VRF-based mining
Simple non-outsourceable cryptocurrency mining

***Abstract*—This paper introduces VRF-based mining, a simple and effective way of making pooled mining impossible. Instead of using hash functions, VRF-based mining uses Verifiable Random Functions (VRFs) for Proof-of-work (PoW)-based consensus. As VRF binds an output with a secret key, a pool operator should reveal his private key to outsource the mining process to miners, and miners can anonymously steal cryptocurrency in the pool operator's wallet.**

We revisit the definition of non-outsourceability in existing research, and show that, while being simple and efficient, VRF-based mining achieves the same level of non-outsourceability as the state-of-the-art construction. We discuss several considerations on instantiating VRFs for VRF-based mining. In addition, we show that VRF-based mining is easy to implement, and introduces small overhead compared to hash-based mining. Moreover, we discuss why partial outsourcing in VRF-based mining is unprofitable, and how to make it even more costly.

I. Introduction

Bitcoin [1] started the era of cryptocurrency. In Bitcoin, each participant maintains its own ledger, called blockchain. A blockchain is formed as a chain of blocks, and each block contains a batch of transactions. Blockchain is designed to be append-only. Participants keep appending blocks to the blockchain. To add a new block, a participant should solve a Proof-of-Work (PoW) [2] puzzle, which is computationally hard. Once solving the puzzle, the participant appends his block to the blockchain, and get some coins as reward. By adaptively adjusting the difficulty of PoW puzzles, PoW-based consensus can stabilise the speed of generating blocks. Participants of PoW-based consensus are also known as miners, the process of solving PoW puzzles is known as mining, and the computation power used for mining is known as mining power.

Today, mining pools dominate the mining power of most blockchains using PoW-based consensus. Due to the limited rate of producing blocks, miners' rewards are highly volatile. A mining pool allows miners to mine on blockchain in the name of the pool operator, and the pool operator distributes the mining reward to solo miners in a fine-grained way. In this way, a miner can get reward more stably by joining a mining pool.

However, mining pools lead to the centralisation of mining power, which contradicts Bitcoin's design objective - decentralisation. Mining power centralisation weakens PoW-based consensus' security. A mining pool with sufficient mining power can perform numerous types of attacks to break the consensus, such as selfish mining attacks [3] and 51% attacks [1]. In addition, mining pools can censor the blockchain by deciding which transactions to include. To date (01/12/2019), four largest mining pools control more than 51% Bitcoin mining power [4], which can be a lurking threat of Bitcoin [5], [6].

A. Our contributions.

In this work, we introduce *VRF-based mining*, a surprisingly simple but effective approach to make pooled mining impossible. It employs Verifiable Random Functions (VRFs) [7] rather than hash functions for mining. With VRF, mining takes one's secret key as input, so a pool operator should reveal his secret key to miners in order to outsource mining. In this way, any miner can steal mining reward of a block owned by the pool operator anonymously.

Our contributions are as follows:

We propose the idea of VRF-based mining, and describe its detailed construction. We replace the hash function in PoW-based consensus with a VRF, and a block in the blockchain should contain the VRF hash and its proof, but does not need to contain the digital signature. Verifying the VRF hash can prove both the correctness of the hash and the authorship of the block in the same time.

We revisit the definition of non-outsourceable cryptocurrency mining. In particular, we show non-outsourceability consists of punishing pool operators and the unaccountability of punishing. We then show how VRF-based mining achieves the same level of non-outsourceability as Miller et al. [8].

We discuss how to instantiate VRFs for VRF-based mining. VRF has four tweakable components, namely the elliptic curve and two hash functions mapping strings from/to elliptic curve elements, and a normal hash function. We discuss considerations on choosing them for VRF-based mining.

We evaluate the feasibility of VRF-based mining. We implement Elliptic curve-based VRF (EC-VRF) specified in [9] using Go programming language. We compare its performance with three mining algorithms (SHA256D, Scrypt and CryptoNight), and evaluate its performance. Our results show that VRF-based mining is easy to implement and introduces small overhead.

We show that partial outsourcing in VRF-based mining is unrealistic. Partial outsourcing is that, a pool operator interactively outsources computation that does not need the secret key to miners. We show that partial outsourcing is unprofitable, as it's both computation-intensive and I/O-intensive. We also discuss how to instantiate VRFs to make it even more costly.

B. Paper organisation.

§II and §III describes preliminaries and the construction of VRF-based mining, respectively. §IV revisits the definition of non-outsourceability, and shows VRF-based mining achieves

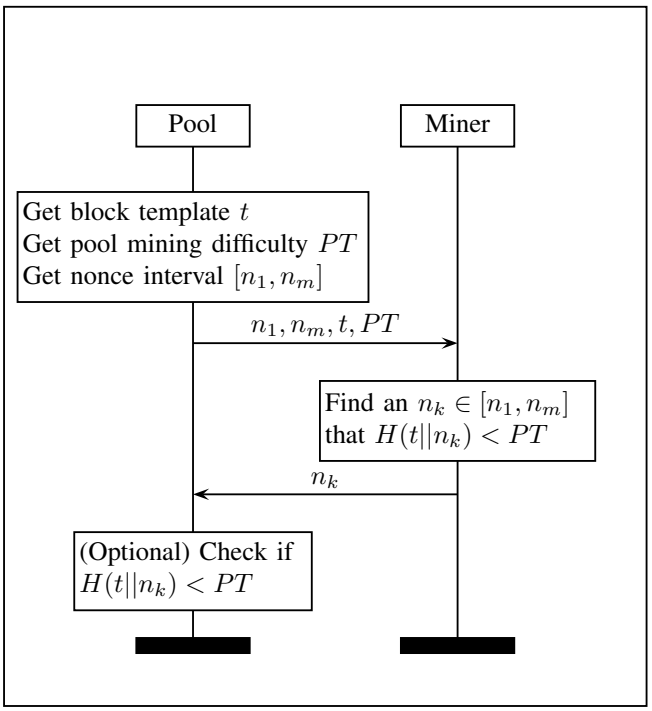

Fig. 1: Outsourcing in hash-based mining.

better non-outsourceability than existing proposals. §V discusses concerns of instantiating VRFs for VRF-based mining. §VI provides an experimental analysis on the practicality of VRF-based mining. §VII discusses why partial outsourcing in VRF-based mining is unprofitable. §VIII discusses potential problems in VRF-based mining and their solutions. §IX summaries related work, and §X concludes this paper. We attach the pseudocode of the standardised EC-VRF construction [9] in Appendix.

II. Preliminaries

A. Cryptocurrency mining

A miner mines on a blockchain with PoW-based consensus by solving crypto puzzles. The crypto puzzle is usually constructed using cryptographic hash functions, and works as follows. First, a miner constructs a block template t, which consists of transactions to include, hash of the last block, and other block metadata. Second, the miner should find a nonce n such that the hash of $t||n$ is smaller than a difficulty parameter. Once the miner find a valid nonce, he can append this block to the blockchain. To solve a PoW puzzle, miners iteratively computes hash values of different nonces until finding a hash satisfying the difficulty parameter. The difficulty can be parametrised by limiting the interval of hashes. For example, Bitcoin controls the difficulty of finding a nonce by specifying the number of leading zeros of hashes.

B. Mining pools

In a mining pool, the pool operator outsources the mining process by allowing miners to find nonces for him. First, a pool operator specifies the pool difficulty PT - a difficulty lower than the blockchain network - and a search interval $[n_1, n_m]$ of nonces. Then, the pool operator sends the block template t, PT, n_1 and n_m to the miner, and the miner starts to find a nonce in $[n_1, n_m]$ which satisfies PT. Once finding a valid nonce n_k, the miner sends n_k back to the pool operator. The pool operator then verifies whether n_k produces a hash satisfying PT, and records the miner's contribution (a.k.a. a share) if valid. After a time period (say 24 hours), the pool operator calculates the total contribution of each miner, and distributes the mining reward to miners according to their submitted shares. As PT is easier to solve, calculating the mining power of a miner using shares is more fine-grained than using blocks. In this way, each miner is rewarded in a more fine-grained way, so more stably.

C. Verifiable random functions

Verifiable Random Function (VRF) [7] is a public-key version of cryptographic hash function. In addition to the input string, VRF involves a pair of a secret key and a public key. Given an input string and a secret key, one can compute a hash. Anyone knowing the associated public key can verify the correctness of the hash, and can also verify the hash is generated by the owner of the secret key. Formally, a VRF consists of four algorithms: $\mathsf{VRFKeyGen}$, $\mathsf{VRFHash}$, $\mathsf{VRFProve}$ and $\mathsf{VRFVerify}$.

- $\mathsf{VRFKeyGen}(1^\lambda) \rightarrow (sk, pk)$: On input a security parameter 1^λ, outputs the secret/public key pair (sk, pk).
- $\mathsf{VRFHash}(sk, \alpha) \rightarrow \beta$: On input sk and an arbitrary-length string α, outputs a fixed-length hash β.
- $\mathsf{VRFProve}(sk, \alpha) \rightarrow \pi$: On input sk and α, outputs the proof π for β.
- $\mathsf{VRFVerify}(pk, \alpha, \beta, \pi) \rightarrow \{True, False\}$: On input pk, α, β, π, outputs the verification result $True$ or $False$.

A VRF should preserve the following three security properties [9]:

- **Uniqueness**: Given a secret key sk and an input α, $\mathsf{VRFHash}(sk, \alpha)$ produces a unique valid output.
- **Collision Resistance**: It is computationally hard to find two inputs α and α' that $\mathsf{VRFHash}(sk, \alpha) = \mathsf{VRFHash}(sk, \alpha')$.
- **Pseudorandomness**: It is computationally hard to distinguish the output of $\mathsf{VRFHash}(sk, \alpha)$ from a random string if not knowing the corresponding public key pk and proof π.

Algorithm 3 in Appendix A describes the Elliptic-curve-based VRF (EC-VRF) construction standardised in draft-goldbe-vrf [9].

III. VRF-based mining

Instead of using a hash function, VRF-based mining uses a VRF to produce hashes that satisfy the difficulty requirement. VRF takes both an input and a secret key to produce a hash. The owner of this secret key can produce a proof proving the ownership of his hash. Thus, to outsource the mining process, the pool operator has to reveal his secret key to

miners. However, with the secret key, any miner in the mining pool can steal all mining rewards.

Algorithm 1: Work(sk, t, NT).

Input: The secret key sk, the block template t, and the blockchain difficulty NT
Output: The block blk, the VRF output out, and the VRF proof π

```
Initialise n, out, blk                    ▷ Initialise variables
while n+ = 1 do                           ▷ Refresh the nonce
    blk ← ConstructBlock(t, n)            ▷ Assemble block
    out ← VRFHash(sk, blk)                ▷ Produce VRF output
    if out < NT then                      ▷ If meeting difficulty
        break                             ▷ Mining successful
    end
end
π ← VRFProve(sk, blk)                     ▷ Produce the proof
return blk, out, π                        ▷ Return block, hash and proof
```

Algorithm 2: Verify(blk, out, π, NT)

```
pk ← blk.txs[0].scriptPubKey              ▷ Find miner's pk
require(out < NT)                         ▷ Check if satisfying diff
/* Here VRFVerify(·) ensures:                                    */
/* 1. out is generated by the owner of sk                        */
/* 2. out is a valid output of VRFHash(sk, blk)                  */
require(VRFVerify(pk, blk, out, π))
... ▷ Verify other fields
... ▷ Verify transactions
```

Cryptocurrency mining consists of two components, namely mining blocks and verifying blocks. We call the process of mining a block Work, and the process of verifying a block Verify. Algorithm 1 and 2 describe Work and Verify of VRF-based mining, respectively.

Work. Miners run Work to mine new blocks. More specifically, a miner - with his private key sk the block template (a complete block without nonce) t - keeps searching for a nonce n that can make the (VRF) hash out of the block blk to meet the blockchain difficulty NT. Once finding a valid n, the miner produces the proof π of out, and appends blk (with n), out and π to the blockchain.

Verify. Upon incoming blocks, miners run Verify to verify their validity. While other verifications are the same as in hash-based mining, Verify in VRF-based mining should additionally run VRFVerify$(\cdot)$ to verify 1) whether out is produced by the owner of sk, and 2) whether out is a valid output of VRFHash(sk, blk).

Block structure. Different from hash-based mining, in VRF-based mining a block should additionally attach out and π, but no longer need the signature of a block. Other miners without knowing sk cannot produce out, but can use π to verify out is generated by someone knowing sk. Through proving the authorship of out, π also proves the authorship of the block. Thus, miners only need to sign out, but do not need to sign blocks.

IV. Non-outsourceability analysis

In this section, we revisit existing definitions on non-outsourceability, and show that VRF-based mining achieves the same level of non-outsourceability as the state-of-the-art construction - the Strongly non-outsourceable scratch-off puzzle [8].

A. Revised definitions

Miller et al. [8] first formalise cryptocurrency mining as scratch-off puzzles, and formally define non-outsourceability. They define two levels of non-outsourceability, namely **Weak non-outsourceability** and **Strong non-outsourceability**.

- **Weak non-outsourcability**: If the pool operator outsources the mining process, miners can always steal the reward of mining.
- **Strong non-outsourcability**: In addition to **Weak non-outsourcability**, the pool operator cannot link the stolen mining reward with the miner who steals it.

Basically, **Weak non-outsourceability** defines the punishment of outsourcing, while **Strong non-outsourceability** covers both the punishment and the anonymity of the stealer. We call the property defining the punishment of outsourcing **Punish-mining-reward**. The anonymity of thieves defined in [8] is a special case of **Transaction Unlinkability** [10]: Given two arbitrary transactions, it is hard to know whether their outputs belong to the same secret key. To steal the mining reward in [8], the miner should create a transaction, of which the input is the mining reward and the output points to his address. Finding out who steals the mining reward is equivalent to finding out the address holding the output, and we call the property making such attempt infeasible **Stealing-unlinkability**.

B. Non-outsourceability of VRF-based mining

VRF-based mining achieves both **Punish-mining-reward** and **Stealing-unlinkability**. In VRF-based mining, the pool operator outsources mining by revealing his secret key to miners. The best thing the pool operator can do is to use a new secret for each block, and stealers can only steal the mining reward of a single block with a secret key. Thus, VRF-based mining achieves **Punish-mining-reward**. Similar with the Strongly non-outsourceable scratch-off puzzle [8], VRF-based mining achieves **Stealing-unlinkability** by allowing a stealer to use a freshly generated address to steal cryptocurrency. To receive the stolen cryptocurrency of the pool operator, the stealer can create a new address, and construct a transaction of which the stolen cryptocurrency directs to this address. As this new address has no historical transactions, linking the transaction stealing cryptocurrency with other transactions can be impossible. Then the stealer can then spend his stolen cryptocurrency anonymously using numerous techniques, such as mixing services [11][12][13][14] and stealth addresses [10].

V. Instantiating VRF

In order to implement VRF-based mining, one needs to first instantiate the VRF. VRF in Algorithm 3 has four configurable

components, including the elliptic curve and three hash functions. Three hash functions are: $H_1(\cdot)$ mapping an arbitrary-length string into an elliptic curve element, $H_2(\cdot)$ mapping an elliptic curve element into a fixed-length string, and $H_3(\cdot)$ mapping an arbitrary-length string into a fixed-length string.

In this section, we discuss considerations on choosing these four components for VRF-based mining.

A. Elliptic curve

As neither blockchains and VRF limits the choice of elliptic curves, any elliptic curve can be adapted. For example, among prominent elliptic curves, Ed25519 [15] can be a promising choice. First, Ed25519 works on a prime field with the prime number $2^{255}-19$, which provides sufficient enumeration space for VRF. Second, Ed25519 supports EdDSA, a fast and secure digital signature algorithm. Last, Ed25519 is compatible with existing blockchains, as numerous blockchains and projects using VRF adapt Ed25519 as their underlying elliptic curve.

B. Hashing a string to an elliptic curve point $H_1(\cdot)$

$H_1(\cdot)$ is a hash-to-curve function, which should prevent distinguishing behaviours: adversaries cannot learn any pattern of the input from its hash. In addition, the hash-to-curve function used in VRF should be deterministic, otherwise the hash will be unreproducible.

A standardisation document [16] specifies several hash-to-curve functions that fit into different elliptic curves and satisfy our requirements: Icart Method [17], Shallue-Woestijne-Ulas Method [18], Simplified SWU Method [19] and Elligator2 [20]. Within these hash-to-curve functions, Elligator2 is the recommended one for Ed25519.

C. Hashing an elliptic curve point to a string $H_2(\cdot)$

$H_2(\cdot)$ hashes an elliptic curve point to a fixed-length string. It can be divided to two steps: 1) encoding an elliptic curve point to a string, and 2) hashing the string using a normal hash function. The encoding step can be bidirectional and unencrypted, so can be done simply by converting a big integer to a string. The hashing step should be cryptographically secure, so can adapt any existing cryptographically secure hash functions.

D. Normal hash function $H_3(\cdot)$

$H_3(\cdot)$ is only used in VRFProve (proving the authorship of hashes) and VRFVerify (verifying the authorship of hashes). The overhead of proving and verification should be minimised, so fast and cryptographically secure hash functions (such as Keccak [21] and BLAKE [22]) are suitable for $H_3(\cdot)$.

E. Memory-hard mining

For ASIC-resistant cryptocurrencies using memory-hard hash functions (e.g., Ethereum [23] and Monero [24]), there are some concerns on designing $H_1(\cdot)$ and $H_2(\cdot)$. To make mining memory-hard, VRFHash should be memory-hard. VRFHash of the standardised VRF consists of one $H_1(\cdot)$ hashing, one scalar-point multiplication and one $H_2(\cdot)$ hashing. Thus, making VRF-based mining memory-hard can be achieved by making $H_1(\cdot)$ or $H_2(\cdot)$ memory-hard. This can be achieved by embedding a memory-hard hash function (such as Ethash [25] and CryptoNight [26]) in $H_1(\cdot)$ or $H_2(\cdot)$.

VI. Practicality of VRF-based mining

In this section, we experimentally show that VRF-based mining is easy to implement and introduces small overhead. First, we compare the performance of VRFs with three other hash functions used for cryptocurrency mining, namely SHA256D used in Bitcoin, Scrypt used in Litecoin and CryptoNight used in Monero. The comparison shows that VRF can be much faster than Ethereum and CryptoNight. Second, we measure the runtime of each step ($H_1(\cdot)$, point multiplication and $H_2(\cdot)$) of VRFHash. The result shows that the elliptic curve scalar multiplication dominates VRFHash's runtime, which can be optimised in the future.

A. Experimental setting

We implement the standardised EC-VRF in Algorithm 3 using Go programming language. In particular, we use Ed25519 [15] as the underlying elliptic curve, Elligator [20] as $H_1(\cdot)$, SHA-3 as the hash function. Ed25519, SHA-3 and the encoding of points on Ed25519 are supported by Go's standard library. We do not apply any optimisation on the EC-VRF implementation. We use open-source implementations basing on their popularity on https://github.com/ for SHA256D [1], Scrypt [2], and CryptoNight [3].

All experiments run on a MacBook Pro with a 2.2 GHz Intel Core i7 Processor and a 16 GB DDR4 RAM. For each group of experiment, we run an algorithm for ten times, then take the average data as the experimental results.

B. Comparison between VRF and other mining algorithms

We compare the performance of VRF with other mining algorithms. Figure 2 shows the runtime of VRF and other algorithms. First, SHA256D is much faster than other algorithms, as SHA256D is simply executing SHA256 twice. Second, VRFVerify is slightly slower than VRFProve, and VRFProve is slightly slower than VRFHash. Last, although without optimisation, all functions of VRF are much faster than Scrypt and CryptoNight. This means that VRF is easy to implement and introduces little overhead, so suitable for cryptocurrency mining.

C. Runtime breakdown of VRF

We profile VRFHash by evaluating its runtime of each step. Figure 3 shows that, the elliptic curve scalar multiplication $\gamma \leftarrow h^{sk}$ takes 88% of VRFHash's running time. This is as expected, as we use SHA-3 as the hash function for $H_1(\cdot)$ and $H_2(\cdot)$, and SHA-3 is designed to be fast. Meanwhile, we calculate the elliptic curve scalar multiplication using the trivial double-and-add method without any optimisation, thus is much slower than $H_1(\cdot)$ and $H_2(\cdot)$. Even so, VRFHash is

[1] https://github.com/seehuhn/sha256d
[2] https://github.com/elithrar/simple-scrypt
[3] https://github.com/Equim-chan/cryptonight

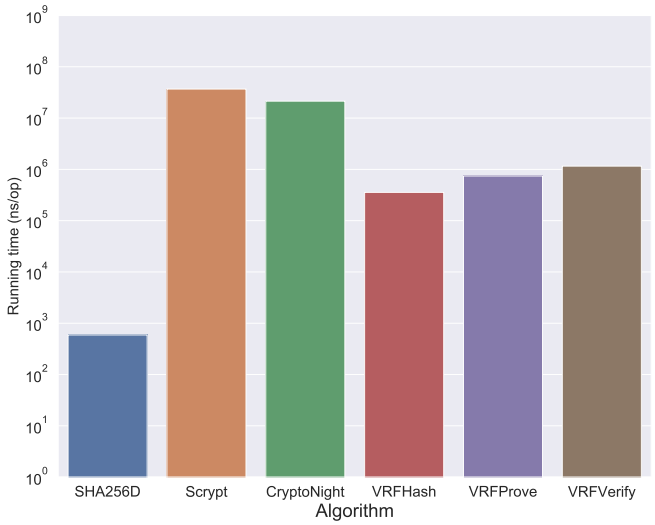

Fig. 2: Comparing the runtime of VRF and other mining algorithms.

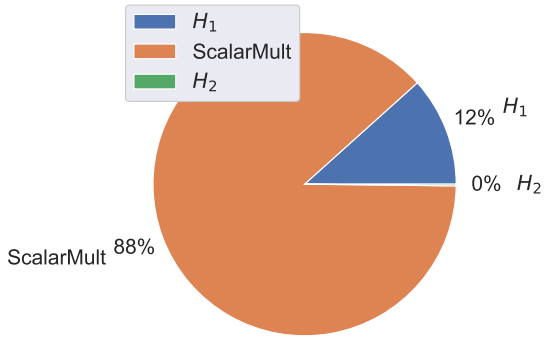

Fig. 3: Runtime breakdown of VRFHash.

still much faster than Scrypt and CryptoNight. This further proves that VRFs introduce little overhead.

VII. Profitability of Partial Outsourcing

The pool operator may still have opportunity to partially outsource mining. VRFHash consists of three steps: $h \leftarrow H_1(\alpha)$, $\gamma \leftarrow h^{sk}$ and $\beta \leftarrow H_2(\gamma)$. The non-outsourceability is from the second step $\gamma = h^{sk}$, as it requires the knowledge of the pool operator's secret key sk. The pool operator can outsource the first step $h = H_1(\alpha)$ by distributing different αs to miners, and also the last step $\beta = H_2(\gamma)$ by distributing different γs to miners.

However, such partial outsourcing is very inefficient and unprofitable compared to outsourcing in hash-based mining, due to the computing and I/O overhead. In this section, we show that partial outsourcing in VRF-based mining is unprofitable as it's computation-intensive and I/O intensive. We also show that making VRF fast can make VRF-based mining even more costly.

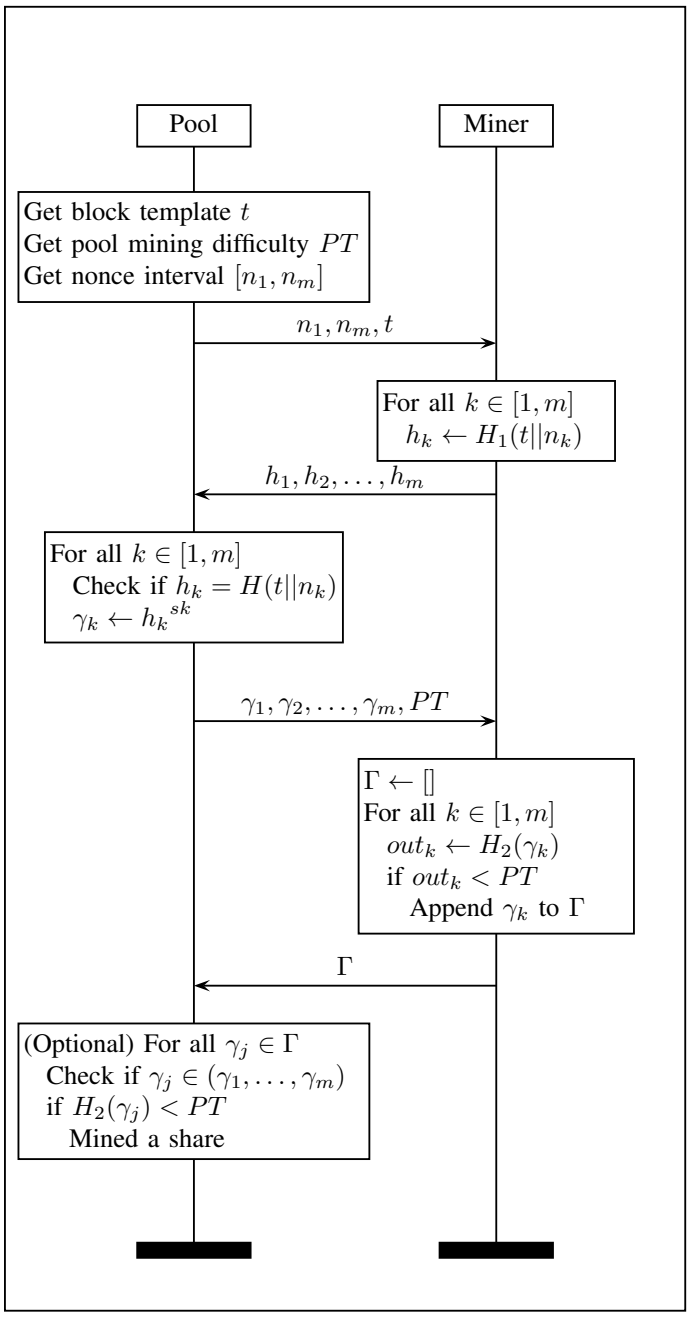

Fig. 4: Partial outsourcing in VRF-based mining.

A. Partial outsourcing in VRF-based mining

Figure 4 describe the process of outsourcing $H_1(\cdot)$ and $H_2(\cdot)$. In VRF-based mining, the second step h^{sk} is non-outsourceable, but the pool operator can outsource $H_1(\cdot)$ or $H_2(\cdot)$ interactively.

To outsource $H_1(\cdot)$, the pool operator generates a search interval $[n_1, n_m]$ of nonces, then sends the interval and the block template t to a miner. After that, the miner computes

$H_1(\cdot)$ with each nonce in $[n_1, n_m]$ as input, then sends back all $H_1(\cdot)$ hashes to the pool operator. If the pool operator does not trust miners (which reflects to the real world), he will verify these $H_1(\cdot)$ hashes before the second step i.e., multiplying each hash with his secret key sk.

After the second step, the pool operator obtains a series of $(\gamma_1, \gamma_2, \ldots, \gamma_m)$, and he can outsource $H_2(\cdot)$ as follows. The pool operator first sends these γs as well as the pool difficulty PT to the miner. Then, the miner calculates $H_2(\cdot)$ hashes of these γs, compare the hashes with PT, and sends back γs that satisfy the difficulty (denoted as Γ). Similarly, the pool operator optionally verifies the correctness of each of Γ, and accumulates the mined shares to the miner's total contribution.

B. The cost of verifying hashes from miners

The first obstacle of partial outsourcing is verifying hashes from miners. If the pool operator does not trust miners, he should verify both $H_1(\cdot)$ and $H_2(\cdot)$ hashes from miners. For outsourcing $H_1(\cdot)$, the pool operator should verify all of $h_1, h_2, \ldots, h_m$. For outsourcing $H_2(\cdot)$, The pool operator should verify σs satisfying PT. This introduces significant overhead, which is equivalent to mining by the pool operator himself. Thus, partial outsourcing unprofitable.

C. Partial outsourcing is I/O intensive

To workaround the verification overhead, the pool operator and the miners should trust each other. The pool operator and miners are still possible to collaborate, as pooled mining is beneficial for them. For the pool operator, he can earn some fees from the miners. For the miners, they can stabilise their revenue from mining.

However, even if they trust each other, partial outsourcing can still be unprofitable. VRF-based mining can be extremely I/O intensive, while the pool operator's server suffers from limited bandwidth. To outsource a single $H_1(\cdot)$ in VRF-based mining, the pool operator should at least receive a $H_1(\cdot)$ hash from the miner To outsource a single $H_2(\cdot)$, the pool operator should at least send a σ to the miner. Hash-based mining does not suffer from this problem. In hash-based mining, the pool operator only sends and receives constant-size data for each share. By increasing the share difficulty, the pool operator will receive fewer shares and support more miners.

We model the maximum hashrate the pool operator can support as follows. Consider a pool operator with bandwidth BW (in bytes/s), and the pool operator can achieve optimal bandwidth utilisation on mining. Assume the pool operator should transfer N bytes per hash. Thus, the maximum hashrate is

$$\text{Maximum hashrate} = \frac{BW}{N}$$

Either a $H_1(\cdot)$ hash or a γ (a point on the elliptic curve) is at least 32 bytes. If outsourcing both $H_1(\cdot)$ and $H_2(\cdot)$, the maximum hashrate the pool operator can support is Maximum Hashrate $= \frac{BW}{64}(h/s)$. If outsourcing only one of them, the maximum hashrate is Maximum Hashrate $= \frac{BW}{32}(h/s)$.

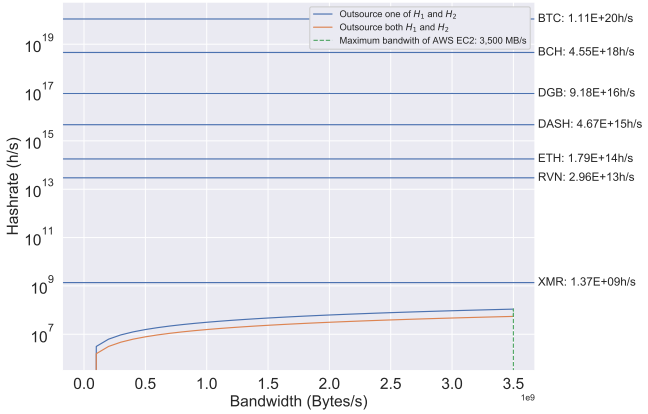

Fig. 5: Server bandwidth v.s. maximum hashrate. Data of server bandwidth and hashrate were fetched from [27] and [28].

Figure 5 shows relationship between the pool operator's network bandwidth maximum and the maximum hashrate. The result shows that, even a server with most bandwidth from AWS (I3EN Metal with 3,500 MB/s) can only support less than $\frac{1}{10}$ of Monero's total mining power. Within those mainstream cryptocurrencies, Monero's hashrate is the least. Even if the pool operator rents a cluster of I3EN Metal machines for more bandwidth, I3EN Metal is quite expensive (10.848 USD per hour), leading to high expense of maintaining a mining pool. Therefore, partial outsourcing in VRF-based mining is costly and unrealistic. By embedding lightweight mining algorithms in VRFs such as SHA256D, VRF-based mining can be maximally I/O-intensive.

VIII. Discussions

While eliminating mining pools can contribute to decentralisation, it will also introduce new problems. In this section, we discuss potential problems introduced by VRF-based mining, as well as how to address them. In particular, we focus on two problems: high reward variance and secret key leakage in memory.

A. High reward variance

As aforementioned, miners cannot obtain stable income via solo mining, so they choose pooled mining. In other words, unstable income may discourage miners to mine. With weaker incentive of mining, the blockchain will have less mining power, which weakens the security of PoW-based consensus.

An approach to address this problem is fine-graining the mining reward scheme. Miller et al. [8] proposes the Multi-tier reward scheme. In Multi-tier reward scheme, the mining difficulty is divided into different levels, and miners can mine blocks satisfying different difficulty levels arbitrarily. In this way, mining reward is distributed in a fine-grained way, so lowers the reward variance. StrongChain [29] introduces the

notion of Collaborative PoW, where miners are incentivised to mine blocks together and the mining reward is distributed in proportion to miners' contributions.

Another approach is to increase the rate of mining blocks. With more blocks mined in a time unit, the mining reward can also be more stable. For example, proposals on blockchain scalability such as sharding [30] and DAG [31] can stabilise the mining reward variance, although they are not designed for it.

B. Secret key leakage in memory

Different from hash-based mining, in VRF-based mining the secret key is kept in plaintext in memory all the time, as mining requires the secret key. This gives opportunity for adversaries to steal the secret key from the memory. For instance, the mining software has a bug, and the hacker can exploit this bug (even without logging in to the mining machine) to access the memory space of the mining software. In this way, the hacker can easily steal the secret key.

Unfortunately, keeping secret keys in memory is inevitable for VRF-based mining. To achieve non-outsourceability, the secret key should be combined to the mining process. As long as the mining is continuous and endless, the process should keep the plaintext secret key in memory. Not only VRF-based mining, but also all protocols combining the secret key to mining suffer from this issue.

Nevertheless, keeping secret keys in memory does not make VRF-based mining to be impractical. VRF-based mining is not the only protocol that requires in-memory secret keys. For example, TLS protocol requires servers to keep secret keys of their SSL certificates in memory. Also, the miner can use a new secret key for each block, so leaking a secret key does not lead to the stealing of mining reward in the future.

There have been several different ways to protect in-memory secret keys. First, the miner can isolate the scalar multiplication to a software or hardware enclave. Note that the encalves are unnecessary to be general-purpose. Moreover, the process of the mining software can proactively destroy itself once detecting anomalous memory access attempts. Furthermore, the miner can isolate mining from the blockchain node. For example, he can run the mining process independently with his blockchain node process. In this way, if an adversary compromises his node but not the operating system, the adversary still cannot read the private key in another process. In addition, he can run the mining process in an offline machine that can only communicate with his blockchain node. Moreoever, Oblivious RAM (ORAM) is a promising primitive to protect sensitive data in memory. ORAM allows CPUs to access data in memory while the data in memory is encrypted and the access pattern is hidden.

IX. Related work

To the best of our knowledge, VRF-based mining is the first construction that makes pooled mining *impossible*. In this section, we briefly review related research on preventing mining pools, and compare them with VRF-based mining. We classify related research to two types, namely mining protocols trying to address pooled mining, and decentralised mining pools.

A. Mining protocols

There are two mining protocols aiming at discouraging or breaking mining pools: the *non-outsourceable scratch-off puzzle* [8] and the *Two Phase Proof-of-Work* (*2P-PoW*) [32]. Table Ia summarises our comparisons.

Non-outsourceable scratch-off puzzle. Miller et al. [8] formalises cryptocurrency mining as *Scratch-off puzzles*, defines *Non-outsourceable scratch-off puzzles*, and proposes two constructions. One of these two constructions achieves **Weak non-outsourceability** (i.e., miners can steal the mining reward), and the other achieves **Strong non-outsourceability** (i.e., miners can anonymously steal the mining reward).

In the weak non-outsourceable scratch-off puzzle, mining consists of three steps. First, the miner creates a Merkle tree randomly. Second, the miner searches for a valid nonce. More specifically, the miner samples some leaves of the Merkle tree randomly according to nonce, and hashes their Merkle proofs together to produce a hash. If this hash meets the difficulty requirement, the nonce is considered valid. Last, the miner binds the valid nonce and his block template together to produce a valid block.

In order to outsource the mining process, the mining pool should distribute the search space of nonces to miners. Miners can steal the mining reward by repetitively searching for a valid nonce (the first two steps) then binds this valid nonce with his own block template (the last step). However, in this protocol, all miners with the same view of the blockchain share the same search space of nonces, as the search space of nonces only relies on the previous block hash, rather than both the previous block hash and the block template like in Bitcoin. By exploiting this fact, the pool operator can identify the miner who steals the mining reward. For example, the pool operator can link the nonce in the stolen block with the miner who takes charge of the search space covering this nonce. To achieve **Strong non-outsourceability** (i.e., make the stealing behaviours anonymous), the strong non-outsourceable scratch-off puzzle replaces the plaintext nonce in the block with a Zero Knowledge Proof proving the statement "I know a valid nonce".

As discussed in §IV, while being simpler and more efficient than non-outsourceable scratch-off puzzle, our VRF-based mining achieves **Strong non-outsourceability**.

2P-PoW. Eyal and Sirer proposes 2P-PoW [32], a mining protocol that discourages pooled mining. In 2P-PoW, there are two phases, and each phase has a difficulty parameter. A miner should find a nonce that makes the block to pass two phases: 1) the Sha256d hash of the block meets the first difficulty, 2) the Sha256 hash of the signature of the block meets the second difficulty. As the second requirement requires the private key, pool operators cannot outsource the second phase.

Compared to VRF-based mining that makes pooled mining impossible, 2P-PoW is partially outsourceable, as its first

TABLE I: Comparison with related work.

(a) Comparison between mining protocols.

	VRF-based mining	Puzzle-1 [8]	Puzzle-2 [8]	2P-PoW [32]
Punishment	New reward	New reward	New reward	New reward
Stealing	Unlinkable	Linkable	Unlinkable	Unlinkable
No partial outsourcing	✓	✓	✓	✗
Support randomised signatures	✓	✓	✓	✗
No complex cryptography	✓	✓	✗	✓

(b) Comparison with decentralised mining pools.

	VRF-based mining	P2Pool [33]	SmartPool [34]	BetterHash [35]
Complexity	-	Blockchain	Smart contract	-
Decentralisation	Mining	Mining	Mining	Select txs

phase is outsourceable. In addition, 2P-PoW should use deterministic digital signatures, while commonly used digital signatures (e.g., ECDSA, EdDSA) rely on randomisation. If the signature is non-deterministic, the pool operator can make use of all nonces that are generated by miners and meet the first difficulty. For example, given a nonce meeting the first difficulty, the pool operator repetitively generates signatures to meet the second requirement. Moreover, how to adjust two difficulties still remains unknown and requires some simulations.

B. Decentralised mining pools

Decentralised mining pool is a type of mining pool that works in a decentralised way. More specifically, miners mine in the name of themselves rather than the pool operator, but they share reward in a fine-grained way. In this way, miners are rewarded stably while mining power is not aggregated to pool operators. Table Ib summarises the comparison between VRF-based mining with decentralised mining pool approaches.

P2Pool [33] is a decentralised mining pool for Bitcoin. P2Pool runs a share-chain among all miners in the pool, and the share-chain includes shares submitted to the pool in sequence. During mining, the coinbase transaction records the number of shares submitted by each miner, and distributes the mining reward according to miners' contribution. In this way, once mining a Bitcoin block, the coinbase transaction will become valid, and miners will be rewarded according to their contribution. However, P2Pool suffers from several challenges. First, handling the difficulty of mining shares in P2Pool is hard. If the difficulty is high, a miner's reward will still be volatile. If the difficulty is low, there will be numerous low-difficulty shares, which introduces huge overhead on broadcasting shares or even spamming attacks. Second, frequent share submissions amplifies the influence of network latency which leads to high orphan share rate. Last, P2Pool is vulnerable to temporary dishonest majority [36].

SmartPool [34] is another decentralised mining pool, which uses a smart contract to replace the centralised pool operator. As relying on smart contracts, SmartPool cannot work on blockchains without smart contracts. In addition, as blockchains achieve limited throughput, transaction processing might be congested and mining reward might be delayed, especially when a large number of miners participate in the SmartPool. Moreover, as the SmartPool smart contract should verify the validity of blocks, miners should submit the whole block (with transactions) to the SmartPool. Verifying blocks introduces significant overhead on computing (so expensive transaction fees), and storing blocks in the SmartPool smart contract also introduces significant overhead on storing the blockchain.

BetterHash [35] is another decentralised mining protocol, which has been integrated into **Stratum V2** [37], the next generation of the *Stratum* [38] pooled mining protocol. In *BetterHash*, the block operator allows miners to choose transactions and construct blocks in his name, rather than constructing block templates by himself. Thus, *BetterHash* only contributes to the decentralisation of constructing blocks, but does not contribute to the decentralisation of mining power.

X. Conclusion and future work

In this paper, we propose VRF-based mining, that can make pooled mining in Proof-of-work-based consensus impossible. VRF-based mining is simple and intuitive: Miners produce hashes of blocks using VRFs rather than hash functions, so a pool operator should reveal his private key to outsource the mining process to other miners.

As aforementioned, ruling out pooled mining can achieve better decentralisation but may harm the incentive of mining. How to achieve decentralisation while preserving the incentive of mining is still an open challenge.

References

[1] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Annual International Cryptology Conference*. Springer, 1992, pp. 139–147.

[3] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Communications of the ACM*, vol. 61, no. 7, pp. 95–102, 2018.
[4] "Pool stats - btc.com," last accessed on 24/11/19. [Online]. Available: https://btc.com/stats/pool
[5] J. Yu, D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo, "Repucoin: Your reputation is your power," *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1225–1237, 2019.
[6] R. Han, Z. Sui, J. Yu, J. Liu, and S. Chen, "Sucker punch makes you richer."
[7] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*. IEEE, 1999, pp. 120–130.
[8] A. Miller, A. Kosba, J. Katz, and E. Shi, "Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 680–691.
[9] S. Goldberg, D. Papadopoulos, and J. Vcelak, "draft-goldbe-vrf: Verifiable random functions.(2017)," 2017.
[10] N. Van Saberhagen, "Cryptonote v 2.0," 2013.
[11] G. Maxwell, "Coinjoin: Bitcoin privacy for the real world," in *Post on Bitcoin forum*, 2013.
[12] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 486–504.
[13] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 345–364.
[14] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "Tumblebit: An untrusted bitcoin-compatible anonymous payment hub," in *Network and Distributed System Security Symposium*, 2017.
[15] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, 2012.
[16] S. Scott, N. Sullivan, and C. A. Wood, "Hashing to elliptic curves," Internet-Draft draft-irtf-cfrg-hash-to-curve-03. Internet Engineering Task ..., Tech. Rep., 2019.
[17] T. Icart, "How to hash into elliptic curves," in *Annual International Cryptology Conference*. Springer, 2009, pp. 303–316.
[18] M. Ulas, "Rational points on certain hyperelliptic curves over finite fields," *arXiv preprint arXiv:0706.1448*, 2007.
[19] E. Brier, J.-S. Coron, T. Icart, D. Madore, H. Randriam, and M. Tibouchi, "Efficient indifferentiable hashing into ordinary elliptic curves," in *Annual Cryptology Conference*. Springer, 2010, pp. 237–254.
[20] D. J. Bernstein, M. Hamburg, A. Krasnova, and T. Lange, "Elligator: Elliptic-curve points indistinguishable from uniform random strings," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 967–980.
[21] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2013, pp. 313–314.
[22] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, "Sha-3 proposal blake," *Submission to NIST*, vol. 92, 2008.
[23] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.
[24] Monero - secure, private, untraceable. [Online]. Available: https://www.getmonero.org
[25] E. Wiki, "Ethash," *GitHub Ethereum Wiki. https://github.com/ethereum/wiki/wiki/Ethash*, 2017.
[26] M. J. Seigen and T. Nieminen, "Neocortex, and antonio m. juarez. cryptonight hash function. cryptonote, march 2013."
[27] Amazon ebs–optimized instances. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-optimized.html
[28] Cryptocurrency mining profitability calculator. [Online]. Available: https://www.coinwarz.com/cryptocurrency
[29] P. Szalachowski, D. Reijsbergen, I. Homoliak, and S. Sun, "Strongchain: Transparent and collaborative proof-of-work consensus," *arXiv preprint arXiv:1905.09655*, 2019.
[30] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, 2019, pp. 95–112.
[31] C. Li, P. Li, D. Zhou, W. Xu, F. Long, and A. Yao, "Scaling nakamoto consensus to thousands of transactions per second," *arXiv preprint arXiv:1805.03870*, 2018.
[32] E. Ittay and E. Sirer, Gün, "How to disincentivize large bitcoin mining pools," 2014. [Online]. Available: http://hackingdistributed.com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools/
[33] F. Voight, "p2pool: Decentralized, dos-resistant, hop-proof pool," 2011.
[34] L. Luu, Y. Velner, J. Teutsch, and P. Saxena, "Smartpool: Practical decentralized pooled mining," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1409–1426.
[35] M. Corallo, "Betterhash mining protocol(s)," 2019, last accessed on 08/12/19. [Online]. Available: https://github.com/TheBlueMatt/bips/blob/betterhash/bip-XXXX.mediawiki
[36] A. Zamyatin. Decentralized mining pools: Security and attacks. [Online]. Available: https://www.alexeizamyatin.me/files/Decentralized_Mining-Security_and_Attacks.pdf
[37] "Stratum v2 — the next generation protocol for pooled mining," 2019, last accessed on 08/12/19. [Online]. Available: https://stratumprotocol.org/
[38] "Stratum mining protocol - bitcoin wiki," 2015, last accessed on 08/12/19. [Online]. Available: https://en.bitcoin.it/wiki/Stratum_mining_protocol

APPENDIX

Algorithm 3 describes the Elliptic-curve-based VRF (EC-VRF) construction standardised in draft-goldbe-vrf [9].

Algorithm 3: The Elliptic-curve-based VRF (EC-VRF) construction standardised in draft-goldbe-vrf [9].

Preliminaries:

- G is a cyclic group of prime order q with generator g.
- $H_1(\cdot)$ hashes an arbitrary-length string into an element in G.
- $H_2(\cdot)$ hashes an element in G into a fixed-length string.
- $H_3(\cdot)$ hashes an arbitrary-length string into a fixed-length string.
- $\mathsf{random}([x, y])$ uniformly and randomly picks a number in $[x, y]$.

Algorithm $(sk, pk) \leftarrow \mathsf{VRFKeyGen}\,(1^\lambda)$ **:**

$sk = \mathsf{random}([0, q-1])$
$pk = g^{sk}$
return (sk, pk)

Algorithm $\beta \leftarrow \mathsf{VRFHash}\,(sk, \alpha)$ **:**

$h = H_1(\alpha)$
$\gamma = h^{sk}$
$\beta = H_2(\gamma)$
return β

Algorithm $\pi \leftarrow \mathsf{VRFProve}\,(sk, \alpha)$ **:**

$h = H_1(\alpha)$
$\gamma = h^{sk}$
$k = \mathsf{random}([0, q-1])$
$c = H_3(g, h, g^{sk}, h^{sk}, g^k, h^k)$
$s = k - c \cdot sk \pmod{q}$
$\pi = (\gamma, c, s)$
return π

Algorithm $\{0, 1\} \leftarrow \mathsf{VRFVerify}\,(pk, \alpha, \beta, \pi)$ **:**

$u = pk^c \cdot g^s$
$h = H_1(\alpha)$
if $\gamma \notin G$ **then**
  return 0
end
$v = \gamma^c \cdot h^s$
if $c \neq H_3(g, h, pk, \gamma, u, v)$ **then**
  return 0
end
return 1