension scheme, and how to ensure that the platform can evolve with technology over time, while still relying on a blockchain protocol for immutability?

Possibly, if blockchain technology becomes mature enough, pension provision might become available in areas of the world that currently do not enjoy a robust institutionalized pension system. Thinking further, will developed countries still need institutionalized pension systems in the future? Or will it be more efficient and transparent to provide autonomous pension services on a blockchain platform? The team is currently investigating questions that emerged in the first phase of the experiment. The road to travel from here is still long, but the team feels ready to tackle these challenges head-on. ■

Literature

_ Sabin, M., 2011, A Fast Bipartite Algorithm for Fair Tontines. Available at SSRN: https://ssrn.com/abstract=1848737 or http://dx.doi.org/10.2139/ssrn.1848737

Notes

1 Luuk van Benthem MSc AAG, PGGM, ALM-advisor, Loes Frehen MSc AAG, APG, Actuary, Joris Cramwinckel MSc, Ortec Finance, Technologist en Rik de Kort MSc, PGGM, Data Scientist.

The authors were part of a larger team, and would like to thank the team for helping to make this article possible.