Debond: Decentralised Securities Solution

Yu LIU, Songbo WANG, Samuel Gwlanold EDOUMOU, Toufic BATRICE

info@debond.org

debond.org,

Abstract

Debond is a decentralised securities solution based on Ethereum Virtual Machine [1]. Earlier versions of the ERC-20 [2] LP tokens are a simple mathematical proof of the loan, fungible representation of the amount of the asset deposited into a smart contract, etc. In comparison, Debond uses EIP-3475: Multiple Callable Bonds Standard [3] to identify each single deposit. This means that the liquidity provider signs a derivatives agreement with the issuer. On which, both parties agree certain repayment conditions and interest rate. EIP-3475 can divide a LP into countless small sub-categories. Each part represents a derivative contract. This law of obligation can be then sold on a secondary market, like any derivative product in the current centralised economic system.

Contents

1 Definition

1.1 Variable introduction

change order of variable introduction

Variable	Description
$B\,(c, n, a)$	Bond balance associated to address a, of class c, and of nonce n. $$\begin{cases} c \subseteq \mathbb{Z} \wedge 0 < a < 2^{256} - 1 \\ n \subseteq \mathbb{Z}\ 0 < a < 2^{256} - 1 \\ a \subseteq \mathbb{Z}_{\text{hex}} \wedge 0 < a < 2^{256} - 1 \end{cases}$$
$B_{total}(\langle c_i \rangle_{i=1}^{C})$	The sum of all the balances of all the nonce of all the classes in the sequence $\langle c_i \rangle_{i=1}^{C}$. $$B_{total}(\langle c_i \rangle_{i=1}^{C}) = B_{sum}(n_l)$$
$B_{\text{sum}}(n)$	The sum of all the balances of all the nonce from the first nonce 0 to the n-th nonce . $$\sum_{i=0}^{n} B(\langle c_i \rangle_{i=1}^{C}, n)$$
$\langle c_i \rangle_{i=1}^{C} = \langle c_1,\, c_2,\, c_3,\, \cdots,\, c_C \rangle$	A sequence of classes, C=the number of elements in the sequence.
$b_{erc20}(a)$	Balance of an ERC-20 token of an address a
F_n	The n-th Fibonacci number, with $F_0 = 0, F_1 = F_2 = 1, F_3 = 2, \ldots$
n_{now}	Nonce of the current block time T_l of a class c
$n_l(c)$	Nonce of the last nonce issued defined by `last_nonce[c]`
$\langle n_i \rangle_{i=1}^{N} = \langle n_1,\, n_2,\, n_3,\, \cdots,\, n_N \rangle$	A sequence of nonce.
T	Number of seconds defined by `block.timestamp`
T_{year}	Number of seconds in a year T_{year}=31536000
T_{epoch}	`time_epoch` Length of a nonce in time

Variable	Description
$T_{\text{start}}(c, n)$	Number of seconds of the starting time of a bond class and of a bond nonce, defined by `bondClass[c]._nonceInfo[n][0]`
. $T_{\text{maturity}}(c, n)$	Number of seconds of the maturity time of a bond class and of a bond nonce, defined by `bondClass[c]._nonceInfo[n][1]`
T_l	Number of seconds of the current block time defined by `time.now`
$\Delta n(dT)$	Difference in nonce of a difference in time dT $$\Delta N(dT) = \frac{dT}{T_{epoch}}$$
ΔT	Time lapse between two timestamps T_1, T_2
$\Delta T(c, n)$	Time lapse from the starting time of a nonce to the start of the next nonce $$\Delta T(c, n) = T_{start}(c, n+1) - T_{start}(c, n)$$
$s_{\text{collateral}}$	All collateralised supply of a toke, locked in LP $$s_{\text{collateral}} = \sum_{i=0}^{n} l_i n$$
s_{total}	Output of ERC-20 `totalSupply()` function, the total supply of a token
s_{max}	Output of ERC-20 `maximumSupply()` function, the maximum supply of a token
t_i, l_i	t_i represents an ERC-20 token, and l_i is the amount of liquidity of the corresponding token
$P\left(t_1, t_2; l_1, l_2\right)$	Price of unit token t_1 of t_2, calculated by $P = \frac{l_2}{l_1}$
$P_{determined}\left(t_1, t_2; l_1, l_2, T\right)$	The arbitrated price of unit token t_1 of t_2, calculated by $P = \frac{l_2}{l_1}$ at a giving time T

Variable	Description
$C_{\text{DBIT}}(s_{\text{total}})$	Minting Cost, the contract determined price of unit DBIT in USD, calculated by: $$1.05^{\log_e s_{\text{total}}}$$
$C_{\text{DGOV}}(s_{\text{total}}, s_{\max})$	Price of unit DGOV in DBIT, calculated by: $$100 + (S_{total}10000/S_{max})^2/100000$$
$k(t_1, t_2)$	Total liquidity of the automatic market maker (AMM) for tokens t_1, t_2, the amount of the tokens l_1, l_2 being fixed by the relation $l_1 l_2 = k(t_1, t_2)$
$p(t_1, t_2)$	The pair liquidity pool of t_1, t_2
$l_{determined}(t_1, t_2; l_1)$	The output liquidity l_2 of contract determined price (CDP) method
π^*	The principal used in an investment.
I	The face interest from an interest settlement $I = \pi\iota$
ι	The face interest rate of an investment from an interest settlement $\frac{I}{\pi}$
$\iota_{benchmark}$	The benchmark interest rate of an investment from an interest settlement. A variable defined by `IR.benchmarkIR` and can be updated by `governanceContract.updateBenchmarkIR()` $$2\iota_{benchmark} = \iota_{float} + \iota_{fix}$$
$\iota_{float}(t)$	The floating interest rate of an investment from an interest settlement. Based on benchmark IR and the ratio of floating rate bond to fixed rate bond of the same settlement token t. $$\iota_{float}(t) = 2\iota_{benchmark}\frac{B_{fix}(t)}{B_{float}(t) + B_{fix}(t)}$$

Variable	Description
$\iota_{fix}(t)$	The fixed interest rate of an investment from an interest settlement. Based on benchmark IR and the ratio of floating rate bond to fixed rate bond of the same settlement token t. $$\iota_{fix}(t) = 2\iota_{benchmark}\frac{B_{float}(t)}{B_{float}(t) + B_{fix}(t)}$$
ι_{total}	The total nominal interest rate of an investment $$\sum_{i=0}^{n} \iota_i$$
ι_{compound}	The compound interest rate of an investment, with n times of settlement $$(\prod_{i=1}^{n}(1+\iota_i)) - 1$$
$\iota_{\text{remaining}}$	The remaining interest rate of an investment from χ-th interest settlement to n-th $$\sum_{i=\chi}^{n} \iota_n$$
λ	The amount of a loan
Γ	The value of the collateral
ρ	The collateral rate of a derivative $\frac{\lambda}{\Gamma}$
$T_{estimate}(n)$	The estimated redemption time based on the average liquidity flow over last month μ_{month}, and the additional liquidity needed. $$T_{maturity}(n) + \frac{B_{\text{sum}}(n)(1+\iota_{benchmark}) - B_{\text{sum}}(n_l)}{\mu_{month}} T_{epoch}$$

Variable	Description
R_{DBIT}	DBIT received as a reward for voting for a proposal $$v \sum_i \frac{r^i_{DBIT}}{V_i}$$
v	The amount of vote tokens used to vote for proposal
r^i_{DBIT}	DBIT reward distributed on day i
V_i	The amount of vote tokens submitted on day i

2 Introduction

In this paper, we present Debond, a novel derivatives based LP proof system that gives liquidity providers and the borrower a customisable agreement over their loan/debt. It also provides tools to manage the redemption conditions and price ranges in which their capital is used. Using ERC-3475 Multiple Callable Bonds Standard, the systematic way of the liquidity fragmentation and gas efficient transaction, new financial derivatives can be introduced to decentralised finance.

Bonds tend to be less volatile and less risky than other financial products. Other financial derivatives can be used for Hedge and insurance. This reduces the risk involved. Because of these facts, they are some of the most important tools of financial markets. They serve as a stabilizer for the system. They perform well when the market is on decline, as interest rates fall and bond prices rise in turn.

If the main goal of an investor is to always be able to get their principal back from the staking, a stable bond is their best choice. When they stake for a company or a government bond, they pay for their expectation of the future of the borrower. If needed, buyers of the bond can always get some of their liquidity back, either through pledging them for loans or selling on the secondary market. The bond holders, being creditors have a higher priority level then the stockholders. If bankrupt, the asset of the borrower will be first used to repay all the bondholders and then the stock owners.

Decentralised finance is one of the most volatile markets where the trust is hard to find, and the risk is omnipresent. What the market really needs is a decentralised bonds system based on trust-less smart contracts, and

derivatives that can be used to hedge against the risk. The huge bond LP can be used to stabilize the market. The bond holders can then get relatively stable income from the expansion of the market. During a declining phase, The overly sold bond can be burned by Debond Governance Platform, like bad debt being relieved by the state intervention. The economy can then go into the next expansion phase [5].

- **Bond:** Instrument of indebtedness of the bond issuer to the holders.
- **Derivative:** Contract between two or more parties whose value is based on an agreed-upon underlying financial asset (like a security) or set of assets (like an index). Common underlying instruments include bonds, commodities, currencies, interest rates, market indexes, and stocks.
- **LP token:** Fungible mathematical proof of a loan to a liquidity pool $p\left(t_1, t_2\right)$.
- **ERC-3475:** Replacement of the ERC-20 LP token. ERC3475 is non-fungible, computationally structured and gas-efficient.
- **Debond:** Decentralised Bond implemented using ERC-3475.
- **Debond DEX:** Decentralised Bond Exchange, allowing any ERC-3475 bond to be traded on a secondary market.
- **Fixed Interest Rate Debond:** Debond that pays fixed interest over its entire term, defined at issuance.
- **Floating interest rate Debond:** Debond that does not have a fixed rate of interest over its entire term.
- **Coupon:** Interest rate that the issuer pays to the holder. For fixed rate bonds, the coupon is fixed while for floating rate bons, it is variable.
- **Yield:** Rate of return received from investing in the bond.
- **Securitised Debt:** is the tradable security that represents the right over an income-producing asset, or a debt.
- **AMM:** Automated market maker that pool liquidity and make it available to traders according to the formula x*y=k.
- **CDP:** Contract Determined Price, a nonlinear function that controls the minting cost of a certain Token.

- **APM**(t_1, t_2, l_1)**:** Automated Pair Maker method, method that automatically pairs a single token to a LP. LP providers need only one token t_1 to provide liquidity for a pair $p\,(t_1, t_2)$.
- **Debond Bank:** set of smart contracts, trustee and bond issuer.
- **Settlement Currency:** Currency to be used to repay securities or interests in securities.
- **DBIT:** Debond Index Token (DBIT), ERC-20 token, a quantified representation, measuring the investment performance and characteristics of the Debond market. DBIT is also the settlement currency of the Debond ecosystem.
- **DBIT Bond:** The redemption of the DBIT bond is paid in DBIT.
- **Minting Cost:** is the price per token of the bond in the first market, as defined by CDP.
- **QTM:** Quantity Theory of Money tells that the general price level of goods and services is proportional to the money supply in an economy. Debond protocol uses this theory to control the market supply and velocity of the bond settlement token DBIT.
- **D/Capital:** Decentralised model of a Funds Management Entity, allowing the liquidity to be used more efficiently and generate income for the creditors.
- **Proposal Contract:** Logical subject of a VM recognisable proposal. Converted from nature language to a computational representation.
- **DBGP:** Decentralised Bond Governance Platform. Current governance platforms such as snapshot, provide only a place for counting votes. No obligations are imposed to the private key owner whereas DBGP obliges the creator of the proposal to convert his idea into a proposal contract. The access to the governance contract can then be given to this proposal contract. This platform is also used in the management of D/Capital.
- **DGOV:** Governance token of DBGP.

3 Debond

Debond is a financial tool to securitize any form of digital debt or asset into a bond class. This class can be then fragmented into sub-categories. We use

the term "nonce" to represent such sub-categories. Nonces are not interchangeable, We can consider each nonce as a separate ERC-721 derivative contract.

3.1 Features

- **Non-fungible:** Not like any form of ERC-20 LP token, a Debond with different nonce is not mutually interchangeable, hence not fungible.
- **Derivatives:** Each Debond can be regarded as a smart contract based derivative.
- **Fragmented:** Debond can fragment liquidity pool, debt and any form of the digital asset into many pieces. Each fragment represents a separate sub-category of the object in question.
- **Multi-dimensional:** Each ERC3475 Debond stores an array of integers, not like a ERC-20 token can store only a mapping from an address to a balance. This feature can be used to store redemption conditions and different interest rates.
- **Gas-efficient:** Current existing fragmentation of LP or NFT use ERC-20 token standard. This means each fragmentation of NFT or LP needs the deployment of a new smart contract. Thus generate unnecessary gas expense,
- **Multiple fragments:** Because of the high gas spent, the current fragmentation systems must limit the number of classes and sub-categories generated. ERC-3475 can in turn generate almost countless classes and sub-categories without the need to publish a new smart contract.
- **Callable:** the debts that a Debond represents must be fulfilled when the conditions are met. It can be a fixed amount of repayment or any floating interest based on a predetermined index factor.
- **Obligated:** ERC-3475 obliges both the borrowers/debtors and the creditors to fulfil their promise. This feature can be used to make sure that the LP will not be emptied suddenly. In return the borrower will ensure the promised yield.
- **Marketable:** Debond is itself a financial product that can be exchanged and speculated on a secondary market.
- **Securitised:** An derivative issuer can design a marketable financial instrument by merging or pooling various derivative contract into one group. The issuer can then sell this group of repackaged assets on the secondary market.

3.2 EIP-3475

A standard interface for contracts that manage multiple callable bonds. A single contract includes any given number of bond classes, bond nonce, bond balance of an address. This standard provides independent functions to read, transfer any collection of bonds, as well as allow bonds to be redeemed from the bond issuer if certain conditions are met. This token standard can replace current ERC-20 LP token. ERC-3475 has a more complex data structure, which will allow the LP token to store more information, and allow the developer to build more sophisticated logic for the redemption and reward system of the DEFI project in question.

This API standard allows for the creation of any number of bond types in a single contract. Existing LP token standards like ERC-20 require deployment of separate factory and token contracts per token type. The need of issuing bonds with multiple redemption data can't be achieved with existing token standards. ERC-3475 Multiple Callable Bonds Standard allows for each bond class ID to represent a new configurable token type, and for each bond nonce to represent an issuing date or any other forms of data in uint256. Every single nonce of a bond class may have its own metadata, supply and other redemption conditions.

Current LP token is a simple ERC-20 token, which has not much complicity in data structure. To allow more complex reward and redemption logic to be built, we need a new LP token standard that can manage multiple bonds, store much more data and is gas efficient. ERC-3475 standard interface allows any tokens on solidity compatible block chains to create its own bond. These bonds and derivatives with the same interface standard can be exchanged in the secondary market. And it allows any 3rd party wallet applications or exchanges to read the balance and the redemption conditions of these tokens. ERC-3475 bonds and derivatives can also be packed into separate packages. Those packages can in their turn be divided and exchanged in a secondary market.

New functions built in ERC-3475 Multiple Callable Bonds Standard, will allow the users to economize their gas fee spend. Trading and burning of ERC-3475 Bonds will also multiply tokens market cap, heLPing it to recover from recession period. Existing structures, such as AMM exchanges or lending platforms can be updated to recognize ERC-3475 bonds or derivatives.

ERC-3475 is a tokenized exchangeable asset. A bond contract can store the data of 2^{256} bond classes and for each class, 2^{256} bond nonces. Unlike a typical ERC-20 or ERC-721 token, an ERC-3475 bond is semi-fungible and multi-dimensional. Any bond nonce of a class is non-fungible from another

nonce or class. It has an identified uint256 array, in which we can store the redemption time, interest rate, redemption conditions and other info. For example:

3.2.1 Balances

need to be more clear

`"bondClass[class]._balances"` is the array of bond class to a nonce, and to the balance. `mapping(address =>mapping(uint256=> uint256)) _balances;` is the balance of an address are associated with the following 3d array: *B (a,c,n)*, where a is the vector of address, c is the vector of class and n is the vector of nonce.

$$\begin{cases} c \subseteq \mathbb{Z} \wedge 0 < a < 2^{256} - 1 \\ n \subseteq \mathbb{Z} \wedge 0 < a < 2^{256} - 1 \\ a \subseteq \mathbb{Z}_{hex} \wedge 0 < a < 2^{256} - 1 \end{cases}$$

e.g.

```
bondClass[1].0x2d03B6C79B75eE7aB35298878D05fe36DC1fE8Ef=> (5 =>
500000000));
```

this example gives the balance of:
address 0x2d03B6C79B75eE7aB35298878D05fe36DC1fE8Ef possess 500000000 of bond class 1, bond nonce 5.

3.2.2 Multidimensional data structure

```
mapping (uint256=> uint256[]) _nonceInfo;
string[] nonceInfoDiscription;
bondClass[class]._nonceInfo[n][];
bondClass[class].nonceInfoDiscription[];
```

`_nonceInfo` is the mapping from a bond nonce to a list of uint256 variables, list of bond nonce to bond info. In this list, the 0th variable MUST be $T_{start}(c, n)$, the starting time of the nonce in question. the 1st variable MUST be $T_{\text{maturity}}(c, n)$ the maturity time of the nonce in question. Other variables can be described by `bondClass[class].nonceInfoDiscription[]`; and defined by code.

e.g.

```
["1615584000","1616584000",(uint256)...]
```

this example gives the starting time of the bond nonce of the bond class, which is 1615584000; the maturity time, 1616584000...

3.2.3 Bond transfer

`"transferBond()"` allows the transfer of any number of bond types from an address to another.
The"`"_from"` argument is the address of the holder whose balance is about to decrees.
The "`"_to"` argument is the address of the recipient whose balance is about to increase.
The `"class"` is the list of classes of bonds or derivatives, the first bond class created will be 0, and so on.
The `"nonce"` is the list of nonce of the given bond class. This param is for distinctions of the issuing conditions of the bond.
The `"_amount"` is the list of amounts of the bond that will be transferred from "_from"address to "_to" address.

$$\begin{cases} B\left(a_1, c, n\right) := B\left(a_1, c, n\right) - \_amount, \\ B\left(a_2, c, n\right) := B\left(a_2, c, n\right) + \_amount, \\ \qquad\qquad if \_amount \leq B\left(a_1, c, n\right) \end{cases}$$

$$\begin{cases} B\left(a_1, c, n\right) := B\left(a_1, c, n\right), \\ B\left(a_2, c, n\right) := B\left(a_2, c, n\right), \\ \quad if \_amount > B\left(a_1, c, n\right) \end{cases}$$

```
function transferBond(address _from, address _to, uint256[]
calldata class, uint256[] calldata nonce, uint256[] calldata
_amount) external returns(bool);
```

e.g.
`transferBond(0x2d03B6C79B75eE7aB35298878D05fe36DC1fE8Ef,`
`0x82a55a613429Aeb3D01fbE6841bE1AcA4fFD5b2B,`
`[1,2,4], [42,61,25],[500000000,60000000,150000000]);`
This example shows the transfer from `"_from"` address, to `"_to"` address, `"500000000"` of bond class `"1"` nonce `"42"`, `"60000000"` of bond class `"2"` nonce `"61"`, `"150000000"` of bond class `"3"` nonce `"25"`. Returns a bool. `"true"` if passed.
*NOTE: This transfer is atomic i.e., If one transfer in the transaction fails, the function will revert all the previous passed transfers within the transaction.

4 Debond Automated Pair Maker

Debond's Automated Pair Maker (APM) is constituted of one address representing a "single consolidated pool". This means to whichever pool the liquidity is added, tokens are always sent to the same address. This prevents unnecessary gas fees spending when new tokens are added and also helps in consolidating liquidity.

Further, unlike traditional AMMs, the price for any given token cannot be directly correlated to the token reserve held by the pool. This is because, in Debond's APM, aggregated token reserve belongs to several pools. Further, Debond's APM uses the concept of *Virtual Liquidity Pools (VLPs)*: The VLP is a mapping within the APM which tracks the fraction of reserves of each token for given pair of tokens. This mechanism is inspired from Uniswap.

The APM handles 3 main functions:

1. *Adding liquidity:* The user adds liquidity in denomination of one type of token only, and an equivalent amount of the other token from the given pair is minted. This is possible because the other token is always DBIT or DGOV, minted by Debond Bank then supplied to the APM. Further, addition of liquidity is done through the Debond Bank while staking/buying bonds, effectively linking both processes. On the contrary, the user can directly add liquidity to the APM with this method, but they will not be issued bonds to remove the liquidity later. Thus, the process resembles the mechanism of classical AMM and their LP tokens.
2. *Swap:* This process is same as classical AMM following the formula $X \times Y = K$.
3. *Removing Liquidity:* The user can remove the provided liquidity for its original token only through redemption of bonds via Debond Bank on or after maturity date. During redemption, the token amount being redeemed by the provider is deducted from the aggregated APM reserve for that token. The mechanism and advantages for this are explained further in this chapter.

4.1 APM Functional Architecture

Consider the following example. Suppose that for an hypothetical sweepstake, when the participant registers, he/she is given 1 *entry*. For n partici-

pants, there will be n entries. Suppose the grand price P is divided equally among all entries n, then each participant with 1 entry would get price of value $\frac{P}{n}$.

Suppose that a VIP person gets m extra entries. This is as if he/she participates m times more. Thus, the total entries would thus become $n+m$ and each entry would thus fetch $\frac{P}{n+m}$ price.

By proof of induction, the VIP person, who has $1+m$ entries in total (1 original, m extra), would thus be allocated price of $\frac{P}{n+m} \times (1+m)$. Suppose that the grand price P is now doubled, the price won per entry is also doubled. However, the number of entries $n+m$ do not change as a consequence.

This simple concept is extremely useful in the context of Debond's APM when the grand price P from the above example is replaced by total reserve R for a token (say R_A represents total reserve/balance of token A), participants are replaced by token pairs (say all pairs for token A only, e.g. Pair $A-B$, $A-C$, $A-D$, etc.) and entries of each pair represent how much portion of reserve of a token belongs to them (say if pair $A-B$ has 1 entry out of 10, it has 10% of R_A). Further, in Debond's APM:

1. Total entries for a given token represent total reserve for that token.
2. Initially, when the token is listed/supported, 1 entry is given for adding 1 token.
3. When the liquidity is removed directly by the bank, the entries are not changed but only the reserve of that token changes.
4. This changes the one-to-one peg between the entries and the total reserve for that token as now more entries represent less tokens. However, aforementioned **point 1 always remains true**.
5. After point 4, whenever new liquidity is added or removed through swap, appropriate entries are added or removed respectively.
6. For a given token, the appropriate no. of entries to be added/removed becomes:

   $$\textit{Amount of Entries} = \frac{\textit{Total Entries}}{\textit{Total Reserve}} \times \textit{Amount of Token (added/removed)}$$

   since point 1 is always valid.

4.2 APM Implementation

For implementation, 3 things for each token are tracked through 3 mappings.

- `totalReserve[token address]`: stores the total balance of a given token.
- `totalEntries[token address]`: stores the total entries of a given token across all its pools.
- `entries[tokenA address][tokenB address]`: stores the number of entries of a given tokenA in tokenA-tokenB pair virtual pool.

For the rest of the paper we define the following notation:

- E_{t_A}: the total entries of token t_A across all its pools
- ϵ_{t_A,t_B}: the number of entries of token t_A in the pair (t_A, t_B)
- R_{t_A}: the reserve (total balance) of token t_A
- r_{t_A,t_B}: the balance of token t_A within the virtual pair (t_A, t_B)

4.2.1 Get Reserve

The balance of a given token t_A within the (t_A, t_B) virtual pair is given by:

$$r_{t_A,t_B} = \frac{\epsilon_{t_A,t_B}}{E_A} R_A$$

The balance of a given token t_B within the (t_A, t_B) virtual pair is given by:

$$r_{t_B,t_A} = \frac{\epsilon_{t_B,t_A}}{E_B} R_B$$

4.2.2 Adding Liquidity in Pair

When the liquidity Δ_{t_A,t_B} is added for given token t_A in (t_A, t_B) virtual pair, an appropriate amount of entries for token t_A are to be added. The entries are given by:

$$\epsilon_{t_A,t_B} = \frac{E_A}{R_A} \Delta_{t_A,t_B}$$

The three mappings are to be updated as well:

- ϵ_{t_A,t_B} += Δ_{t_A,t_B}
- E_{t_A} += Δ_{t_A,t_B}
- R_{t_A} += Δ_{t_A,t_B}

The same procedure is followed for the other token in the virtual pair.

4.2.3 Removing Liquidity in Pair

When the liquidity Δ_{t_A,t_B} is removed for given token t_A in (t_A, t_B) virtual pair, an appropriate amount of entries for token t_A are to be removed. The entries are given by:

$$\epsilon_{t_A,t_B} = \frac{E_A}{R_A} \Delta_{t_A,t_B}$$

The three mappings are to be updated as well:

- ϵ_{t_A,t_B} -= Δ_{t_A,t_B}
- E_{t_A} -= Δ_{t_A,t_B}
- R_{t_A} -= Δ_{t_A,t_B}

The same procedure is followed for the other token in the virtual pair.

4.2.4 Swapping

Swapping token t_A for token t_B can be dissected into a two step process: 1) Adding liquidity for token t_A (see section 4.2.2) and then 2): Removing liquidity for token t_B (see section 4.2.3).

The APM uses the standard AMM formula to get the right amount of token t_B for input token t_A. According to constant product formula, we have:

$$r_{t_A,t_B} \times r_{t_B,t_A} = k$$

where k is constant.

If Δ_{t_A} of token t_A is swapped, the amount Δ_{t_B} of token t_B to be received is given by the following formula:

$$\Delta_{t_B} = r_{t_B,t_A} - \frac{k}{r_{t_A,t_B} + \Delta_{t_A}}$$

4.2.5 Removing Liquidity

This is different from removing liquidity in pair and can be done by the bank when redemption of a bond happens. In this case, only the `totalReserve` changes and `entries` remain the same.

The reserve of the given token will decrease to:

$$R_{Token} := R_{Token} - \Delta_{Token}$$

5 Contract determined price calculation

Debond uses the contract determined price (CDP) mechanism to estimate the amount of Debond tokens (DBIT or DGOV) to mint when a trade happens through a given pair. Indeed, the amount of Debond tokens to mint depends on the collateralized supply of the given token, which is used to calculate the token price in USD.

5.1 CDP price of DBIT

From the collateralized supply of DBIT S_{dbit}^{coll}, the CDP price of DBIT to USD can be calculated by using the following formula:

$$C_{\text{DBIT}}(S_{dbit}^{coll}) = \begin{cases} 1 & \text{if } S_{dbit}^{coll} <= 1000 \\ \\ 1.05^{\log_2(S_{dbit}^{coll}/1000)} & \text{if } S_{dbit}^{coll} > 1000 \end{cases}$$

Figure 1 shows the minting price of DBIT to USD in terms of the DBIT collaterlized supply s_{dbit}^{coll}. As expected, the price remains constant when the collaterlized supply S_{dbit}^{coll} is less than 1000. For higher values of S_{dbit}^{coll} the price increases. However, even if S_{dbit}^{coll} increases by six order of magnitude

(from 10^3 to 10^9), the increase in the minting price C_{DBIT} remains bounded between 1 to 2.6 USD. This guaranties the token to be quasi-stable in the primary market, while its price in secondary market can be more volatile, since in secondary market the price follows the supply and demand rule.

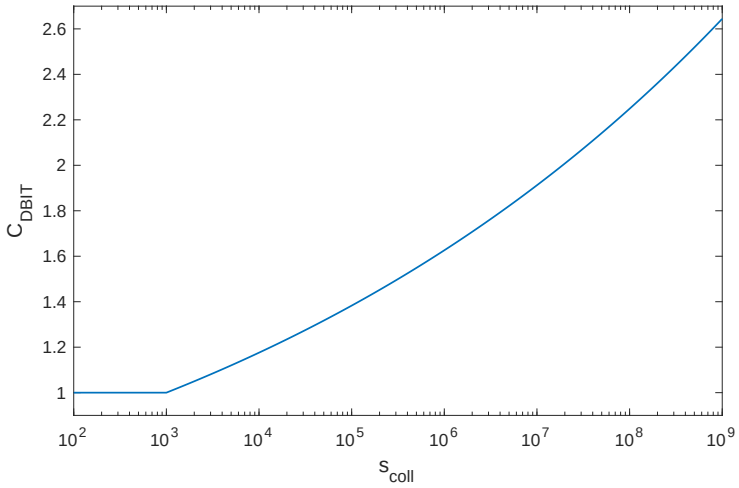

Figure 1: DBIT minting price C_{DBIT} to USD in terms of S_{dbit}^{coll}

5.2 Decentralized Bonds Index Token(DBIT)

DBIT is an ERC-20 token pegged to the bond index. The overall expectation of the Debond market will be reflected in the form of a tokenized Index. The price of DBIT is highly associated with the general performance of the Debond market [15].

5.2.1 Settlement currency

DBIT will be used for bond interest payment and secondary bond market transaction. The final goal of DBIT is to become the market settlement currency. It's part of the index system that reflects the overall trend of the bond market.

5.3 Decentralised bond governance token (DGOV)

DGOV is the governance token of a decentralised governance platform, a shareholding proof of their investment in D/Capital. By holding DGOV,The

holder of DGOV indirectly holds the share of the On-chain and off-chain projects that D/Capital invested in. This allows the better use of the Debond asset, and share profit with the DGOV holder.

5.3.1 Bonus share

Apart from the governance reward when voting for a proposal, there will be bonus share from D/Capital and speculative actives. The profit will be pulled in to the DGOV swap pair, thus giving additional bonus reward to the DGOV holder.

5.3.2 CDP price of DGOV

As for DBIT, CDP can be used to mint DGOV when adding liquidity to a pair (DGOV, Token). Unlike DBIT, however, CDP doesn't allow to calculate DGOV price to USD directly, instead, it allows to get DGOV price to DBIT in terms of DGOV collaterlised supply S^{coll}_{dgov}:

$$C_{dgov}(S^{coll}_{dgov}) \;=\; 100 + (S^{coll}_{dgov}/33333)^2$$

Since the CDP formula above represents the DGOV price in DBIT, it can be used to calculate the price of DGOV in USD by multiplying the result by the CDP price of DBIT to USD. Therefore, the CDP price of DGOV to USD is

$$C_{DGOV,USD} \;=\; C_{dgov}(S^{coll}_{dgov}) \cdot C_{dbit}(S^{coll}_{dbit})$$

by substituting both $C_{dgov}(S^{coll}_{dgov})$ and $C_{dbit}(S^{coll}_{dbit})$ by their formula, the CDP price of DGOV in USD becomes

$$C_{\text{dgov}}(S^{coll}_{dgov}, S^{coll}_{dbit}) \;=\; \begin{cases} 100 + (S^{coll}_{dgov}/33333)^2 \\ \\ 1.05^{\log_2(S^{coll}_{dbit}/1000)} \times \left[100 + (S^{coll}_{dgov}/33333)^2\right] \end{cases}$$

where in the above formula, the condition is that of DBIT-USD price calculation, means that if S^{coll}_{dbit} is less than or equal to 1000, the first formula is used, otherwise, the second formula is used. Figure 2 shows the variations of the minting price C_{dgov} of DGOV to USD in terms of both S^{coll}_{dgov} and

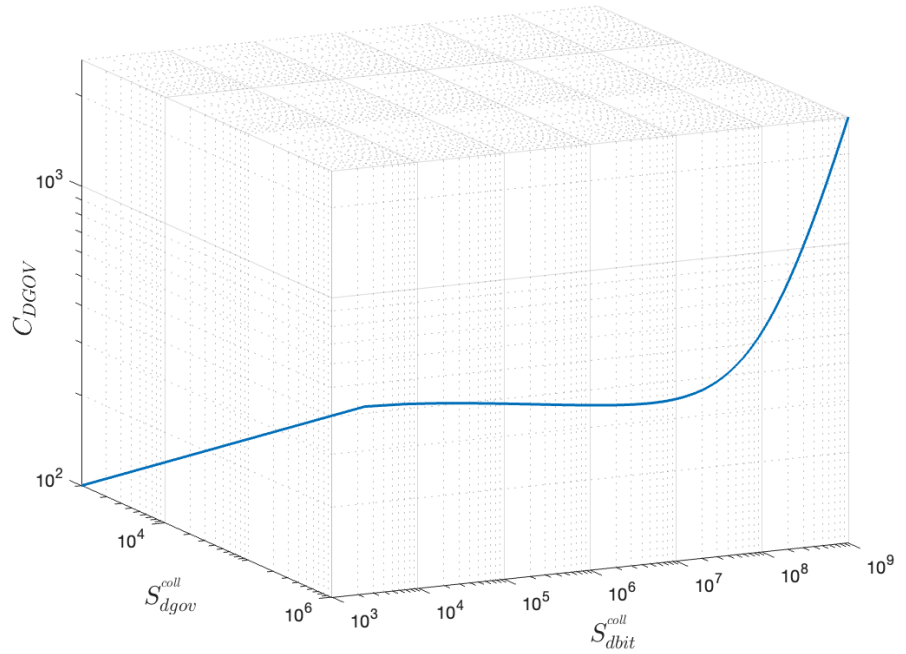

Figure 2: DGOV minting price C_{DGOV} in USD in terms of S^{coll}_{dgov} and S^{coll}_{dbit}

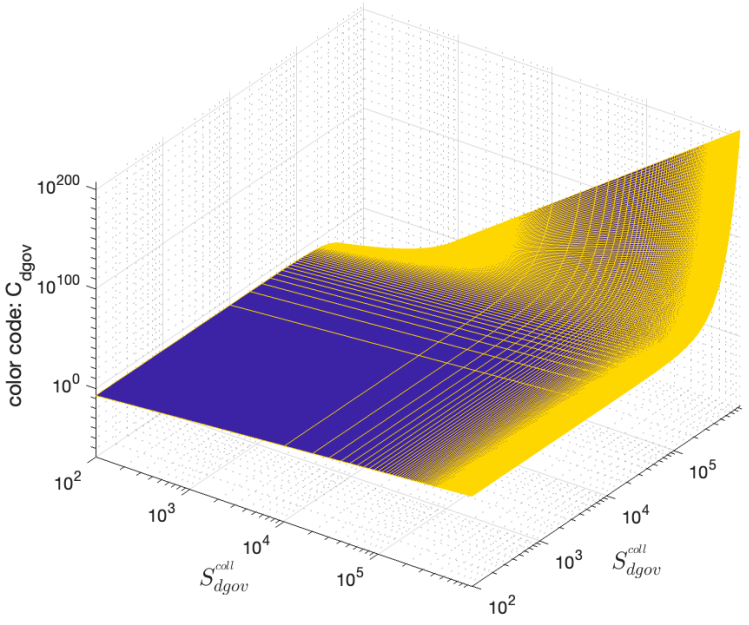

Figure 3: color code - C_{DGOV} in terms of S^{coll}_{dgov} and S^{coll}_{dbit}

S^{coll}_{dbit}. For small values of the token supplies $S^{coll}_{dbit} < 10^7$ and $S^{coll}_{dgov} < 10^6$ the

minting price of DGOV varies slowly between 100 and 400 USD.

For $S_{dbit}^{coll} > 10^7$ the price increases sharply to reach more than 2000 USD at when $S_{dbit}^{coll} = 10^9$. This means that the more DBIT minted through issuing DBIT-Token bonds, the more the increase of DGOV minting price to USD.

6 APPLICATIONS

We can regard Debond as a set of applications built on top of Ethereum block-chain. The first category is the debt accounting system, providing users with more powerful ways of managing their staking, loan and interact with smart contracts using their pledged assets. This includes the securitisation of LP token, financial derivatives, bonds etc.. The second category is the P2P landing application, where a loan is issued based on the specified terms and pledged assets involved. There is also an off-chain possibility open to the transformation of an off-chain debt or pledge to on-chained cryptographic data. Finally, there are applications such as hedging contracts or D/Capital, a smart contract based hedging or venture funds. With this application, the pledged asset involved, will be managed by by D/Capital. It will be used to fund on-chian and off-chain projects, hedging funds and other supercilious investment.

Debond systems have many applications ranging from financial derivatives to fixed-rated bond products. Individual bond represents the debt of a digital currency. Debond products are easy to implement. Using the same ERC-3475 interface, anyone can create and build his own bond product. Those bonds or derivatives are, in theory, the database structured object, intelligible by other smart contracts. The array of integers is the basic building module of an ERC-3475 bond. Every bond contract has the method listed below:

```
1  pragma solidity 0.8.14;
2
3  interface IERC3475 {
4      function balanceOf(address account, uint256 classId,
           uint256 nonceId) external view returns (uint256);
5
6      function transferFrom(address from, address to, uint256
           classId, uint256 nonceId, uint256 amount) external;
7
8      function issue(address to, uint256 classId, uint256 nonceId
           , uint256 amount) external;
9
10     function redeem(address from, uint256 classId, uint256
           nonceId, uint256 amount) external;
```

```
11
12      function burn(address from, uint256 classId, uint256
            nonceId, uint256 amount) external;
13
14      function totalSupply(uint256 classId, uint256 nonceId)
            external view returns (uint256);
15
16      function nonceInfos(uint256 classId, uint256 nonceId)
            external view returns (uint256[] memory);
17
18      function approve(address spender, uint256 classId, uint256
            nonceId, uint256 amount) external;
19
20      function setApprovalFor(address operator, uint256 classId,
            bool approved) external;
21
22      function batchApprove(address spender, uint256[] calldata
            classIds, uint256[] calldata nonceIds, uint256[]
            calldata amounts) external;
23
24      function redeemedSupply(uint256 classId, uint256 nonceId)
            external view returns (uint256);
25
26      function activeSupply(uint256 classId, uint256 nonceId)
            external view returns (uint256);
27
28      function burnedSupply(uint256 classId, uint256 nonceId)
            external view returns (uint256);
29
30      function symbol(uint256 classId) external view returns (
            string memory);
31
32      function classInfos(uint256 classId) external view returns
            (uint256[] memory);
33
34      function classInfoDescription(uint256 classInfo) external
            view returns (string memory);
35
36      function nonceInfoDescription(uint256 nonceInfo) external
            view returns (string memory);
37
38      function nonceInfos(uint256 classId, uint256 nonceId)
            external view returns (uint256[] memory);
39
40      function isRedeemable(uint256 classId, uint256 nonceId)
            external view returns (bool);
41
42      function allowance(address owner, address spender, uint256
            classId, uint256 nonceId) external view returns (
            uint256);
43
44      function isApprovedFor(address owner, address operator,
            uint256 classId) external view returns (bool);
45
```

```
46      event Transfer(address indexed _operator, address indexed
            _from, address indexed _to, uint256 classId, uint256
            nonceId, uint256 amount);
47
48      event Issue(address indexed _operator, address indexed _to,
             uint256 classId, uint256 nonceId, uint256 amount);
49
50      event Redeem(address indexed _operator, address indexed
            _from, uint256 classId, uint256 nonceId, uint256 amount
            );
51
52      event Burn(address indexed _operator, address indexed _from
            , uint256 classId, uint256 nonceId, uint256 amount);
53
54      event ApprovalFor(address indexed _owner, address indexed
            _operator, uint256 classId, bool _approved);
55  }
```

With the methods listed above, a front-end interface or Debond wallet can read the balances, class, nonce, progress, redemption time and other info from a bond contract.

6.1 Debond Wallet

Debond wallet allows users to manage accounts and their ERC-3475 assets in a variety of ways. Built on top of a hot-wallet such as Metamask[8], Debond wallet allows users to read, buy, redeem, exchange and transfer any ERC-3475 standard asset. Debond wallet is a web app which was designed specially for ERC-3475 assets. It allows any user to have an easier access to Web3 infrastructure, interact with the Ethereum ABI, and communicate with the EVM smart contract. Debond wallet is user friendly and compatible with all web3 browsers (like Chrome with Metamask plug-in).

Debond wallet supports any ERC-3475 bond or derivatives. Those financial products can be displayed, traded and read on a Debond wallet without any development needed. This feature of the "Bond system" displays the transition function described further below in this document.

```
1   pragma solidity 0.8.14;
2
3   contract Bond {
4       function transferBond (
5           address _from,
6           address _to,
7           uint256[] calldata class,
8           uint256[] calldata nonce,
9           uint256[] calldata amount
10      ) external returns(bool);
```

```
11 }
```

With this interface, Debond wallet will use web3 to interact with the smart contract. Debond-based DEFI projects act as a derivative issuer. They can potentially include other feature and method. By doing so, the market will explore all the possibilities until it finds several most well-designed derivatives. Debond wallet provides a derivative infrastructure for this potential market.

Existing liquidity pool can be adapted to support ERC-3475. The implementation of ERC-3475 bond to replace the existing ERC-20 LP token, will be gradually taking place. The Debond wallet will be a practical way to display all your staking. In the form of bonds or derivatives, the infrastructure will be adapted to the needs of other DEFI project developers. Debond wallet serves as a way of displaying all ERC-3475 compatible DEFI project's LP data. Using ERC-3475 interface, a DEFI project can simply use Debond wallet to interact with their smart contract, without any adaptation or front-end developments needed.

6.2 Debond DEX

A open secondary market is the most easy way to exchange Debond. Common AMM method [9] can't be adapted to ERC-3475 bonds. The derivatives created with ERC-3475 need the implantation of a specially designed auction system. The main challenge in building such secondary bond market is that the majority of the bonds need to be bound together before putting into an open order. The Debond DEX uses dutch auction method. Buyers will compete with other parties to find the real market price of a derivatives.

6.2.1 Debond Auction Object

Each auction is defined by the auction id, the list of Debond Bonds and their corresponding ids and the auction params. The later are given by the object `AuctionParam`, and characterize all parameters that define the auction, like the auction duration, the max price and the min price, the auction owner, the auction state, the curving price, etc.

```
1 pragma solidity 0.8.14;
2
3 contract ExchangeStorage {
4     struct AuctionParam {
5         address owner;
6         uint256 startingTime;
```

```
7           uint256 endingTime;
8           uint256 duration;
9           address erc20Currency;
10          uint256 maxCurrencyAmount;
11          uint256 minCurrencyAmount;
12          AuctionState auctionState;
13          bool curvingPrice;
14          address successfulBidder;
15          uint256 finalPrice;
16      }
17
18      struct Auction {
19          uint id;
20          AuctionParam auctionParam;
21          mapping(uint256 => ERC3475Product) products;
22          uint256[] productIds;
23      }
24
25      function createAuction(
26          address owner,
27          uint256 startingTime,
28          uint256 duration,
29          address erc20CurrencyAddress,
30          uint256 maxCurrencyAmount,
31          uint256 minCurrencyAmount,
32          bool curvingPrice
33      ) external onlyExchange;
34  }
```

The code cited above shows an auction object. It's a package of any given number of bond object bounded together, as long as they are from the same Debond contract. For example, a user can bound several high risk sub-prime bonds with some stable prime bond. Using this method, the user is creating a new derivative product.

6.2.2 Nash equilibrium: Bond price calculation

Using dutch auction method, Debond DEX implements Nash equilibrium[10] to allow the market to negotiate the right price of the bond. The Debond DEX is built for any ERC-3475 bond, and uses DBIT as the only settlement currency. Because the index nature of DBIT, the price of DBIT to USD will be less volatile then other ERC-20 tokens. All the ongoing orders and the pledged asset behind those bonds will assure that the DBIT will reflect the weighted average performance of Debond and the DEFI market in question.

let us consider that $P_{equilibrium}(i)$ here be a set of all possible price ranges for bidder i. Which $i = 1, 2, 3 \ldots N$ from the first bidder to the last one. Let us suppose that each bidder knows the set of possible price ranges of

each other player. With $p^* = (p_i^*, p_{-i}^*)$ be a set of possible price ranges for each bidder; knowing that the $n-1$ sets of prices mean that all other bidder except i. Let $u_i(s_i, s_{-i}^*)$ be the auctioneer's payoff as a function of knowing each other player's set of prices. The set of price p^* is a Nash equilibrium if:

$$u_i(p_i^*, p_{-i}^*) \geq u_i(p_i, p_{-i}^*) \text{ of each } p_i \in P_{equilibrium}(i)$$

As a set of Debond objects is being auctioned, the auctioneer sets a starting price p_{start} and a ending price p_{end}, remained that p_{start}>p_{end}

When an auction starts,

$$T_{start} := T_l$$

Auctioneer set the duration ΔT of the auction at the start of an auction. We know

$$T_{end} := T_{start} + \Delta T$$

Therefor the change of price in time can be expressed in a linger function:

$$c = \frac{\Delta P}{\Delta T}$$

$$\begin{cases} T \in [T_{start}, T_{end}], \\ P_T = f'(T) = P_{start} - (P_{start} - P_{end})\dfrac{T - T_{start}}{\Delta T} \end{cases}$$

With this in mind, the real closing price of a set of Debond is:

$$P_{real} = \begin{cases} P_T, if P_T \leq max P_{equilibrium}(i) \\ max P_{equilibrium}(i), if P_T > max P_{equilibrium}(i) \end{cases}$$

6.3 DBIT and DGOV Bond

Both DBIT and DGOV bonds are zero coupon bond. In which the face value is repaid at the time of maturity. That definition assumes a positive time value of money. It does not make periodic interest payments or have so-called coupon.

There are buying and staking method and they both use APM and CDP features, which allows to add liquidity in the virtual pool using only one type of token. The APM system will use the CDP function to automatically mint the needed amount of DBIT to be used to pair up with the staking asset. The difference between buying and staking is that the buying method actually mints new DBIT according to the value of the pledged asset. But the staking method uses the function to transfer the according amount of DBIT from the DBIT LP to the investor.

At the redemption of the DBIT bond, investors get the amount of DBIT bonds redeemed.The graph below shows the common procedure of buying a DBIT bond. The interest and principal will be paid off by DBIT.

6.3.1 Market types

We define here, the interest payment of the redemption of DBIT bonds in the first market. As the DBIT earned from selling bonds or borrowing loans is the second market.

- **First market:** Buying method is the method of using a digital asset to buy the bond. This means that the principle will be paid back in the form of settlement token, that is, at redemption, the buyer receive the principle and the interests in DBIT. Where the staking method uses a digital asset as a staking collateral to get the bond. In that case, the staking asset is transferred back to the investor at redemption. The price of Debond in the first market is defined by CDP method introduced earlier in this article.
- **Secondary market:** On a secondary market, the price of the derivatives, typically bond is heavily influenced by the Pull to Par effect. In which the price of a bond converges to face value as time passes. At maturity the price of a debt instrument in good standing should equal its par (or face value). It results from the difference between market interest rate and the nominal yield on the bond.

6.3.2 Classes

Each bond class is a solidity Object that defines the bond properties like its id, symbol, the bond settlement token, etc. The class also defines the bond interest type, that is the floating or fixed rate interest. One of the class properties, its **nonce**, defines properties related to the bond supply, maturity period, the issuance date, the liquidity of the bond settlement token, etc. Debond allows the creation of new nonce every 24 hours, this means that bonds issued same day have all same nonce, in that case, the issuance day, one of the nonce properties can be used to distinguish two bonds.

```
1 pragma solidity 0.8.14;
2
3 contract DebondERC3475 {
4     struct Class {
```

```
5           uint256 id;
6           bool exists;
7           string symbol;
8           uint256[] infos;
9           IDebondBond.InterestRateType interestRateType;
10          address tokenAddress;
11          uint256 periodTimestamp;
12          mapping(address => mapping(uint256 => bool))
                noncesPerAddress;
13          mapping(address => uint256[]) noncesPerAddressArray;
14          mapping(address => mapping(address => bool))
                operatorApprovals;
15          uint256[] nonceIds;
16          mapping(uint256 => Nonce) nonces;
17          uint256 lastNonceIdCreated;
18          uint256 lastNonceIdCreatedTimestamp;
19      }
20  }
```

The face interest rate can be modified by the governance contract. In terms of the interest calculation method, there are 2 types and in total 10 classes of Debond:

- **Floating rate bond:** It's a high risk subprime bond. Though the face interest rate is higher than the fixed rate bond, the exact maturity date varies according to the liquidity flow of the pledged asset. Meaning that in terms of the annual interest rate, the IR is variable. During a crisis, the floating rate bond holder may suffer from impermanent loss.
- **Fixed rate bond:** It's a low risk bond. Investing in such bonds will guarantee the face interest rate and safeguard the pledged asset from impermanent loss. The face interest is determined on the date of the purchase of the bond. An algorithm will consider the ratio of floating rate, fixed rate bonds and several other factors, and gives a deterministic face IR. Once the bond is issued, the face interest rate will never be changed. Fixed rate bond holders have also a higher priority of creditors. During a crisis, their pledged asset will be safeguarded and returned properly.

Each class is a `constant int256`. In the case of DBIT and DGOV bond, there are minimum 6 digits from the right to left.

- The fist three digits represent the maturity period. Defined by:

$$T_{\text{maturity}}(c, n) - T_{\text{start}}(c, n)$$

- The forth digit represents the type of the bond. By 0=`floating_rate`, by 1=`fix_rate`.
- The fifth digit represents the issuing method. By 0=`buyBondFromToken()`, by 1=`stakingForBondFromToken()`.
- The sixth digit and after represents the i-th settlement token t_i.

Classes

Type & Maturity	Class(c)	Nonce (N)	Nominal IR	Preferred∗
Floating rate 0.5 year	$t_i, m, 0, 000$	(1,5)	ι_{float}	2nd
Fix rate 0.5 year	$t_i, m, 1, 000$	(1,5)	ι_{fix}	1st

∗Preferred Creditor is the order in which the creditor has the priority of the claim, when the debtor don't have enough fund to repay the all the debt.

6.3.3 Nonce

In DBIT and DGOV bond, nonce marks the difference in maturity date. For example, the nonce `15` represents the maturity date in 1st December 2021, nonce `16` represents the maturity date in 2nd December 2021. The interval of two bond is defined by T_{epoch}. By a sequence of nonce $\langle n_i \rangle_{i=1}^{N}$ = mean that the investor will receive bonds of N different maturity date.

When purchasing the bond from first market, investor can choice either one maturity date or multiple. In the case of multiple maturity dates, the total nominal interest rate of an investment is the sum of the nominal interest rate of every bond nonce. In the case of one maturity date, the protocol will add together the nominal interest rate of each possible nonce. This mechanism is described in detail in the section 4.4.4 Interest rate of this article.

If the investor choice multiple maturity dates, he will receive bonds with N nonce $\langle n_i \rangle_{i=1}^{N} = \langle n_1, n_2, n_3, \cdots, n_N \rangle$. The number of nonce N is based on the maturity time. We arbitrarily fixed the multiple nonce of 0.5 year bond to 5. For the rest of maturity time is quadratic of the times of maturity time in year Y over 0.5. N must be an integer $\mathbb{Z}$.

$$\begin{cases} \langle n_i \rangle_{i=1}^{N} = \langle n_1, n_2, n_3, \cdots, a_N \rangle, \\ N = 4 + \sqrt{\dfrac{Y}{0.5}}, \\ N := \lceil N \rceil = \min\{x \in \mathbb{Z} \mid N \leq x\} \end{cases}$$

The sequence is based on several factor listed below:
N = the length of sequence, described above.
n_l = the nonce of time now T_l, defined by `time.now`.
$\Delta n_{\text{maturity}}$ = difference in length from the issuing time to the maturity date.
F_n= the n-th number of a Fibonacci sequence.
x= a variable measuring the length of each Fibonacci unit in $\Delta n_{\text{maturity}}$

$$\langle n_i \rangle_{i=1}^{N} = \begin{cases} x = \dfrac{\Delta n_{\text{maturity}}}{F_{N+1}}, \\ \langle (n_l + F_2 x),\ (n_1 + F_3 x),\ (n_l + F_4 x),\ \cdots,\ (n_l + F_{i+1} x) \rangle \end{cases}$$

6.3.4 Interest rate

There are several factors involved in the calculation of the real interest rate ι_{real}.

$\iota_T = \iota_{weighted}(T)$ the instantaneous weighted nominal short rate at time T
$p_{determined}(T - \Delta T)$ = the CDP price when the bond is issued
p_T = value (in USD) of the settlement token received at maturity date T
σ_T = instant short rate volatility
W_T = a standard Brownian motion under a risk-neutral probability measure;
dW_T= its differential.

First we calculate the instantaneous nominal short rate ι at time T. This nominal interest rate is calculated from benchmark IR and the ratio of floating rate bond to fixed rate bond of the same settlement token. Then we consider the difference in the price of settlement token over difference in time dT. Finally we add the random walk factor W_T.

$$d\iota_{real}(T) = [\frac{p_T}{p_{determined}(T - \Delta T)}(1 + r_T) - 1]dT + \sigma_T\, dW_T$$

The benchmark interest rate is the interest defined by `IR.benchmarkIR` and can be updated by `governanceContract.updateBenchmarkIR()`. It's the basic interest rate from which we can calculate the floating rate interest ι_{float} and fixed rate interest ι_{fix}.

$$2\iota_{benchmark} = \iota_{float} + \iota_{fix}$$

However this changes is not entirely proportional:

$$\frac{\iota_{float}}{\iota_{fix}} = 1 \quad \text{only when} \quad \frac{B_{float}(t)}{B_{fix}(t)} = 4$$

For $0 < c < 1$, define:

$$\varphi_c(x) = \frac{2^{-1/(1-c)x}}{2^{-1/(1-c)x} + 2^{-1/c(1-x)}}$$

This$\varphi_c : [0,1] \to [0,1]$ is bijective, sigmoid (sigma-function like) and infinitely differentiable. Particularly, it satisfies the following:

$$\begin{cases} \varphi_c(0) = 0, \\ \varphi_c(c) = \dfrac{1}{2}, \\ \varphi_c(1) = 1 \end{cases}$$

Figure 4 shows the variation of the sigmoid function φ in terms of the ratio $x = B_{fix}/(B_{fix} + B_{float})$ with a fixed parameter $c = 0.2$. For $x <= c/2$, φ is almost constant, this means that when $B_{fix} \to 0$ or when $B_{float} \gg B_{fix}$, the sigmoid function doesn't vary. The same behavior of the sigmoid function is observed when $x > 2c$, this is equivalent to having $B_{fix} \gg B_{float}$. For intermediate values $c/2 < x < 2c$, the sigmoid function φ increases sharply reaching nearly its maximum at $x \approx 2c$.

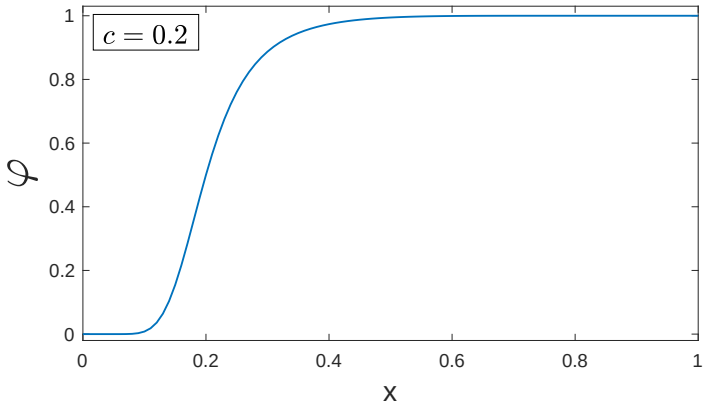

Figure 4: sigmoid function with $c = 0.2$

The interest rate $\iota_{float}(t)$ of an floating rate bond is based on benchmark IR and the ratio of floating rate bond to fixed rate bond of the same settlement token t.

$$\iota_{float}(t) = 2\iota_{benchmark} \cdot \varphi_{0.2}\left(\frac{B_{fix}(t)}{B_{float}(t) + B_{fix}(t)}\right)$$

The interest rate $\iota_{fix}(t)$ of an fix rate bond is based on benchmark IR and the ratio of floating rate bond to fixed rate bond of the same settlement token t.

$$\iota_{fix}(t) = 2\iota_{benchmark} - \iota_{float}(t)$$

Figure 5 shows the variations of both floating and fixed interest rates in terms of the ratio $x = B_{fix}/(B_{fix} + B_{float})$. The floating interest rate variations follows exactly the variations of the sigmoid function. This is expected since the floating interest rate ι_{float} is proportional to φ, the benchmark interest rate $\iota_{benchmark}$ being a constant fixed by the governance contract. For $x < 2c$ or $B_{fix} \ll B_{float}$ or simply $B_{fix} \to 0$, the fixed interest rate becomes higher than the floating interest rate. This is the case when most of the bonds on the market are floating rate bonds, in that case, the system needs to adapts the interest rates in such a way to offer higher rate for fixed rate bonds than floating one, since having more floating rate than fixed rate may lead to default payment situation. However, as it will be shown in next section, this default payment is removed by introducing an estimate redemption time which varies according to the liquidity available. On the other hand, when $x > 2c$ or $B_{fix} >> B_{float}$, the fixed rate ι_{fix} goes to zero while the floating rate ι_{float} goes to the benchmark rate $\iota_{benchmark}$. This is the case when most bonds on the market are fixed rate bonds, therefore, the floating rate can be increased to incentive investors to buy floating rate bonds. For $x < c/2$, the fixed rate goes to the benchmark rate while the floating rate goes to zero, in that case, investors buys more fixed rate bonds since the rate is at its maximum and a fixed rate would never change. When $x = c$, both floating and fixed rate become equal. This happens when $B_{float} = 4B_{fix}$, therefore, when the liquidity flow of floating rate bonds becomes four times that of fixed rate bonds, the system starts to adjust the interest rates ins such a way to have $\iota_{fix} < \iota_{float}$ in case $x > c$. The condition $B_{float} < 4B_{fix}$ is required in order to maintain a higher floating rate and lower fixed rate.

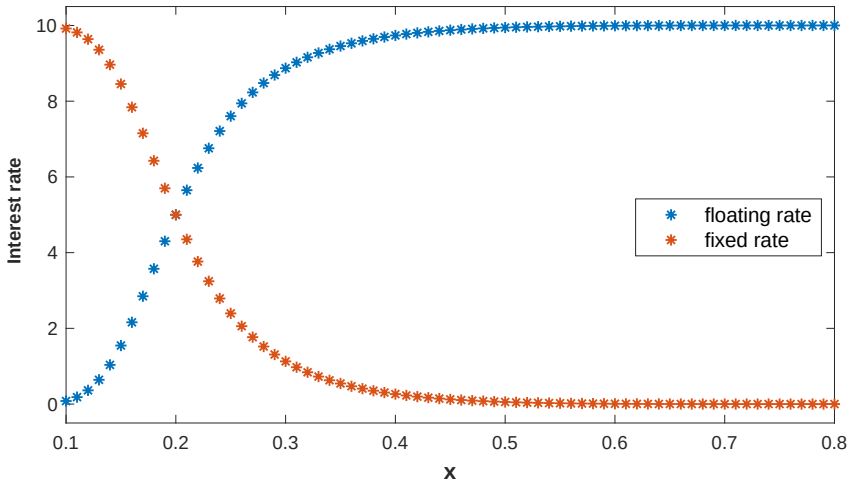

Figure 5: floating and fixed interest rates

Finally we can calculate the weighted interest rate $\iota_{weighted}$:

$$\begin{cases} \iota_{weighted} = \iota(1 - 1/2^{\kappa}), \\ \kappa = \lceil \frac{\Gamma}{\Lambda} \rceil, \\ \Gamma := 1, \\ \Lambda = 1 - \frac{\mu_{total}}{\mu_{month}} \end{cases}$$

with:
μ_{month}= the average total balances of all the nonce in last month $\frac{T_{month}}{\Delta T_{epoch}}$.
μ_{total}= the average total balances of all the nonce from the fist nonce 0 to the last n_l.

6.3.5 Bonds maturity

The fixed rate bond has only one redemption condition, the maturity time $T_{maturity}$ on the bond. After this mandatory redemption date, the bond will be redeemed without any other questions. Fixed rate bonds are matured once:

$$T_{maturity}(n) \leq T_l$$

Since floating rate bonds have an estimated redemption date $T_{estimate}(c, n)$, which is not fixed. Thus the dT factor is not fixed. Although the IR of fixed rate bond is constant after purchase, the IR may vary according to the different ratio between the fixed rate bond and the Floating rate bond.

$$\begin{cases} B_{\text{sum}}(n)(1 + \iota_{benchmark}) \leq B_{\text{sum}}(n_l), \\ and, T_{maturity}(n) \leq T_l \end{cases}$$

$T_{estimate}(n)$ is the estimated redemption time based on the average liquidity flow over last month μ_{month}, and the additional liquidity needed.

$$T_{maturity}(n) + \frac{B_{\text{sum}}(n)(1 + \iota_{benchmark}) - B_{\text{sum}}(n_l)}{\mu_{month}} T_{epoch}$$

6.3.6 Preferred Creditor

When the redemption crises appears, the collateralised liquidity is not enough to fulfil all the requirement of liquidation. In that case, a mechanism called

Preferred Creditor will take place. The bond with longer redemption period will have a priority of creditors.This means if the pledged asset is no longer enough to pay all the debt during a period, the pledged asset will be firstly used to repay the Preferred Creditor.

Bondholder can choice to liquidate his or her bond before the redemption date. But by doing so, the bondholder will suffer from different degree of losses based on their preferred creditor order.

A crisis by definition is when:

$$B_{sum}(n)(1 + \iota_{benchmark}) > B_{sum}(n_l)$$

There are 10 different classes of Debond (see section 4.4.2 Classes for more information). In this scenario, first we calculate the deficit D of the bond product,

$$D = B_{sum}(n)(1 + \iota_{benchmark}) - B_{sum}(n_l)$$

Then we attribute the deficit to each bond class from the less preferred to the most preferred. The preference order is defined by:

$$O_{preferred} = \langle o_i \rangle_{i=1}^{C} = \langle 1,\, 2,\, 3,\, \cdots,\, C \rangle$$

According to this order we can calculate the rate of how much remains $R_{rate}(o_i)$.

$$\begin{cases} \sum_{k=1}^{C-1} k = 45, if C = 10, \\ R_{rate}(o_1) = 1 - 0, \\ R_{rate}(o_2) = 1 - 1(\frac{1}{45}), \\ R_{rate}(o_3) = 1 - 2(\frac{1}{45}), \\ R_{rate}(o_4) = 1 - 3(\frac{1}{45}), \\ \quad ... \\ R_{rate}(o_C) = 1 - (C-1)(\frac{1}{45}) \end{cases}$$

Finally we can calculate the remaining value of the bonds R_{o_i} after the

bankruptcy .

$$\begin{cases} \frac{R_{o_1} = R_{rate}(o_1)B_{total}(o_1)}{1+\iota_{weighted}(o_1)} \\ \frac{R_{o_2} = R_{rate}(o_2)[\frac{\sum_{i=2}^{C} B_{total}(o_i)}{B_{sum}(n_l)}]B_{total}(o_2)}{1+\iota_{weighted}(o_2)} \\ \frac{R_{o_3} = R_{rate}(o_3)[\frac{\sum_{i=3}^{C} B_{total}(o_i)}{B_{sum}(n_l)}]B_{total}(o_3)}{1+\iota_{weighted}(o_3)} \\ \frac{R_{o_4} = R_{rate}(o_4)[\frac{\sum_{i=4}^{C} B_{total}(o_i)}{B_{sum}(n_l)}]B_{total}(o_4)}{1+\iota_{weighted}(o_4)} \\ ... \\ \frac{R_{o_C} = R_{rate}(o_C)[\frac{\sum_{i=C}^{C} B_{total}(o_i)}{B_{sum}(n_l)}]B_{total}(o_C)}{1+\iota_{weighted}(o_C)} \end{cases}$$

6.4 Securitised loan

The creditor of a securitized loan is no longer a smart contract, but a bond purchaser. The debtor has to first pledge his/her digital assets (ERC-20, ERC-721, ERC-3475 etc.) to a smart contract, which then issues the loan bond representing the securitised loan. Based on the creditor's settings of the amount of the loan, repayment method, interest rate, due date, etc. These bonds are going to be placed on the secondary market and traded using the Dutch auction method. If a prospective debtor believes that the bond is set up in a rational manner, he/she will purchase the bond. Upon the completion of the transaction, the creditor receives the loan and the debtor obtains the bond of the collateralised assets. If the creditor repays the loan before the repayment date, he or she can reclaim his or her pledged asset. If not, the bond will be redeemable after the marked repayment date by the bondholder. Until then, the debtor can split or pack the bond into packages and sell them on the secondary market (Debond DEX). The creditor agreed to repay the principal and interests by the specified date in accordance with the established rules. After all the principal and interests are repaid to the creditor(s), the relevant bond of the pledged asset will be invalidated.

6.4.1 Loan contract creation

The method below shows how a loan order is created. By using a custom structure called ERC20LOAN, a user can create their own loan agreement. The object of a loan is the custom structure demonstrated below. A user can create such a structured object by inputting parameters into the EVM smart contract.

```
struct ERC20LOAN {
bool auctionStatus;
address seller;
uint256 startingPrice;
uint256 endingPrice;
uint256 auctionTimestamp;
uint256 auctionDuration;
address bondAddress;
uint256 interestRate;
uint256 loanDuration;
uint256[] bondClass;
uint256[] bondNonce;
uint256[] bondAmount;
}
```

```
function createERC20Loan (ERC20LOAN memory _ERC20Loan) public returns(bool)
require(contract_is_active=true,"contract is not active");
_ERC20Loan.auctionTimestamp=now;
require(_addERC20Loan(_ERC20Loan)==true,"can't create more auction");
require(IERC3475(bond_contract).writeInfo(_ERC20Loan));
require(_addPledgedERC20Asset(_ERC20Loan)==true,"can't move to custody");
return(true);
}
```

6.4.2 P2P securitised loan

Securitised loan has several advantages over traditional DEFI lending protocol, as shown below:

- **No enforced liquidation:** There is a risk of liquidation in the traditional DEFI lending protocol [14], since the collateral will be liquidated if their value falls below a certain level. In contrast, given that the securitised loans can be traded in the secondary market, their price will also decrease along with the collateral. This allows the market

to find the most appropriate price for the bonds without resorting to liquidation.

- **Customized loan agreement:** Interest rate, loan amount, repayment method and date are customizable. Unlike traditional loan agreements, securitised loans allow the creditor to set their own interest rate, loan amount, and repayment schedule. The creditors, debtors and the market will use the game theory to find a If the creditor's settings are reasonable.

- **Transferable securitised loan:** Securitised loans can be divided and assigned to others in the form of bond, turning the loan into a type of speculative investment. The fluctuation of the bond price in the secondary market can therefore create a new speculative market.

- **Higher usage rate of the pledged assets:** Traditional lending protocols often ask for over-collateralization while offering a loan of roughly half of the collateral value. This percentage is fixed in most protocols and cannot be adjusted according to the market conditions accordingly. In the securitised loan system, the creditors and the debtor are engaged in a game theory scenario where they try to find a Nash equilibrium. This allows the collateral rate to be aligned with the market supply and demand.

- **NFT pledged lending:** conventional lending agreements cannot accurately estimate the value of NFT collateral through an oracal. Because of the non-fungible nature of NFTs, NFT pledged loans have been inefficient. In a securitised loan, it is the creditor who appraises the NFTs and competes with each other to determine a best valuation before giving the loan to the creditor.

- **Loan bond splitting and bundling:** both NFT bonds and ERC20 token bonds can be split or bundled for trading. Riskier bonds can be bundled together for trading in the secondary market at a higher value. The same can be done to NFT bonds to spread the debtor's risk.

6.5 Debond Derivatives

A part from being used as a tool of issuing bond, Debond can also be used to create all kinds of financial derivatives. This include anything from option, future, swaps, etc..

Debond can also be implemented for off-chain derivatives. By projecting the securitised contract to the block-chain network and using oracle machines

to feed the latest market data [11], Debond can also be used to facilitate the transaction of off-chin bonds. This application requires an audited entity as the portal between the Debond and off-chin derivatives. The use cases of the Debond derivatives are:

- **Hedging:** Hedge or to mitigate investment risk, by entering into a derivative contract whose value moves in the opposite direction of their position and cancels part of the potential lost.
- **Option:** Create options with the value of the derivative is linked to a predetermined condition or event (e.g., the underlying asset reaching a specific price level).
- **Leveraging:** Provide leverage, such that a small movement in the underlying value can cause a large difference in the value of the derivative.
- **Speculation:** Make profit from speculating the market movements. (e.g. moves in a given direction, stays in or out of a specified range, reaches a certain level).
- **Arbitraging:** Allowing a riskless profit by simultaneously entering into transactions into two or more markets.

Some of the common variants of derivative contracts are as follows:

6.5.1 Forwards

Tailored contract between two parties, where payment takes place at a specific time in the future at today's pre-determined price.

6.5.2 Futures

Contracts to buy or sell an asset on a future date at a price specified today. A futures contract differs from a forward contract in that the futures contract is a standardized contract written by a clearing house that operates an exchange where the contract can be bought and sold; the forward contract is a non-standardized contract written by the parties themselves.

6.5.3 Options

Contracts that give the owner the right, but not the obligation, to buy (in the case of a call option) or sell (in the case of a put option) an asset. The price at which the sale takes place is known as the strike price, and is specified at the time the parties enter into the option. The option contract also specifies a maturity date. In the case of a European option, the owner has the right to require the sale to take place on (but not before) the maturity date; in the case of an American option, the owner can require the sale to take place at any time up to the maturity date. If the owner of the contract exercises this right, the counter-party has the obligation to carry out the transaction. Options are of two types: call option and put option. The buyer of a call option has a right to buy a certain quantity of the underlying asset, at a specified price on or before a given date in the future, but he has no obligation to carry out this right. Similarly, the buyer of a put option has the right to sell a certain quantity of an underlying asset, at a specified price on or before a given date in the future, but he has no obligation to carry out this right.

6.5.4 Binary options

Contracts that provide the owner with an all-or-nothing profit profile.

6.5.5 Warrants

Apart from the commonly used short-dated options which have a maximum maturity period of one year, there exist certain long-dated options as well, known as warrants. These are generally traded over the counter.

6.5.6 Swaps

Apart from the commonly used short-dated options which have a maximum maturity period of one year, there exist certain long-dated options as well, known as warrants. These are generally traded over the counter.

7 Governance

Although the idea of a decentralised governance has been mentioned often, the general concept is that the community will be come together forming a kind of "decentralized autonomous organization" or DAO. It is a virtual entity that has a certain weight over the voice of an address. The shareholders behind this address can then express their point of view through voting. In practice, any manipulation of the governance is through a centralized process, no obligation being imposed to the possessor of the master key owner. Platform such as snapshot[17] serves only as a vote counting service which imposes no obligation on the execution of the proposal. In the same sense, decentralised bond governance platforms allow the community to participate in the governance of the DEFI project. The members would collectively decide on how the project should invest its funds or what the project should do next.

7.1 Proposal

Decentralised bond governance platform (DBGP) uses an on-chain program called "proposal" as the object of a voting procedure. This means that the proposal in question is in the form of an open-sourced smart contract. The inputs of every single method called from the proposal contract to the governance contract will be written in the open-sourced contract. Meaning that an underlying obligation is imposed on the governance system to execute the proposal as expressed in the content of code. These methods can be used for spending the funds, issuing allocations, updating a smart contract, etc. We can see in the code below, the access to a governance function is given to a passed proposal contract, which is defined by a proposal class `_class` and a proposal nonce `_nonce`.

```
function changeTeamAllocation(
    uint128 _class,
    uint128 _nonce,
    address _to,
    uint256 _newAmount
) public returns(bool) {
    require(_class <= 1, "Gov: class not valid");
    require(
        checkProposal(_class, _nonce) == true,
        "Gov: proposal not valid"
    );
```

```
    require(
        msg.sender == proposal[_class][_nonce].proposalAddress,
        "Gov: not proposal owner"
    );
    proposal[_class][_nonce].executionInterval -= 1;
    require(
        distributedPPM - allocationPPM + _newAmount <= budgetPPM
    );

    distributedPPM += _newAmount - allocationPPM;
    allocationPPM = _newAmount;

    return true;
}
```

The calling of the governance function is strictly limited to an open sourced proposal contact. Here is an example of a proposal contract. In this example, the `updateBondContract()` function of the governance contract is called from the proposal contract. The new updated contract address and other parameters need to be written on the open-soured solidity code of the proposal contract.

```
  contract proposal {
    address public gov_address;

    constructor(address dev_address) public {
      gov_address = dev_address;
    }
  }

  function updateBondContract(
    uint256 _class,
    uint256 _nonce,
    address _newBondAddress
  ) public returns(bool) {
    require(
      IGovernance(gov_address).updateBondContract(
        _class, _nonce, _newBondAddress
      ) = true
    );
  }
```

This system can be used ranges from salaries and investments to even more

complex mechanisms such as an option, an algorithmic speculation bot. Decentralised governance platform can be used to apply any actions which can be expressed in mathematical expression. We believe that the idea of a decentralised governance is relying on the obligation to execute the collective will of the community. On this matter, most of the previous governance system has only the features of a vote counter.

7.2 D/Capital

The term D/Capital essentially replicates the legal trappings of a venture capital, an investment bank or a nonprofitable organisation. Through D/-Capital, Debond can invest or incubate any on-chain project. An Estonia-registered legal entity, called Debond Protocol [18] will be used as a gateway to the off-chain investment. The banking account of Debond Protocol will be audited by an accounting firm. All the investment decisions of the D/-Capital, whether on-chain or off-chain, all need to be approved by voting on the DBGP.

Along these lines, D/Capital can use the pledged asset of Debond to build up LP for another DEFI project, invest in their seed phase, etc..

D/Capital will work like an investment bank, it borrows money from the investors and uses their investment to fund startups, buying stocks and speculating on the financial market. Using the cryptographic block-chain technology for the transparency enforcement, D/Capital will be accessible and trustworthy, like or even better than an traditional investment bank.

7.3 Voting

By pledging the governance token DGOV into the governance storage contract, the user will receive a specified token called Debond Vote Token (DVT). DVT can be used in the voting process of a proposal contract. Like any Debond product, redemption conditions are applied on DVT. In order to redeem one's DGOV, the DVT holder must wait until a specified maturity date. After this specified redemption date, DVT can be redeemed and the bondholders will receive the DGOV deposited and a small percent of DBIT as the reward of their participation. An amount of DVT worth the amount of DGOV unstaked by the user is then burned.

7.4 DBIT rewards after redemption of DVT

In order for users to vote for a proposal, they need to buy and stake DGOV tokens in a dedicated contract. Users who stake DGOV tokens receive DVTs that are required to vote for proposals and to redeem their DGOV tokens at the end of staking. All users who voted for a proposal are rewarded with some DBITs according to the number of DVT they submitted and the day when they voted, the more the DVTs used to vote, the more the DBIT reward received, and the earlier they vote, the more the DBIT reward. Furthermore, DBIT rewards are subdivided into periods of 24 hours and all users that vote at a given period share not only reward of the day they voted, but also rewards from all remaining days until the vote is over. The daily DBIT reward any user who voted for a proposal will get is given by the following formula

$$R^i_{DBIT} = (\text{DBIT reward distributed for day } i) \times \frac{\text{user amount of vote tokens submitted}}{\text{total amount of vote tokens on day } i}$$

Let's assume v to be the amount of DVTs the user submitted for the proposal, V_i the total amount of DVTs submitted for the proposal on day i and r^i_{DBIT} the DBIT reward which will be distributed for day i, then the total DBIT reward R_{DBIT} a user gets is the sum of daily rewards: $R_{DBIT} = \sum_i R^i_{DBIT}$

$$R_{DBIT} = v \sum_i \frac{r^i_{DBIT}}{V_i}$$

As an example, let's consider a proposal that distributes 30 DBITs the first voting day, 10 DBITs the second voting day and 10 DBITs the third day. Let's assume that the first day 3 users vote for the proposal, one user votes the second day and one more the last day. In total 5 users have voted for the proposal and at the end of the vote they will share 50 DBITs. Table-2 shows the scenario considered here, and by applying the formula above one can calculate the total DBIT rewarded to each user.

day-1 (30 DBIT)	day-2 (10 DBIT)	day-3 (10 DBIT)
voter-1 (10)	10	10
voter-2 (5)	5	5
voter-3 (2)	2	2
-	voter-4 (20)	20
-	-	voter-5 (10)
(17 DVT)	(37 DVT)	(47 DVT)

Table 2: user's vote and number of votes per day

- voter-1 : $R_{DBIT} = 10 \times \left[\frac{30}{17} + \frac{10}{37} + \frac{10}{47}\right] \approx 22.48$
- voter-2 : $R_{DBIT} = 5 \times \left[\frac{30}{17} + \frac{10}{37} + \frac{10}{47}\right] \approx 11.24$
- voter-3 : $R_{DBIT} = 2 \times \left[\frac{30}{17} + \frac{10}{37} + \frac{10}{47}\right] \approx 4.49$
- voter-4 : $R_{DBIT} = 20 \times \left[\frac{10}{37} + \frac{10}{47}\right] \approx 9.66$
- voter-5 : $R_{DBIT} = 10 \times \left[\frac{10}{47}\right] \approx 2.13$

	voter-1	voter-2	voter-3	voter-4	voter-5	Total
reward	22.48	11.24	4.49	9.66	2.13	50

Table 3: DBIT rewards earned by voters

8 TOKENS

Debond issues two native tokens DBIT and DGOV. In Debond system, tokens are minted by redeeming bond. User will first issue bonds by buying

or staking DBIT or DGOV, then at maturity date, the bond can be redeemed to a settlement currency DBIT or DGOV. The detailed description of this process can be found in section "4.4 DBIT and DGOV bond".

The **Debond Token** contract implements the mint of Debond tokens (DBIT and DGOV) in three cases: airdrop tokens, allocated tokens and collaterised tokens. Each mint of Debond tokens must be done from one of these three mint processes with the right caller: bank contract for collaterised supply, governance contract for allocated supply and airdrop contract for airdrop supply.

8.1 Airdrop tokens

The airdrop supply is part of the total supply allocated to users in a whitelist and must be locked for a given period before tokens can be withdrawn.

8.2 Allocated tokens

The allocated supply is part of the total supply that is allocated for some tasks like rewarding users who vote for proposals, etc.

8.3 Collaterised tokens

The collaterised supply is part of the total supply that refers to tokens that are minted by providing a collateral. This can be the case when a Debond bond is issued by proving stable coin tokens as collateral.

9 CONCLUSION

Debond protocol was originally conceived as an upgraded version of DEFI yield farming system, providing extended settings such as LP fragmentation and derivatives creation, etc.. We believe that the native use case of smart contract should be the financial derivatives. Since ERC-20 token can't fulfill this task, we propose a possible solution.

What we think is exciting about this project is that the update of the current yield farming system will bring more changes, possibilities and believers to the DEFI system, pushing DEFI to the next stage, where a more complex

economic system can be designed using the ERC-3475 bond standard. We are expecting it to go beyond the boundary of our imagination. New protocols around decentralized options, derivatives and any other form of financial product, among many existing or non-existing such concepts. It has all the potentials needed to substantially increase the efficiency and computational complexity of DEFI. Debond is an open-ended platform by design. We are looking forward to seeing a series of applications and use-cases built on top of the ERC-3475 infrastructure.

Hayek once said in an interview [19], "Freezing in its most primitive form, money was never allowed to be involved since its invention." We firmly believe that the Decentralized autonomous organisation or DAO is a possible answer to many social problems that we are facing in our current economic system. The Crypto-anarchism would allow the Debond system to grow organically as a decentralized community. By assimilating developers and investors alike, the Debond Eco-system will be evolving to a point that the collective intelligence will find an optimal solution to any problems that we will be facing.

References

[1] Ethreum Whitepaper, https://ethereum.org/en/whitepaper/

[2] EIP-20:Token Standard, https://eips.ethereum.org/EIPS/eip-20/

[3] EIP-3475:Multiple Callable Bonds, https://eips.ethereum.org/EIPS/eip-3475

[4] Volckart, Oliver (1997). "Early beginnings of the quantity theory of money and their context in Polish and Prussian monetary policies, c. 1520–1550". The Economic History Review. Wiley-Blackwell. 50 (3): 430–49.

[5] O'Sullivan, Arthur; Sheffrin, Steven M. (2004). Economics: Principles in action. Upper Saddle River, New Jersey 07458: Prentice Hall. pp. 197, 507. ISBN 0-13-063085-3

[6] Fisher Irving, The Purchasing Power of Money, http://public.econ.duke.edu/ kdh9/Courses/Graduate

[7] Bitcoin: A Peer-to-Peer Electronic Cash System, https://bitcoin.org/bitcoin.pdf

[8] Metamask Documentation, https://docs.metamask.io/guide/

[9] Documentation of Uniswap v1, https://docs.uniswap.org/protocol/V1/introduction

[10] Nash, John (1951) "Non-Cooperative Games" The Annals of Mathematics 54(2):286-295.

[11] Christos Papadimitriou. Computational Complexity. Addison-Wesley, 1994. Section 14.3: Oracles, pp. 339–343.

[12] Liebowitz, Martin L. and Homer, Sidney (29 April 2013). Inside the Yield Book (third ed.). Hoboken NJ: John Wiley & Sons, Inc. p. 117. ISBN 978-1-118-39013-9.

[13] The Maker Protocol: MakerDAO's Multi-Collateral Dai (MCD) System, https://makerdao.com/en/whitepaper/#notes

[14] AAVE protocol whitepaper, https://github.com/aave/aave-protocol/blob/master/docs/Aave_Protocol_Whitepaper_v1_0.pdf

[15] Kelly, Frank K. and Weight, David J. (1997). The Handbook of Fixed Income Securities (Frank J. Fabozzi, Editor). New York: McGraw-Hill. p. 129-130. ISBN 0-7863-1095-2.

[16] Pearson, Karl (1894). "On the dissection of asymmetrical frequency curves". Philosophical Transactions of the Royal Society A. 185: 71–110. Bibcode:1894RSPTA.185...71P. doi:10.1098/rsta.1894.0003

[17] SNAPSHOT documentation https://github.com/snapshot-labs/snapshot

[18] Company registration information https://www.infoproff.com/en/companies/search/

[19] An Interview with F. A. Hayek (1984) https://www.youtube.com/watch?v=s-k_Fc63tZI