Presenting: VeeDo

a STARK-based VDF Service

StarkWare

Follow

StarkWare

--

1

Listen

Share

Introduction

We are happy to announce that VeeDo — our STARK-based Verifiable Delay Function (VDF) service — is now live on Mainnet. This post is intended as a primer on VeeDo. It starts by sketching VeeDo's building blocks. We then describe our first application for VeeDo, a PoC for a randomness beacon, and compare our solution to existing randomness solutions. Finally, we discuss some future applications and business models we are contemplating for VeeDo.

The VeeDo Cryptographic Building Blocks

Verifiable Delay Functions (VDFs) were first suggested in the work ofBoneh et al. As implied from their name, VDFs are functions that use computation to provide a delay, a time lag.

Importantly, it should be very difficult and/or expensive for anyone to reduce the amount of time delay by more than a constant, well-understood, factor.

The VDF that VeeDo uses is a function f

that is:

1. Slow to compute
2. Fast to invert (i.e., computing f

^(-1)

is fast)

1. One can provide a proof π for the computational integrity of f

, in a way that bypasseses the need to replay the slow computation of the function f

.

To achieve the above properties of a VDF, VeeDo combines the delay function f

with another cryptographic primitive, the STARK protocol.

Delay function

The delay function f

gets as an input a pair: a field element s

as a seed, and a number of iterations n

. It runs a single loop for n

iterations, starting from that seed. (The number of iterations, n

, well over a million, is a parameter that can be tweaked to achieve the desired delay). Each iteration of the loop is also simple: find the root of a low-degree (cubic) polynomial modulo a prime p

that is “large”, in the VeeDo PoC, a 126-bit prime. The root-finding step is roughly 42x harder than the inverse direction, of evaluating a cubic polynomial.

STARK Protocol

The verifiable

property of the VDF, i.e., its poly-logarithmic verification complexity, is achieved by the STARK protocol. After the computation of f

(s,n) is done, three elements <s, n, f

(s,n)> are sent to the STARK prover. The prover then proves that the computation is valid. Both the computation off

on s

and the proof-generation are done off-chain. <s, n

, f

(s,n)>, along with the proof, are then sent to the Verifier, a smart-contract running on-chain. The efficient STARK Verifier runs in time that is poly-logarithmic in n (the length of f

computation).

Note that for small delays, the naive verification method of computing n iterations of f

^(-1)may be sufficient. The role played by the STARK proof here is to exponentially

compress the naive verification process (from n to poly-log(n)) when large delays are needed.

Our analysis points to another important property of VeeDo’s VDF: it is hardware-resistance

. In computing the VDF, GPUs and FPGAs do not have an advantage over a standard 64-bit CPU in terms of latency. Moreover, we have a detailed understanding on the bound on improvement one could expect to gain from custom ASIC for this computation. A detailed analysis of these aspects will be included in a white-paper we will publish on VeeDo.

First: Randomness

Background

The first application we intend to tackle with VeeDo is trustless and un-biasable randomness on Ethereum. Currently, Ethereum’s most widely-used source of randomness is the block hash. There are over 3,500 smart contracts using it. Block-hash randomness is easy to use, it is accessible (generated every block), and inexpensive. These very properties also make block-hash randomness easy to manipulate, and researchers have naturally demonstrated this (see for example, Bunz et al.). Two other sources of randomness are Chainlink’s oracle-based randomness service and Keep Network’s random beacon (based on a BLS Threshold Relay). To remain unbiased, these solutions either require an honest majority, or assume non-collusion of the operator and the oracle provider. Thus, their trust requires crypto-economic mechanisms and non-trivial trust assumptions.

VeeDo: Trustless, Unbiasable Randomness Beacon

Using a VDF as a randomness beacon was originally proposed by Justin Drake, and is the leading proposal for the randomness of the Beacon Chain in Eth 2.0. The basic idea is to use a source of high entropy as the seed s to the VDF (e.g. the block hash), and then run the VDF for n iterations and use its output f

(s,n) as the unbiased randomness. Assuming no one knows to predict f

(s,n) from s in less than t seconds, we can ask players to make all sorts of moves and bids before t seconds transpire, and use the result f

(s,n) as the randomness for events that happened before t seconds have lapsed. In other words, the unbiasability of a VDFs randomness is not predicated on crypto-economic or trust assumptions; rather, it relies on assumptions about the maximal speed of sequential computations.

PoC Description

The PoC is comprised of an off-chain component and on-chain smart contracts. The off-chain component includes the heavy computational parts of the VDF, namely, calculating the delay function and generating the proof. On chain there are two smart contracts: a Verifier smart contract which verifies the proof, and a Beacon smart contract where we register the calculated randomness (for validated proofs).

The specifics of the flow is as follows:

1. Pick a block every 820 blocks (approximately once every 3 hours)
2. Compute the seed s

from the block hash

1. Compute the VDF output f

(s,n)

— run the delay function f

on s

for n

iterations

1. Generate a proof attesting to the validity of the computation
2. Send the proof to theVerifier smart contract
3. Send the new randomness to theBeacon smart contract

In the PoC, the number of iterations is 1022*25–1, which corresponds to an approximately 3.5-minute delay (on a strong, latest generation, CPU, each iteration takes ~634 ns). The new randomness along with the proof is expected to be accepted on-chain approximately 20 minutes after the chosen block was mined. This time can be improved, but it is not part of the scope of this PoC. For a more detailed description of the technical aspects, check out the github repository.

An important disclaimer: as a PoC, there are no guarantees on uptime, nor on the service's longevity. This is an exploratory move, intended to solicit comments and ideas from the community.

Future Applications

There are several other applications we're looking into, beyond randomness. We will briefly discuss two of them: timelocks and a next-generation PoW mechanism. We're confident that many other applications will emerge, as the ecosystem starts exploring VeeDo in earnest.

Timelocks: timelocks are an important cryptographic building block. They allow information to be sealed for a predetermined period of time, and then made public. Typically, timelocks are implemented using a commitment scheme. The main drawback of such schemes is that the committing party is still required to be online at a later time, to reveal the information committed. Implicitly, this also means that the committing party has the option to avoid revealing its commitment. This amounts to a free option, and if this option isn't properly priced, the market's efficiency is diminished. Two interesting use-cases in permissionless blockchains are decentralized voting and sealed-bid auctions. VeeDo can help: as we've mentioned above, one interesting property of VeeDo's delay function is that the inverse (f

^(-1)) is much shorter to compute. A user interested in timelocking a secret b

(e.g., a bid or a vote) may quickly compute and publish s=f

^(-1)(b, n). The public may now learn b

by computing f

(s,n), resulting inb

being revealed with the desired delay.

NextGen PoW: a robust PoW algorithm remains a challenging problem in the blockchain space. The problem is amplified when attempting to jumpstart

a blockchain and its token economy. Existing solutions are deficient: airdrops are prone to Sybil attacks. ICOs are limited to accredited investors. Finally, the original vision of “One CPU = One Vote/Token” turned out to be rather elusive, as invariably custom hardware is built to accelerate any given PoW algorithm. The result is a reduced degree of decentralization for the network being mined. VeeDo presents an alternative, which levels the playing field: owners of expensive and customized hardware have very little advantage (and a bounded one) over everywoman (and everyman) with a CPU.

Business Model Innovation

VeeDo is not only a technology experiment, but also a business one. DeFi is growing fast, and the ecosystem is only starting to understand what can be done with the many money Legos being developed these days. One could categorize them by the layer on which they operate:

- Layer-1 (L-1): As Uniswap and others are showing us, there’s plenty one can build with these Legos directly on L-1.
- Layer-2 (L-2): From our very blockchain-centric view of the universe, one could say that the fintech universe (PayPal, Coinbase, etc.) is all L-2.
- L-1/L-2 Interface: Money Legos could be made more powerful if we manage to effectively bridge the L-1/L-2 divide, and to harness the resources available in L-2 (computation & storage in the cloud/mobile) to power interactions on L-1.

VeeDo is our first attempt at exploring this interface. VeeDo’s “frontend” is a smart-contract on Ethereum (L-1); its “backend” computes STARK proofs in the cloud (L-2).

The first set of challenges has to do with service availability: how do we ensure sufficient uptime? How do we ensure that the service in its entirety is sufficiently decentralized? Here there are several directions to explore, including open-sourcing, licensing, and having the smart-contract charge a fee.

Monetizing via a smart-contract fee can present“free-riding” challenges, and particularly when considering public goods, such a robust source of randomness. There too, we have some ideas up our sleeves that we will be exploring in the coming period.

Conclusion

VeeDo is StarkWare’s STARK-based VDF, and its PoC is now live on Mainnet. We see an exciting road ahead with many applications. The first application is using VeeDo to generate robust randomness. This is all experimental stuff, so we’re eager to hear your feedback, discuss applications of interest, etc. Please email us at veedo@starkware.co

~Kineret Segal,

[Tom Brand

](https://twitter.com/bbrandtom)