Ensuro Protocol White Paper

V0.0 (Draft)

April 2021

Abstract

This document describes the definitions and mechanisms behind the Ensuro protocol.

Contents

1 Introduction

Insurance is a massive US5T market [1] that has exceptionally high entry barriers both on the regulatory and capital side. Only a handful of big players compete in this market, creating a de facto oligopoly [2].

The legacy model presents several advantages, such as standardization of standard policies and processes, tight solvency regulations, and economy of scale. However, the advantages benefit only the top of the value chain, at the disadvantage of policyholders, brokers, and the rest of the insurance ecosystem. Insurees face inflated premiums to bear the costs of overheads, capital, and compliance and suffer opacities and delays in claim assessments and payouts in a contentious environment in which transaction costs take up about 40 % of insurance revenues, on average [3]. The lack of competition has fostered and crystalized inefficiencies across the entire value chain, slowing down the innovation pace [4], trimming down segments for brokers, and reinforcing social inequalities. The structure of the insurance ecosystem made it inconvenient for insurances to operate in a space with a low margin and a high number of users. Consequently, established insurance companies often decide not to work in developing countries, forcing people to rely on unregulated and under-collateralized insurers [5]. Moreover, the insurance market proves to be minimally accessible as an investment product: the average individual cannot access insurance risk to diversify their investment portfolios [6].

Ensuro wants to solve these problems. Ensuro's vision is to connect liquidity providers and risk model providers via a pool-based strategy.

Figure 1: The Ensuro Protocol

Ensuro is a decentralized underwriter to support insurance products. It will democratize the possibilities and the benefits of being an insurer while allowing innovative companies to nurture and deploy novel insurance products.

In this white paper, we define the mechanism behind the Ensuro protocol. We first provide a brief introduction of insurance products and an overview of the structure of the protocol. In section 2, we define and explain the protocol's variables. In section 3, we outline the protocol architecture. In section 4, we describe the contracts. Finally, in section 5, we elaborate on the functioning of the protocol, explain its tokenization, and the distribution of rewards and losses among the LPs. In section 6, we conclude.

1.1 Basic concepts

Insurance principles are well-defined and straightforward. In the following, we present a pedagogical example that explains the principles of insurance, the actors involved, and the dynamics of the insurance value chain.

Insurance is related to accidents that generate losses. It follows that insurance addresses the probability and the severity of these losses, and the protection of the insured against such risks.

Our pedagogical example considers flight cancellation insurance. Recently, Alice bought a ticket for 1.000$ and would like to ensure herself the ability to buy a new ticket in the case of a sudden cancellation of the flight. Alice decides to purchase a loan of 1.000$. For that, she has to pay redemption (also called the principal). Additionally, she pays an interest rate of maybe 1%, so she has yearly costs of 20$ (1.000$ loan × 1% interest rate + 10$ annual redemption = 20$).

Alice is skeptical about this solution and reaches out to her friends Bob, Claire, and other 9997 acquaintances to pool together and share the risk of sudden cancellations. We assume that they purchase tickets from, and to, different places, spending 1000$ on their flight tickets. A team of statisticians estimates that these independent flights have the same probability of cancellation of 0.1%.

Assume that each member of Alice's community pools together and pays a 1$ share; this amount is now called the *premium*. 10.000 members collect a total of 10.000$ in premiums. Based on the information given by the statisticians, the expected losses across the group of friends will be 10.000 tickets × 1000$ × 0.1% = 10.000$. Because the sum of all premiums is also $ 10.000, the whole loss can be paid out of the collected premiums without taking a loan; the individual cost for each of Alice's friends is now reduced from 20$ to 1$.

As in the previous story, insurance companies pool several (say, n) similar risks. These individual claims can be thought of as a sequence of random variables $x_1, x_2, \ldots, x_n$, each causing a loss $p_{\to c}$ (the *payout*) with a probability λ_p. The law of large numbers assures that, as the number of claims increases[1], the total loss caused by $x_1, \ldots, x_n$ will converge to $p_{\to c}^{tot} = n \times \lambda_p \times p^{\to c}$; it follows that to break even, at least on average, the insurance company should charge each customer an amount $p_0 = \lambda_p \times p_{\to c}$, called the *pure premium*. In practice, the premium paid by the customers is always greater than the pure premium, as it covers both operational costs and a profit margin.

When insurance companies sell their products, they need to lock some additional backup capital. The function of this backup capital is to cover *unexpected losses*: the insurance needs to be solvent even if a few flights more than anticipated are canceled. We refer to this

[1] If the claims are independent from one another.

backup capital as the *Solvency Capital Requirement*. The SCR is the capital required to ensure that the insurance company will meet its obligations over the next 12 months with at least a given probability - 99.5% for EIOPA Solvency II.

Ensuro democratizes the possibility to participate in the SCR. Liquidity providers can deploy their funds within the protocol, which algorithmically allocates it to backup policies. In this way, everyone can participate in the risk and the rewards of the insurance business.

1.2 Structure

The Ensuro protocol will be composed of three main components:

1. **Liquidity pools**: Ensuro creates dynamic pools governed by a smart contract on the blockchain (from now on, *liquidity pools*) where Liquidity Providers[2] (LPs) can deposit money under the form of stable coins (e.g., USDT, USDC, cUSD), defining their acceptable cashback period, i.e., the amount of time they are willing to wait before receiving the money back once they have requested it[3]. The presence of well-defined cashback periods permits the protocol to plan ahead and compute the amount of capital available at any point in the future. With this forward-focused vision, Ensuro can always operate at the desired solvency level. Insurance is a profitable business. Being the heart and the fuel of the protocol, liquidity providers will share the profits from the pool, depending on their deposit's size and the length of their cashback period[4].

2. **Risk modules**: Ensuro's partners will have access to the protocol's liquidity through our Risk Modules. Each risk module will be connected to a specific insurance product. To protect the Liquidity Providers, the Ensuro Risk Monitoring Module will perform proper backtesting of potential risk modules and Quality Assurance Tests (QA) on the outstanding ones to check that their performance is in line with expectations, and correct possible deviations.

3. **Asset manager**: Ensuro's asset manager component will rule the interaction of Ensuro's reserves with other DeFi protocols. Specifically, the asset manager will deploy a part of or the total fund in other risk-free protocols (e.g., Aave) to provide additional returns to the liquidity providers. We will give further details on the Asset manager in the following versions of this document.

1.3 Governance

A governance token will control the protocol. More details on this token and on the Governance will be released in the following versions of this document.

[2]We use *liquidity providers* and *users* as synonyms.
[3]Each of the pools will have a distinctive cashback period
[4]Intuitively, the protocol will favor liquidity providers that will lend larger sums for longer periods.

2 Variables definition

The following table defines the variables involved in the protocol.

Notation	Description		
	Event.	With *event*, we indicate every act or action that modifies the state of the protocol. Events fall under two categories: • *policy-related*: sale of a new policy, expiration of an existing policy, claim of an existing policy, rebalancing of the capital that backs up the policy among the pools, • *LP-related*: capital is deposited in or withdrawn from the protocol.	
t	**Current timestamp**.	Current number of seconds.	
t_{-1}	**Last update timestamp**.	t_{-1} is updated every time an event occurs.	
Δt	**Time delta**		$\Delta t = t - t_{-1}$
t_{year}	**Number of seconds in a year**		$t_{year} = 31536000$
Δt_{year}	**Time delta as a year percentage**		$\Delta t_{year} = \frac{\Delta t}{t_{year}}$
l_p^π	**Locked part of pool π to back up policy p**		
$t_{e,p}$	**Expiration time of policy**.		
λ_p	**Loss probability**.	Probability that the customer will claim the policy p	
$p_{\to c}$	**Payout of policy**	Amount due to the customer if p is claimed	
$p_{\leftarrow c}$	**Premium**	Amount paid by the customer for policy p	
p_0	**Pure premium**	Pure premium of policy p	$p_0 = \lambda_p \times p_{\to c}$
$p_{\to}$	**Profit**		$p_{\to} = p_{\leftarrow c} - p_0$
$p_{\to l}$	**Share of profit to LPs**		
$p_{\to rm}$	**Share of profit to risk module**		
$p_{\to e}$	**Share of profit to ensuro**		

$p_{\%scr}$	**SCR percentage**		
p_{scr}	**Solvency Capital Requirement**	Solvency Capital Requirement (SCR) for policy p	$p_{scr} = p_{\%scr}\,(p_{\to c} - p_{\leftarrow c})$
r_p	**SCR interest rate of policy** p		$r_p = p_{\to l} \times \left(\frac{t_{e,p}-t}{t_{year}}\right) \times \frac{1}{p_{scr}}$
l^π	**Locked part of pool** π		$l^\pi = \sum_p l_p^\pi$; $p_{scr} = \sum_\pi l_p^\pi$
o^π	**Liquid part of pool** π		
s^π	**Total amount of tokens in pool** π		$s^\pi = o^\pi + l^\pi$
$r^{\pi,m}$	**SCR interest rate in pool** π	Weighted average of the policies' SCR interest rates	$r^{\pi,m} = \frac{1}{l^\pi} \sum l_p^\pi r_p$
r^π	**Interest rate of pool** π		$r^\pi = r^{\pi,m} \times \frac{l^\pi}{s^\pi}$
I_t^π	**Scale factor**	Ongoing interest cumulated by pool π. It gets updated whenever an event occurs.	$I_t^\pi = (r^\pi \Delta t_{year} + 1)\, I_{t-1}^\pi$
$b_{x,t}^\pi$	**Scaled balance.**	Is the share of user x in the ERC20 π-eToken pool at time t. It gets updated on every event (see 5.2). We call $b_{x,t-1}^\pi$ its last updated value.	
$e_{x,t}^\pi$	**π-eToken balance.**	π-eToken balance of user x at time t.	$e_{x,t}^\pi = b_{x,t-1}^\pi \times I_t^\pi$

Table 1: Notation and terminology.

3 Protocol Architecture

We outline the protocol's architecture below .

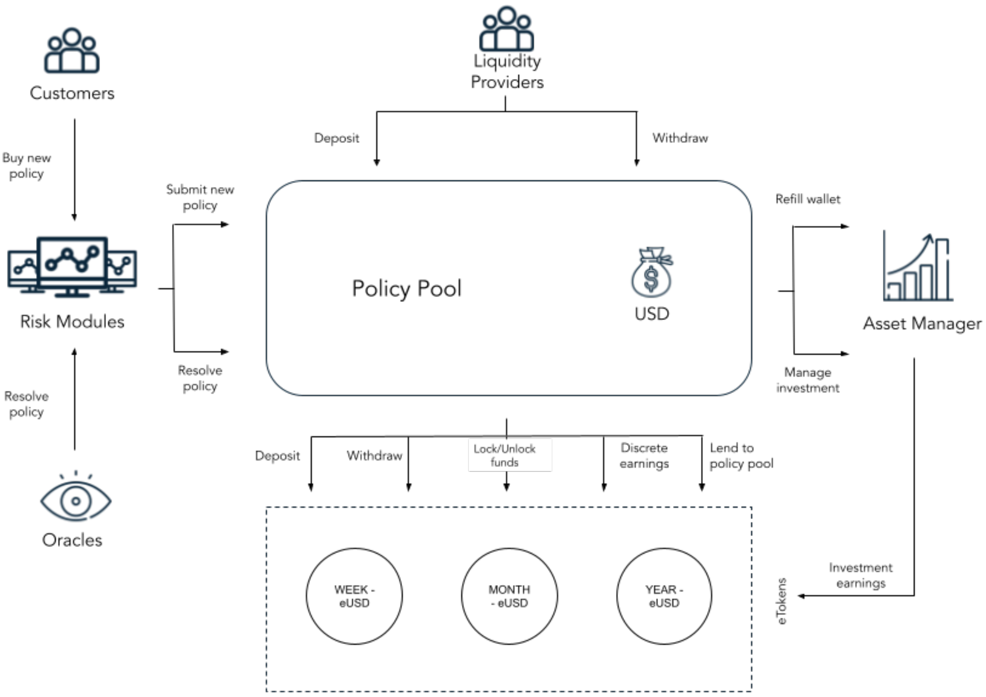

Figure 2: Ensuro's structure.

3.1 Policy Pool

The `Policy Pool` contract is the heart of the protocol. It manages Ensuro's liquidity reserves interacting with the risk model providers (by accepting new policies and locking or releasing funds) and with the liquidity providers, who can deposit and withdraw money from the different liquidity pools. Crucially, it takes care of the tokenization (see sec. 5.2) and of the computation and compounding of interests. We review in sec. 4.1 its most important actions, `Deposit` and `Withdraw`.

3.2 Risk Module

The `Risk Module` contracts allow Ensuro's partners (insurance companies and risk model providers) to interact with `Policy Pool`. Each `Risk Module` can propose new policies to the liquidity pools and takes care of the resolution of outstanding policies. A handful of parameters characterize each policy: its *payout*, *probability of default*, and *expiration date*. From these parameters, it is possible to compute each policy's capital requirements. Thanks to the flexibility of Ensuro's contract, the capital needed to back up the policy can be fully provided by Ensuro's reserves or jointly by Ensuro and the risk module developer. In case of joint coverage, the partner's share is transferred to and locked into the Ensuro's pool and returned to the partner at the policy's expiration[5]. We detail `Risk Module`'s most important actions, `New Policy` and `Resolve Policy`, in sec. 4.2.

The funds locked in a pool are divided into different *buckets* that are accessed (when the protocol needs to reimburse a customer) or replenished (when a premium is collected) in a well-defined order. We provide details of this process in sec. 4.2.3.

3.3 Asset Manager

The `Asset Manager` contract manages the reserves of the protocol. It is responsible, e.g., for reinvesting Ensuro assets in other risk-free DeFi protocols to provide additional returns to the liquidity providers.

[5]The risk module developer becomes effectively a liquidity provider for its own policies

4 Actions

We list the main actions of Ensuro's contracts below.

4.1 `Policy Pool`'s contracts

4.1.1 Deposit

The `Deposit` action allows users to deploy capital in one of the liquidity pools.

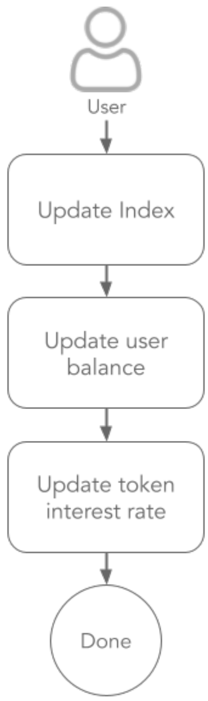

Figure 3: Deposit funds

4.1.2 Withdraw

The `Withdraw` action allows a user to exchange a desired amount of π-eTokens for the corresponding amount of stablecoins in the pool.

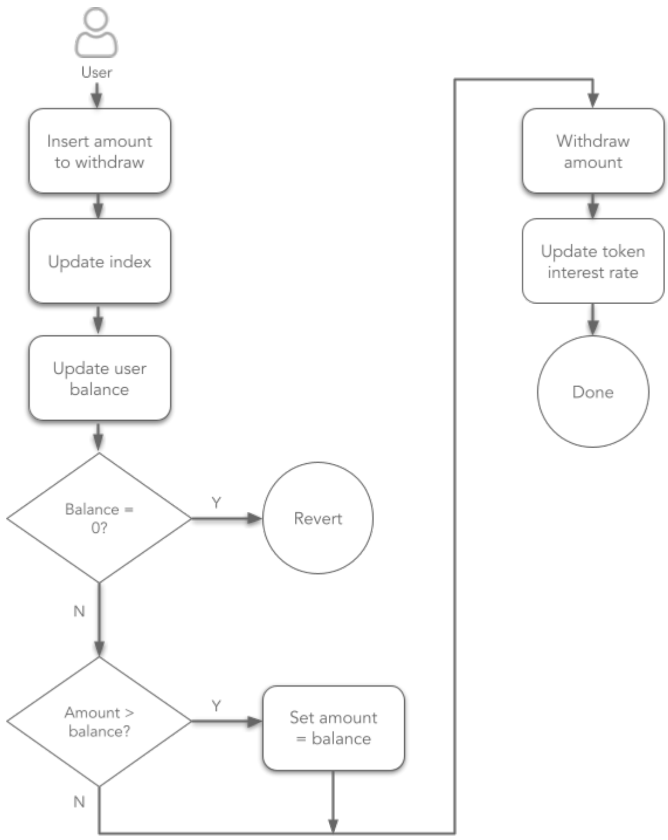

Figure 4: Withdraw funds.

4.2 `Risk Module`'s contracts

4.2.1 New policy

The `New Policy` action is called whenever one of Ensuro's partners sends a request to back up a policy. If the pool contains enough liquidity to meet the *Solvency Capital Requirements*, the new policy is activated, together with an ERC721-compatible *non-fungible token* (NFT). The NFT is transferred to and owned by the customer who acquired the policy. When a policy expires, in the case of a payout, the reimbursement is sent to the owner of the token.

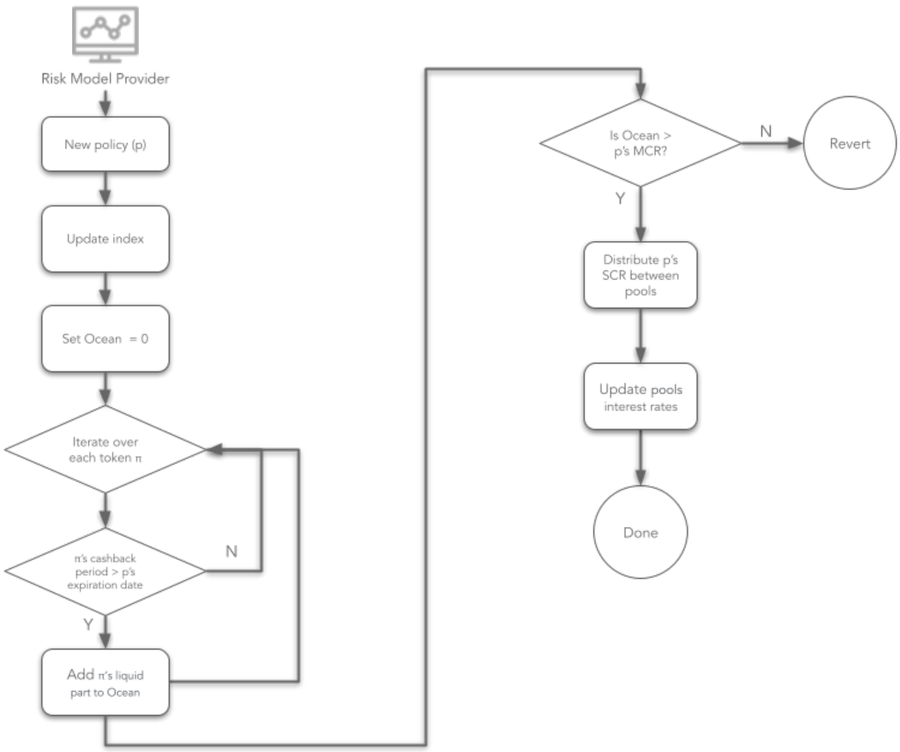

Figure 5: A new policy is sold by the risk model provider. If there is enough liquidity in the pool, SCR funds are locked.

4.2.2 Resolve policy

The `Resolve Policy` action is called whenever a policy expires. An external oracle checks the outcome of the policy and triggers the reimbursement of the customer.

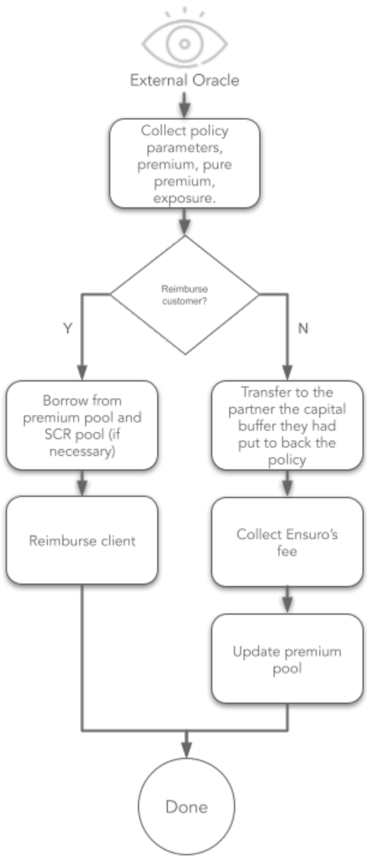

Figure 6: External oracles trigger the resolution of a policy.

4.2.3 Payment Structure

When a policy expires, the `Resolve Policy` action either provides the payout (in case of a claimed policy) or allocates the pure premium. Both actions access three different *liquidity buckets*, shared across the Policy Pool, in an ordered manner. The first one, the *collected pure premiums* bucket, is composed of the pure premiums inherited from past policies[6]. The second one is the *active pure premiums* buckets, composed of the pure premiums of the outstanding policies. The third one, called the eToken loan bucket, is composed of the capital coming from the liquidity pool.

When providing a payout, the `Resolve Policy` action first empties the *collected pure premiums* bucket, then the *active pure premiums* bucket, and finally, the *eToken loan bucket*. When collecting the pure premium of an expired policy, the `Resolve Policy` action first refills the *eToken loan* bucket up to the maximum level (nothing owed to eTokens), then refills the *active pure premiums* bucket (up to the sum of pure premiums of all active policies) and finally refills the *collected pure premiums* bucket (no maximum level). This mechanism was designed to shield liquidity providers from unnecessary volatility. We know that, statistically, the *collected pure premium* bucket will be empty (the loss coming from a product is on average equal to its pure premium), and the *active pure premium* bucket will be consumed by the outstanding policies (for the same reason). We then do not distribute these two buckets to the pool, but rather use them as a buffer for the expected losses. In case of unexpected losses, we tap into the pool's reserves. This can be thought of as a loan from the liquidity pool to the protocol and is later paid back with interests[7].

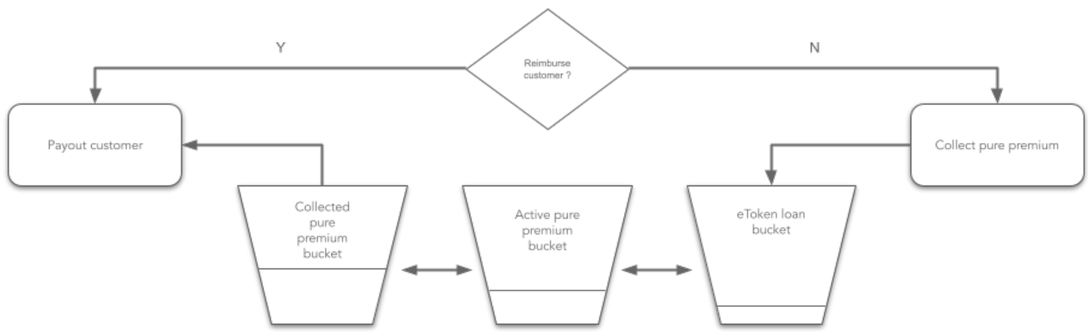

Figure 7: Resolve policy and priority system to access the solvency capital.

[6] We are here referring to the *pure premium*, not to the *full premium* of policies, composed by the *pure premium* and the *profit*

[7] The interest is generated by the Asset Management contract, that yields extra-returns on the active premiums bucket, as well as in temporary modifications of the profitability of the backed-up products.

5 Features

5.1 π-eTokens

The Ensuro protocol is composed of several different pools. Each pool π corresponds to a different *cashback period* for its liquidity providers. When users who deployed capital in a pool π send a withdrawal transaction, they accept to wait for a time *at most* as long as the cashback period of the pool[8].

Each pool π is connected to a different π-eToken. When a user deposits in a specific pool π, she receives a corresponding amount of π-eTokens, that map the liquidity deposited and accrue the interests of the deposited underlying assets. π-eTokens are minted upon deposit and burned on withdrawing. The policies sold create a continuous stream of revenues for the pools (modeled through an interest rate, see sec. 2) that increases the values of the π-eTokens, which only decrease in the case of unexpected losses. π-eTokens are ERC20 compliant and consequently support all the operations of the ERC20 interface (*transfer*, *keep balance*, etc.) plus some additional actions.

5.2 Tokenization

To implement this tokenization strategy, we introduce the following variables. Assume that the interest rate of π is r_t^π, we define the variable

- I_t^π, **Scale factor**: Ongoing interest cumulated by the pool π. Computed as

$$\begin{aligned} I_t^\pi &= (r_t^\pi \Delta t_{year} + 1)\, C_{t-1}^\pi, \\ I_0^\pi &= \ 10^{27} = 1\ ray. \end{aligned}$$

The balances of liquidity providers can be computed from I_t^π through their **scaled balances**. The scaled balance of a user b_x^π represents its share in the pool π. It is updated every time any user mints or burns π-eTokens, i.e., changes its position inside the pool π. b_x^π is computed in a very straightforward manner. If a user deposits an amount m inside the pool π, its scaled balance will increase as

$$b_{x,t}^\pi = b_{x,t_{-1}}^\pi + \frac{m}{I_t^\pi},$$

where $b_{x,t_{-1}}^\pi = 0$ if this in the first deposit by user x. Similarly, if user x withdraws an amount m, his scaled balance is updated as

$$b_{x,t}^\pi = b_{x,t_{-1}}^\pi - \frac{m}{I_t^\pi}.$$

The balance of $e_{x,t}^\pi$ π-eTokens can be computed at any time t as

$$e_{x,t}^\pi = b_{x,t}^\pi I_t^\pi.$$

[8] π can hence be regarded as a time variable; there will be a `1WEEK` pool, `1MONTH` pool, etc.

6 Conclusions

The **Ensuro protocol** builds liquidity pools to cover insurance risk. By allowing everyone to become an insurer, it pushes the boundaries of the current DeFi space. Ensuro provides a fast route to capital for insurtech companies and risk model providers who want to build novel products.

References

[1] McKinsey & Co. Global insurance insight, 2017.

[2] Michael Sonnenholzner and Achim Wambach. *Oligopoly in Insurance Markets*. 09 2006. ISBN 9780470012505. doi: 10.1002/9780470012505.tao003.

[3] Nicola Gennaioli, Rafael La Porta, Florencio Lopez-de Silanes, and Andrei Shleifer. Trust and insurance contracts. Working Paper 27189, National Bureau of Economic Research, May 2020. URL `http://www.nber.org/papers/w27189`.

[4] Sam Friedman, Malika Gandhi, and Mark Purowitz. Accelerating insurance innovation in the age of insurtech, 2019. URL `https://www2.deloitte.com/content/dam/Deloitte/us/Documents/financial-services/us-accelerating-insurance-innovation.pdf`.

[5] Joanne Linnerooth-Bayer, Koko Warner, Christoph Bals, Peter Höppe, Ian Burton, Thomas Loster, and Armin Haas. Insurance, developing countries and climate change. *The Geneva Papers on Risk and Insurance - Issues and Practice*, 34(3):381–400, 2009. doi: 10.1057/gpp.2009.15. URL `https://doi.org/10.1057/gpp.2009.15`.

[6] Evans. Sec director calss for greater retail investor access to private markets, 2020. URL `https://www.artemis.bm/news/sec-director-calls-for-greater-retail-investor-access-to-private-markets/`.