VRF 计算

Jade Xie

2025 年 11 月 1 日

VRF(Verifiable Random Function) 可以理解为产生一个可以验证的随机数的函数，其输入是任意的输入，可以理解为随机种子，其输出为一个随机数和一个证明。对于一个用户，首先其有一个公私钥对，其可以通过自己的私钥生成一个随机数和证明，其他人拿着该用户的公钥和证明就可以验证该随机数确实是由该用户生成的。

本仓库代码中实现的 VRF 属于 EC-VRF，也就是基于椭圆曲线，其中曲线选取为 bls12-381。

1 VRF 计算

主要分为两个阶段，一个阶段是 Prover 用自己的私钥生成随机数以及证明，第二个阶段是 Verifier 对证明进行验证。

1.1 Porver 计算

- 输入：sk, pk, in

其中 sk 为进行签名和输出随机数的私钥，in 为任意的输入。

1. 计算
$$preout = sk \cdot H_{\mathbb{G}}(in)$$
其中 $H_{\mathbb{G}}$ 是 hash to curve 操作。在具体实现过程中，这一步也可以改为计算
$$preout = sk \cdot H_{\mathbb{G}}(keccak256(in))$$
其中 $keccak256$ 为哈希函数。
2. 在 $\mathbb{F}_p$ 中选取随机数 r_1。
3. 计算
$$R = r_1 \cdot \mathbf{G}$$
$$R_m = r_1 \cdot H_{\mathbb{G}}(in)$$
4. 计算
$$c = H_p(in, pk, preout, R, R_m)$$
其中 H_p 是在有限域 $\mathbb{F}_p$ 上进行哈希，先进行哈希，再映射到有限域 $\mathbb{F}_p$ 内。
5. 计算
$$s_1 = r_1 + c \cdot sk$$

- 输出：$c, s_1, preout$

为了减少 Verifier 在链上的计算与验证，这一阶段 Prover 不计算和输出随机数 $out = H(preout, in)$ ，而是直接将 $preout$ 作为证明的一部分发给 Verifier，Verifier 拿到 $preout$ 后可以自己通过哈希算出随机数，否则 Verifier 还需要验证 Prover 发送的 out 是正确的。

1.2 Verifier 验证计算

- 输入：$pk, in, c, s_1, preout$

1. 计算

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

$$R_m = s_1 \cdot H_{\mathbb{G}}(in) - c \cdot preout$$

2. 判断

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

3. 上一步如果相等，计算

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

并输出 out ，否则输出 $false$ 。

这一阶段输出的 out 就可以作为整个 VRF 输出的随机数。