

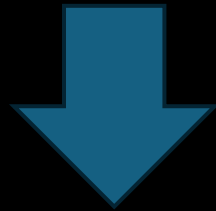


# Dark Pool Evolved

Trustless Randomness (VRF) using EIP-2537  
(Precompile for BLS12-381)

# Problem: MEV in Dark Pools

- Dark Pool as Private DEX
  - Challenge: transaction ordering affected fairness
- MEV (Miner Extractable Value)
  - Miners reorder or censor transaction to gain benefits



Need for Random Ordering

# VRF (Verifiable Random Function)

- Traditional Random
  - Off-chain: oracles
    - Need trust assumption
  - On-chain: generate & proof
    - Gas too high (without precompile)

# May 7, 2025, Pectra (Prague) Upgrade

Better user experience:

- [EIP-7702](#) - Set EOA account code
- [EIP-7691](#) - Blob throughput increase
- [EIP-7623](#) - Increase calldata cost
- [EIP-7840](#) - Add blob schedule to EL config files

Better staking experience:

- [EIP-7251](#) - Increase the MAX\_EFFECTIVE\_BALANCE
- [EIP-7002](#) - Execution layer triggerable exits
- [EIP-7003](#) - General purpose execution layer requests
- [EIP-7004](#) - Only validator deposits on chain

**EIP-2537**

Protocol efficiency and security improvements:

- [EIP-2537](#) - Precompile for BLS12-381 curve operations
- [EIP-2935](#) - Save historical block hashes in state
- [EIP-7549](#) - Move committee index outside Attestation

## EIP-2537 Precompile for BLS12-381 curve operations

Operation	EIP-1108 Gas (BN254)	EIP-2537 Gas (BLS12-381)
G1 Addition	150	375
G1 Multiplication	6,000	12,000
Pairing Check	$34,000 \times k + 45,000$	$32,600 \times k + 37,700$
G1 MSM	—	Dynamic
G2 Addition	—	600
G2 Multiplication	—	22,500
G2 MSM	—	Dynamic
Fp → G1 Mapping	—	5,500
Fp2 → G2 Mapping	—	23,800

Enable on-chain VRF

Place Order

Current Epoch Progress

Block 2 / 5

matching

Matching...

ETH-USD

Price

\$4025.50

24h Change

+1.65%

Liquidity

High

Spread

≈ 4022.5 - 4026.5 USD

Place Order

Identity Required

Buy

Sell

Limit

Market

Amount (ETH)

0.0000

Price (USD)

Trading Activity

Trading Activity by Epoch

Recent epochs with orders

Epoch #56

matching

15 orders

15 pending

Pending Orders

- SELL 1 ETH
- pending
- SELL 1 ETH
- pending
- SELL 1 ETH
- pending
- BUY \* ETH
- pending
- BUY 1 ETH
- pending
- SELL 2 ETH
- pending
- SELL \* ETH
- pending
- SELL 2 ETH
- pending
- BUY \* ETH
- pending

Epoch #55

matching

26 orders

18 pending 8 executed

Epoch #54

completed

29 orders

25 pending 4 executed

Epoch #53

completed

26 orders

18 pending 8 executed

# Epoch-based Ordering

## Place Order

Current Epoch Progress

**Across epochs →**  
earlier epoch takes priority

**Within an epoch →**  
order decided by VRF randomness

## Trading Activity

Trading Activity by Epoch

Recent epochs with orders

Epoch #56 **matching** 15 orders

15 pending

Pending Orders

SELL 1 ETH

pending

SELL 1 ETH

pending

SELL 1 ETH

pending

BUY \* ETH

pending

BUY 1 ETH

pending

SELL 2 ETH

pending

SELL \* ETH

pending

SELL 2 ETH

pending

BUY \* ETH

pending

Epoch #55 **matching** 26 orders

18 pending 8 executed

Epoch #54 **completed** 29 orders

25 pending 4 executed

Epoch #53 **completed** 26 orders

18 pending 8 executed

DEMO

# Prover (Typescript)

- **Input:**  $sk, pk, in$

where  $sk$  is the private key used to generate the signature and randomness output, and  $in$  is the input.

1. Compute

$$preout = sk \cdot H_G(in)$$

where  $H_G$  is the hash-to-curve operation and  $H$  is a general hash function.

2. Select a random scalar  $r_1 \in \mathbb{F}_p$ .

3. Compute

$$R = r_1 \cdot G$$

$$R_m = r_1 \cdot H_G(in)$$

4. Compute

$$c = H_p(in, pk, preout, R, R_m)$$

where  $H_p$  is a hash mapped into the finite field  $\mathbb{F}_p$ .

5. Compute

$$s_1 = r_1 + c \cdot sk$$

- **Output:**  $c, s_1, preout$



## Verifier (Solidity)

1. Compute

$$R = s_1 \cdot G - c \cdot pk$$

$$R_m = s_1 \cdot H_G(in) - c \cdot preout$$

2. Verify

$$c \stackrel{?}{=} H_p(in, pk, preout, R, R_m)$$

3. If the equality holds, compute

$$out = H(preout, in)$$

and output *out*; otherwise, output **false**.

# Proof of Work

Ethical identity, ring VRFs

EC VRF

preout:  $sk \cdot H_G(in)$

compk =  $sk \cdot G + bk$

Reval

$r_1, r_2 \in \mathbb{F}_p$

$R = r_1 G + r_2 k$

$R_m = r_1 H_G(in)$

$C = H_p(ad, in, compk, preout, R, R_m)$

$s_1 = r_1 + c \cdot sk$

$s_2 = r_2 + c \cdot bk$

$\pi = (C, s_1, s_2)$

return signature  $\sigma = (\pi, preout)$

PedVRF.Verify(compk, in, ad,  $\sigma$ )  $\rightarrow (out \vee \perp)$

$\sigma = (preout, C, s_1, s_2)$

input: pk, in, C,  $s_1$ , preout

$R = s_1 \cdot G - C \cdot pk$

$R_m = s_1 \cdot H_G(in) - C \cdot preout$

$C = H(in, pk, preout, R, R_m)$

$R = s_1 \cdot G + s_2 \cdot k - C \cdot compk$

$R_m = s_1 \cdot H_G(in) - C \cdot preout$

outputs  $H(in, preout)$

25.1.1 (SUN) ethstewh@hackernews

ring VRF

$(sk_i, pk_i) \leftarrow \text{Keygen}$

$out \leftarrow H(in, sk_i, H_G(in))$

$compk \leftarrow sk_i \cdot G + bk$

VRF (classical)

$(sk, pk) \leftarrow \text{Keygen}$

$out \leftarrow H(in, sk, H_G(in))$

$compk \leftarrow sk \cdot G + bk$

Proof:

① out 由 sk 产生, 为 zkPKR

② sk 为消息签名

with FS transform

$r_1, r_2 \in \mathbb{F}_p$

$R = r_1 G + r_2 k, R_m = r_1 H_G(in)$

Verify:

$rVRF.Sign(sk, comring, opring, in, ad) \rightarrow \sigma$

$\sigma = (compk, \pi_{ring}, comring, \sigma')$

$\sigma' = PedVRF.Sign(x, b, in, ad')$

$ad' \leftarrow ad \parallel \pi_{ring} \parallel comring$

$rVRF.Verify(comring, in, ad, \sigma) \rightarrow (1, out) \vee (0, \perp)$

$\sigma = (compk, \pi_{ring}, comring, \sigma')$

$ad' \leftarrow ad \parallel \pi_{ring} \parallel comring$

$NIZK_{ring}.Verify((compk, comring), \pi_{ring})$

$PedVRF.Verify(compk, in, ad', \sigma')$

① 注册 R

② 提交订单, 2 R

③ 排序 撮合

dleg-vrf bls12-377

$P_i(e)$

$i = i_1 \dots i_n$

$l = l_1 \dots l_n$

$f_j = l_j e + a_j$

$f_{j,1} = f_j = l_j e + a_j = \delta_{l_j} e + a_j$

$f_{j,0} = e - f_j = (1 - l_j) e - a_j$

$= \delta_{0,l_j} e - a_j$

$i = i_1 \dots i_n$

$P_i(e) = \prod_{j=1}^n f_{j,i_j}$

$P_i(e) = \prod_{j=1}^n (\delta_{i_j} e - a_j) + \prod_{k=0}^{n-1} P_{i,k} e^k$

$= \delta_{i,j} e^n + \prod_{k=0}^{n-1} P_{i,k} e^k$

$u = H(x)$

$P_{256}$

$secp256k1$

$sk \cdot G + r_1 \cdot H$

$c_2 = c_1^{sk}$

$h = g^{sk}$

$rVRF.Sign(sk, comring, opring, in, ad) \rightarrow \sigma$

$\sigma = (compk, \pi_{ring}, comring, \sigma')$

$\sigma' = PedVRF.Sign(x, b, in, ad')$

$ad' \leftarrow ad \parallel \pi_{ring} \parallel comring$

$rVRF.Verify(comring, in, ad, \sigma) \rightarrow (1, out) \vee (0, \perp)$

$\sigma = (compk, \pi_{ring}, comring, \sigma')$

$ad' \leftarrow ad \parallel \pi_{ring} \parallel comring$

$NIZK_{ring}.Verify((compk, comring), \pi_{ring})$

$PedVRF.Verify(compk, in, ad', \sigma')$

$H(x)^y = H(x)^{s' + \frac{sk}{e}}$

$= m_2 \cdot v^e$

hash-vec

$m_1, m_2$

extra-x: hash-vec

$x$

$\pi$

$l$

$(l_j \cdot G + r_j \cdot H) \cdot (a_j \cdot G + s_j \cdot H)$

$= (l_j e + r_j) \cdot (a_j e + s_j)$

$= (l_j e + r_j) \cdot (a_j e + s_j)$

$u = H(x)$

$P_{256}$

$secp256k1$

$sk \cdot G + r_1 \cdot H$

$c_2 = c_1^{sk}$

$h = g^{sk}$