CUNY School of Professional Studies

Master of Science in Data Science

Cryptobonds

Samuel Reeves
Candidate

Prof. Nisran Khansari
Advisor

Abstract

A general purpose blockchain can host smart contracts that can function as fixed income assets. The assets created here calculate a payment amount based on a number of predetermined factors and send it to the owner when they trade the asset to another person. All remaining value stays locked in it to the new owner. This is called a cryptobond. Valuing these assets is non-trivial.

Contents

1 Introduction

1.1 What is a cryptobond?

A cryptobond is a smart contract that functions completely independently on a general-purpose blockchain as a fixed-income asset.

Smart contracts on the Ethereum Virtual Machine, the framework hosted on many common blockchains, have no "daemon" running in the background which can wake up and make transactions on their own. They are written in a high-level language such as Solidity or Vyper, and they compile down to binaries which can be pushed to the network from a full node. At the time a call is made to the smart contract, defined by the functions available in the binary, the smart contract can perform an operation such as transferring ownership, making a payment, or giving a readout of its properties. It is possible to create events to emit log statements at the time an action is performed, however, it is always necessary to call a function from an outside source.

For this reason, these cryptobonds will have a fixed end time $t_{\max}$, and a start time t_0 determined by the timestamp from the moment the asset is traded from the initial creator to the first owner. Every time the bond is traded to a new owner, it will return an amount of its stored value to the owner it is coming from. This value will be calculated at the time of the trade, based on the distribution described in the fixed contract.

In the world of fixed income, risk is equivalent to the return:

$$\text{Risk} = \frac{\text{Return}_{\text{net}} - \text{Principle}}{\text{Principle}}$$

Carl Icahn suggests that we should never buy junk bonds. He is referring to fixed income assets with very high rates of return, perhaps from failing companies or governments on the verge of collapse. There is a considerably greater chance that this class of bond is not repaid at the time of maturity. This uncertainty is difficult to hedge against, and his opinion makes some sense. Very high risk, very high return.

However, in the case of cryptobonds, a payout is guaranteed as long as there are a couple of nodes left to hash blocks of the public transaction history. It's so certain, and risk is so low, that it costs a fee to the node to include a transaction in the timeline. This means the overall Return will always necessarily be worth less than V_0.

If the rates of exchange for the underlying cryptocurrency are static, it's beneficial to treat it like a vault and to be one of the first sellers. However, if they are in constant flux, there is ample opportunity to disagree over the value of all payments X at $t < t_{\max}$.

1.2 Similar Financial Assets

There are many other types of assets that are traded on normal exchanges by normal institutions which function similarly in terms of valuation or returns. Although somewhat counter-intuitive, each of these asset classes is has shared theoretical properties

1.2.1 Zero-Coupon Bonds

1.2.2 High-Risk Assets

1.2.3 Low-Risk Assets

1.2.4 Negative-interest Loans

2 Asset Payment Structure

Recall that our coupon amount is defined by a closed form distribution stretching from the time of the first trade to a predetermined end time. The timestamp at the end of the period when the bond matures, we will call t_{M}. The initial timestamp will be called t_0, and time t_n is the timestamp at the time of the n^{th} transaction.

We will call the total initial value V_0. After t_M, trades are voided, and any remaining value returns to the wallet of the owner of the bond. Then, the bond is burned; it removes itself from the blockchain and is gone forever.

The generalized structure is this:

$$t_\mu = \text{floor}(\frac{t_M - t_0}{2})$$

$$V_0 = \sum_{n=1}^{\infty} w_n V_0$$

$$= \sum_{n=1}^{\infty} x_n + g_n + \varepsilon_n$$

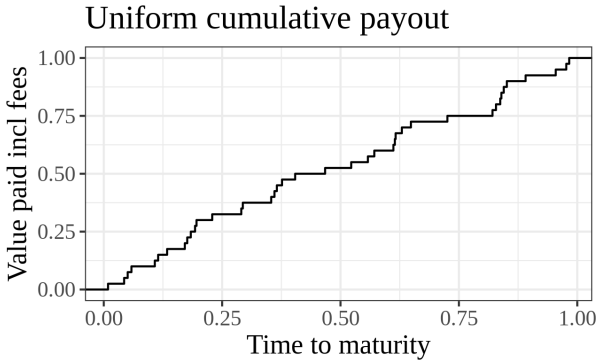
Where, x_n is the payment that arrives at the seller's wallet after gas fee g_n. The weighting coefficient w_n gives the bond the shape of the curve of its payout structure.

2.1 Uniform Distribution

For a uniformly distributed payout structure, w_n is just the percentage of time total time relative to maturity that the bond was owned by the seller. There is no curve, and assuming the transaction costs G are negligible, the magnitude of payouts is just a flat line from t_0 to t_M.

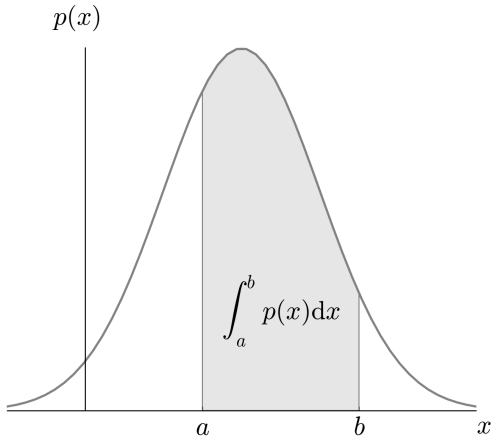
$$w_n = \frac{t_n - t_{n-1}}{t_M - t_0}$$

with a uniform distribution, $\varepsilon = 0$.

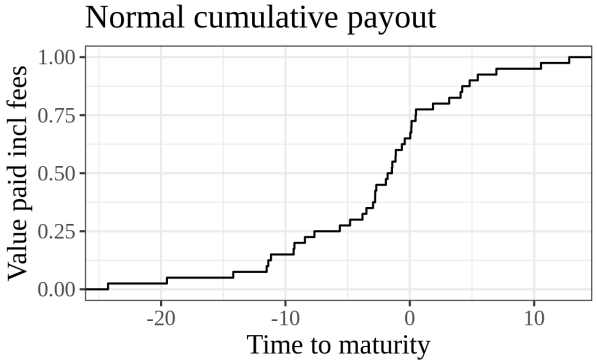


2.2 Normal Distribution

Another way to define a cryptobond payment scheme could be to weight the payments based on a curve. In the discrete uniform case, the weight of the payout is a function of the area under a uniform normal curve for the time period the bond has been held relative to the total time $t_0 \to t_M$.



The time period is weighted continuously from beginning to end by taking the P-value of the Z score for the time period. If you buy the asset close to t_0 and you sell it soon after, the payout will be very small, but the sale price for the asset will be considerably large because the majority of the principle is intact. If you own it and sell it closed to t_μ, the payouts will be larger for the same amount of time, and the sale price will be considerably smaller with only the right tail of the distribution left to pay out.



There are two cases. The weighting function is defined piecewise for areas where t_n and t_{n-1} fall on either side of $t_m u$, the second part of the function describes an area that falls completely on one side of $t_m u$, inclusive. These functions are dependent on the practical method of approximating the Cumulative Distribution Function.

$$w_n = \begin{cases} 1 - P(z_{n-1}) - P(z_n) & t_{n-1} < t_\mu < t_n \\ \left|P(z_n) - P(z_{n-1})\right| & \text{otherwise} \end{cases}$$

We arrive at a Z-score in a familiar way,

$$\sigma = \frac{t_M - t_0}{\sqrt{12}}$$

$$z_n = \left|\frac{t - t_\mu}{\sigma}\right|$$

Because of the limitations of the EVM, it is necessary to use a numerical approximation for the P value which will determine the actual amount of payment x_t. We are unable to use trigonometric or calculus functions. Any chosen approximation with these constraints will be a compromise among low error rate across the spectrum, easily definable constants, and ease of computation.

This presents a theoretical problem, but not a practical one. Time flows in one direction only, and these values for the transaction amounts are calculated at the time the transaction is initiated.

We can use a simple algorithm such as this algorithm published by Polya in 1945,

$$P(z) \approx \frac{1}{2}(1 + \sqrt{1 - e^{\frac{-2z^2}{\pi}}})$$

or a newer one such as this one from Lin 1990,

$$y = 4.2\pi\frac{z}{9 - z}, \qquad 0 \leq z \leq 9$$

$$P(z) \approx 1 - \frac{1}{1 + e^y}$$

or this one published by Winitzki in 2008

$$P(z) \approx \frac{1}{2}\left[1 + \left(1 - \exp\frac{-(\frac{z^2}{2})(\frac{4}{\pi} + 0.147\frac{z^2}{2}}{1 + 0.147\frac{z^2}{2}}\right)^{\frac{1}{2}}\right]$$

It's unclear which is the best compromise. Storage and computing power can be expensive. I suspect that the choice of CDF approximator can be optimized depending on the timespan, V_0, and n.

With a reasonably low n or and a very large timespan:

$$\lim_{V_0 \to \infty} \frac{G}{V_0} = 0$$

3 The Smart Contract

In order to engineer a smart contract, you must start by writing a contract which describes all relevant logic (and nothing else!) in a high-level language. Most prefer Solidity. It is verbose, highly developed already, and for web-coders it is very familiar. It borrows its syntax from Javascript.

If a person or group makes a very useful smart contract, they submit it as an Ethereum Improvement Proposal (EIP), and if it goes through a rigorous vetting process and becomes accepted, it becomes an Ethereum Request for Comment (ERC) with a number indicating when it was accepted.

We have based cryptobonds on ERC-4626, the Tokenized Vault Standard. There are many features removed to reduce the cost of deploying it and for security. The added functions are for computing the magnitude of the payouts and executing calls.

3.1 Vyper and the EVM

There is another contender in the smart contract space called Vyper. This is a newer language, made available with many of the "creature comforts" of Python. It is considerably less capable by design. It does not feature infinite lists at the time of this paper, and many other features available in Solidity are not granted in Vyper for security reasons. They say that Vyper is better at computing an upper bound for gas fees, but I do not know if this is true.

Vyper, like Solidity and the other defunct EVM languages, compiles down to an Assembly-like language called Application Binary Interface (ABI), which can run on EVM. Once you compile and thoroughly test your smart contract, you can push it to the rest of the network from an established full node. It will remain there, able to answer web3 queries until it is burned.

3.2 Gas Fees

4 Valuation

5 Conclusion

6 Code