VRF-Based Mining
Simple Non-Outsourceable Cryptocurrency Mining
(Last Update: August 23, 2020)

Runchao Han[1,2], Haoyu Lin[3,4], and Jiangshan Yu[1]

[1] Monash University
[2] CSIRO-Data61
[3] Bytom Foundation
[4] KZen Research

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 its secret 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 VRF-based mining achieves strong *non-outsourceability* guarantee. In addition, we discuss several considerations on instantiating VRFs for VRF-based mining. Moreover, we experimentally show that VRF-based mining is simple to implement and introduces negligible overhead compared to hash-based mining.

1 Introduction

Bitcoin [27] started the era of cryptocurrency. In Bitcoin, each participant maintains its own ledger. The ledger is structured as a blockchain, i.e., a chain of blocks of transactions. Participants keep executing Proof-of-Work (PoW)-based consensus to agree on blocks and append them to the blockchain. In PoW-based consensus, each participant keeps trying to solve a computationally hard PoW puzzle [17]. Once solving the puzzle, the participant appends its block to the blockchain and gets some coins as reward. By adaptively adjusting the difficulty of PoW puzzles, PoW-based consensus can ensure only one participant solves the puzzle for each time period with high probability. 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.

Due to the limited rate of producing blocks, a miner's reward can be highly volatile if it chooses to mine by himself. To stabilise mining rewards, miners usually choose to join mining pools. 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. While miners are willing to participate in mining pools, mining pools lead to mining power centralisation. Nowadays, four largest mining pools control more than 51% Bitcoin

mining power [5], which can be a lurking threat to blockchain ecosystem. However, mining power centralisation contradicts Bitcoin's design objective – decentralisation, and weakens PoW-based consensus' security [27][18]. To avoid this, researchers have been working on re-decentralising mining power, mainly by non-outsourceable mining [26, 21] and decentralised mining pools [33][24][16]. However, existing approaches are either inefficient or ineffective, which will be discussed in §9 later.

This paper introduces *VRF-based mining*, a surprisingly simple but effective approach to make pooled mining impossible. VRF-based mining replaces hash functions in PoW-based consensus with Verifiable Random Functions (VRFs) [25]. VRF requires a miner to input its secret key to compute the output, so a pool operator should reveal its secret key to miners in order to outsource mining. Thus, any miner can steal mining reward of a block owned by the pool operator anonymously. Our contributions are as follows.

We analyse the non-outsourceability of VRF-based mining. Miller et al. [26] first formalises the *non-outsourceability* of mining in PoW-based consensus. We find *non-outsourceability* consists of two orthogonal properties, namely *punish-mining-reward* and *stealing-unlinkability*. *Punish-mining-reward* specifies that miners can steal mining reward if the pool operator outsources mining. *Stealing-unlinkability* specifies that the pool operator cannot know who steals mining reward. We then show that VRF-based mining achieves *punish-mining-reward*, but not *stealing-unlinkability*. Nevertheless, the pool operator should take extra effort – which makes maintaining mining pools expensive – to identify mining reward stealers and transfer mining reward before being stolen.

We discuss considerations for instantiating VRF-based mining. VRF has four customisable components, namely the elliptic curve, two hash functions between strings and elliptic curve elements, and a normal hash function. We discuss considerations on choosing them for VRF-based mining.

We evaluate feasibility of VRF-based mining. We implement Elliptic curve-based VRF specified in [19] using Go programming language, and compare its performance with three mainstream mining algorithms SHA256D, Scrypt and CryptoNight. Our results show that VRF-based mining is easy to implement and introduces negligible overhead compared to hash-based mining.

We show that partial outsourcing 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.

This paper is organised as follows. Section 2 and Section 3 describes preliminaries and the construction of VRF-based mining, respectively. Section 4 revisits the definition of non-outsourceability, and shows VRF-based mining achieves better non-outsourceability than existing proposals. Section 5 discusses concerns of instantiating VRFs for VRF-based mining. Section 6 provides an experimental analysis on the practicality of VRF-based mining. Section 7 discusses why partial outsourcing in VRF-based mining is unprofitable. Section 8 discusses potential problems in VRF-based mining and their solutions. Section 9 summaries

related work, and Section 10 concludes this paper. We attach the pseudocode of the standardised EC-VRF construction [19] in Appendix.

2 Preliminaries

2.1 PoW-based consensus and mining

Miners participate in PoW-based consensus by *mining*, which works as follows. Given the blockchain's latest view, the miner creates a block template t, which consists of transactions to include, hash of the last block, and other metadata. The miner then tries to solve a PoW puzzle that, the miner should find a nonce n such that $H(t||n)$ is smaller than a difficulty parameter T. Note that $H(\cdot)$ is a one-way hash function, and the miner can only try different nonces until finding a valid one. Once the miner find a valid nonce, it can assemble this nonce and t to a block, and append this block to the blockchain. The miner will get some coins as the reward of mining. By adaptively adjusting T, the rate of new blocks remains stable. If miners mine too many blocks in a previous time period, the protocol will increase T so that miners will mine slower. Otherwise, the protocol will decrease T.

2.2 Mining pools

As the rate of new blocks is limited, the mining reward of miners is highly volatile, especially for miners with less powerful hardware. To stabilise mining reward, miners usually participate in mining pools. Mining pools enable miners to mine collaboratively and distribute reward in a fine-grained way. A mining pool usually relies on a centralised pool operator. All miners mine in the name of the pool operator, and the pool operator distributes rewards in proportion to miners' contribution.

Concretely, a mining pool works as follows. 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.

2.3 Verifiable random functions

Verifiable Random Function (VRF) [25] is a public-key version of cryptographic hash function. In addition to the input string, VRF requires a pair of a secret

key and a public key. Given an input string and a secret key, one can compute an output. Anyone knowing the associated public key can verify the correctness of the output, and can also verify the output is generated by the owner of the secret key. Formally, a VRF consists of four algorithms: $\mathsf{VRFKeyGen}$, $\mathsf{VRFHash}$, $\mathsf{VRFProve}$ and $\mathsf{VRFVerify}$.

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

Informally, VRF should preserve the following three security properties [19]:

- *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 [19].

3 VRF-based mining

Instead of using a hash function, VRF-based mining uses a VRF to produce outputs that satisfy the difficulty requirement. VRF takes both an input and a secret key to produce an output. The owner of this secret key can produce a proof proving the ownership of its output. Thus, to outsource the mining process, the pool operator has to reveal its secret key to miners. However, with the secret key, any miner in the mining pool can steal all mining rewards.

Cryptocurrency mining consists of two components, namely mining blocks and verifying blocks. We call the process of mining a block $\mathsf{Work}(\cdot)$, and the process of verifying a block $\mathsf{Verify}(\cdot)$. Algorithm 1 and 2 describe $\mathsf{Work}(\cdot)$ and $\mathsf{Verify}(\cdot)$ of VRF-based mining, respectively.

Work. In $\mathsf{Work}(\cdot)$, the miner - with secret key sk and block template tpl - keeps searching for a *nonce* that can make the VRF output *out* of the block *blk* to meet the blockchain difficulty T. Once finding a valid *nonce* that makes $out < T$, the miner produces proof π, assembles *blk*, pk, *out*, π to the complete block B. The miner then broadcasts B to the network.

Algorithm 1: $\mathsf{Work}(sk, pk, tpl, T)$.

Input: The secret key sk, the block template tpl, and the blockchain difficulty T
Output: The complete block B
while $nonce+ = 1$ **do** ▷ Refresh the nonce
  $blk \leftarrow \mathsf{ConstructBlock}(tpl, nonce)$ ▷ Assemble block
  $out \leftarrow \mathsf{VRFHash}(sk, blk)$ ▷ Produce VRF output
  if $out < T$ **then** ▷ If meeting difficulty
    break ▷ Mining successful
  end
end
$\pi \leftarrow \mathsf{VRFProve}(sk, blk)$ ▷ Produce the proof
$B \leftarrow (blk, pk, out, \pi)$ ▷ Assemble the complete block
return B

Algorithm 2: $\mathsf{Verify}(B, T)$

$(blk, pk, out, pi) \leftarrow B$ ▷ Disassemble block
Require B's coinbase tx is binding to pk
$\mathsf{require}(out < T)$ ▷ Check if satisfying diff
/* Here $\mathsf{VRFVerify}(\cdot)$ ensures: */
/* 1. out is generated by the owner of sk */
/* 2. out is a valid output of $\mathsf{VRFHash}(sk, blk)$ */
$\mathsf{require}(\mathsf{VRFVerify}(pk, blk, out, \pi))$
... ▷ Verify other fields
... ▷ Verify transactions

Verify. Upon a new block B, the verifier runs $\mathsf{Verify}(\cdot)$ to verify B's validity. In addition to verification rules in hash-based mining, $\mathsf{Verify}(\cdot)$ specifies three extra rules. First, the VRF output out should be smaller than the difficulty parameter T. Second, B's coinbase transaction should be binding to pk. If the transaction's output stores the public key, then it should equal to pk. If the transaction's output stores the address – which is usually a digest of the public key, then it should equal to the digest of pk. Last, the verifier should run $\mathsf{VRFVerify}(\cdot)$ to check if 1) out is produced by the owner of pk, and 2) out is a valid VRF output of blk.

4 Non-outsourceability analysis

In this section, we revisit existing definitions on *non-outsourceability*, and show that VRF-based mining achieves strong *non-outsourceability*.

4.1 Revised definitions

Miller et al. [26] first formalise cryptocurrency mining as *scratch-off puzzles*, and formally 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 stealers defined in [26] is a special case of *transaction unlinkability* [32]: Given two arbitrary transactions, it is hard to know whether their outputs belong to the same secret key. To steal the mining reward in [26], the miner should create a transaction, of which the input is the mining reward and the output points to its 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*.

4.2 Non-outsourceability of VRF-based mining

VRF-based mining achieves *punish-mining-reward*, but not *stealing-unlinkability*. To outsource mining to a miner, the pool operator should share a secret key with it. This enables the miner to steal the mining reward, so VRF-based mining achieves *punish-mining-reward*. The pool operator can give a new secret key and a block template to each miner at every height. If a miner steals mining reward, the pool operator can identify the stealer by the secret key in the transaction stealing the reward. Thus, VRF-based mining does not satisfy *stealing-unlinkability*.

Nevertheless, VRF-based mining makes maintaining mining pools quite expensive. First, the pool operator should monitor stealing behaviours by examining new blocks' coinbase transactions all the time. In addition, every time the pool operator's miners mine a block, the pool operator should transfer mining reward to its own wallet before a miner steals it.

5 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. We discuss considerations on choosing these four components for VRF-based mining.

5.1 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, Edwards25519 [11] (birationally equivalent to Curve25519 [10]) can be a promising choice. First, Edwards25519 works on a prime field with the prime number $2^{255} - 19$, which provides sufficient enumeration space for VRF. Second, Edwards25519 supports Ed25519, a fast and secure digital signature algorithm. Last, Edwards25519 is compatible with existing blockchains, as numerous blockchains and projects using VRF adapt Edwards25519 as their underlying elliptic curve [3][6][4].

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

The first hash function $H_1(\cdot)$ hashes a string to an elliptic curve point. A standardisation document [28] specifies several hash-to-curve functions [20] [31] [15] [12] that fit into different elliptic curves and satisfy our requirements. Within these functions, Elligator2 [12] is the recommended one for Ed25519.

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

The second hash function $H_2(\cdot)$ 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.

5.4 Normal hash function $H_3(\cdot)$

The third hash function $H_3(\cdot)$ hashes an arbitrary string to a fixed-length string. It is only used in $\mathsf{VRFProve}(\cdot)$ and $\mathsf{VRFVerify}(\cdot)$ i.e., producing and verifying proofs of VRF outputs. The overhead of proving and verification should be minimised, so fast and cryptographically secure hash functions such as Keccak [13] and BLAKE [9] are suitable for $H_3(\cdot)$.

5.5 Memory-hard mining

Some PoW-based blockchains — such as Ethereum [36] and Monero [4] — employ memory-hard hash functions in order to minimise advantage of specialised mining hardware and achieve better decentralisation. Different from normal hash functions, the performance of computing memory-hard hash functions is bounded by memory access rather than arithmetic operations in processors. As memory access is much harder to optimise compared to arithmetic operations in processors, high-end hardware cannot outperform low-end hardware greatly in terms of memory-hard functions.

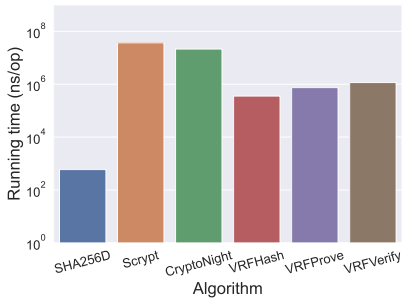
To make VRF-based mining memory-hard, $\mathsf{VRFHash}(\cdot)$ should be memory-hard. $\mathsf{VRFHash}(\cdot)$ of the standardised VRF consists of one $H_1(\cdot)$ hashing, one scalar-point multiplication and one $H_2(\cdot)$ hashing. By making $H_1(\cdot)$ or $H_2(\cdot)$ memory-hard, VRF-based mining will be memory-hard. This can be achieved by embedding a memory-hard hash function such as Ethash [35] and CryptoNight [29] in $H_1(\cdot)$ or $H_2(\cdot)$.

6 Practicality of VRF-based mining

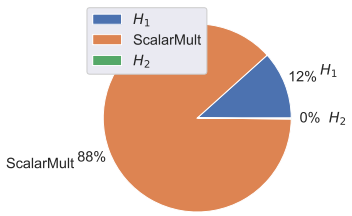
In this section, we benchmark VRFs and compare their performance with existing hash functions for mining. The experimental results show that VRF-based mining is easy to implement and introduces negligible overhead compared to hash-based mining.

6.1 Experimental setting

We implement the standardised EC-VRF in Algorithm 3 using Go programming language, without any optimisation. We use Ed25519 [11] as the underlying elliptic curve, Elligator [12] 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 use open-source implementations of SHA256D [5], Scrypt [6], and CryptoNight [7]. All experiments run on a MacBook Pro with a 2.2 GHz Intel Core i7 Processor and a 16 GB DDR4 RAM. Each group of experiments consists of ten runs, and we take the average value of ten values as the result.



(a) Comparing the runtime of VRF and other mining algorithms.



(b) Runtime breakdown of $\mathsf{VRFHash}(\cdot)$.

Fig. 1: Evaluation of VRF.

[5] https://github.com/seehuhn/sha256d
[6] https://github.com/elithrar/simple-scrypt
[7] https://github.com/Equim-chan/cryptonight

6.2 VRF v.s. existing mining algorithms

We compare the performance of VRF with existing mining algorithms. Figure 1a shows the runtime of VRF and other algorithms. SHA256D is much faster than other algorithms, as SHA256D is simply executing SHA256 twice. In addition, $\mathsf{VRFVerify}(\cdot)$ is slightly slower than $\mathsf{VRFProve}(\cdot)$, and $\mathsf{VRFProve}(\cdot)$ is slightly slower than $\mathsf{VRFHash}(\cdot)$. Last, although without optimisation, all functions of VRF are much faster than Scrypt and CryptoNight.

6.3 Runtime breakdown of VRF

We profile $\mathsf{VRFHash}(\cdot)$ by evaluating its runtime of each step. Figure 1b shows that, the elliptic curve scalar multiplication $\gamma \leftarrow h^{sk}(\cdot)$ takes 88% of $\mathsf{VRFHash}(\cdot)$'s running time. This is because we use SHA-3 as the hash function in $H_1(\cdot)$ 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 $H_2(\cdot)$. There have been optimisation techniques for elliptic curve scalar multiplication, and some miners may exploit them for accelerating mining. This may be unfair to other miners. To avoid this, we suggest to replace $H_1(\cdot)$ and/or $H_2(\cdot)$ with slow hash functions such as Scrypt and CryptoNight.

7 Profitability of partial outsourcing

The pool operator may still have opportunity to partially outsource mining. The mining function $\mathsf{VRFHash}$ consists of three steps: 1) $h \leftarrow H_1(\alpha)$, 2) $\gamma \leftarrow h^{sk}$ and 3) $\beta \leftarrow H_2(\gamma)$. *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 $h = H_1(\alpha)$ and the last step $\beta = H_2(\gamma)$ by distributing different αs and γs to miners, respectively. However, we show that such partial outsourcing is very inefficient and unprofitable compared to outsourcing in hash-based mining, due to the computing and I/O overhead.

7.1 Partial outsourcing

In VRF-based mining, the pool operator can outsource $H_1(\cdot)$ or $H_2(\cdot)$. To outsource $H_1(\cdot)$, the pool operator generates a series of nonces $\{n_1, \ldots, n_m\}$, then sends the series and the block template t to a miner. Then, the miner computes $H_1(n_i)$ for each $i \in [1, m]$, then sends back all $h_i = H_1(\cdot)$ hashes to the pool operator. The pool operator should verify $\{h_1, \ldots, h_m\}$ before starting the second step — multiplying each of $\{h_1, \ldots, h_m\}$ with its secret key sk.

After the second step, the pool operator obtains a series of $\{\gamma_1, \ldots, \gamma_m\}$. To outsource $H_2(\cdot)$, the pool operator first sends $\{\gamma_1, \ldots, \gamma_m\}$ as well as the pool difficulty PT to the miner. Then, the miner calculates $H_2(\gamma_i)$ for each $i \in [1, m]$, compare the hashes with PT, and sends back γs that satisfy the difficulty (denoted as Γ). Upon receiving these γs, the pool operator verifies their correctness and accumulates the mined shares to the miner's total contribution.

7.2 First obstacle: overhead of verification

The first obstacle of partial outsourcing is verifying hashes from miners. For outsourcing $H_1(\cdot)$, the pool operator should verify all of $\{h_1, \ldots, h_m\}$. For outsourcing $H_2(\cdot)$, The pool operator should verify σs satisfying PT. Such verification overhead is even more than the pool operator mining by himself. Thus, partial outsourcing unprofitable.

7.3 Second obstacle: overhead of I/O

To bypass the first obstacle, the pool operator and the miners can trust each other and omit the verification. This is possible as pooled mining is beneficial for them: the pool operator can earn some fees from the miners, while miners can stabilise their mining reward.

However, partial outsourcing can still be unprofitable even without the verification. VRF-based mining is extremely I/O intensive, and the network bandwidth of the pool operator's server is limited. To outsource a single $H_1(\cdot)$, 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.

Assume a pool operator with bandwidth BW (in bytes/s), and the pool operator can achieve optimal bandwidth utilisation on mining. Assume each the pool operator should transfer N bytes for outsourcing each mining attempt. Then, the maximum hashrate that the pool operator can achieve 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 $= \frac{BW}{32}(h/s)$.

Figure 2 shows relationship between the server's network bandwidth and the maximum hashrate the server can achieve. The result shows that, existing servers can only achieve limited hashrate on partial outsourcing. On AWS [1], I3EN Metal is the server with most bandwidth, which is 3,500 MB/s. However, I3EN Metal can support hashrate 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.

8 Discussions

While eliminating mining pools can improve decentralisation, it also introduces new problems. We discuss two problems and their possible solutions, namely 1) high reward variance and 2) secret key leakage in memory.

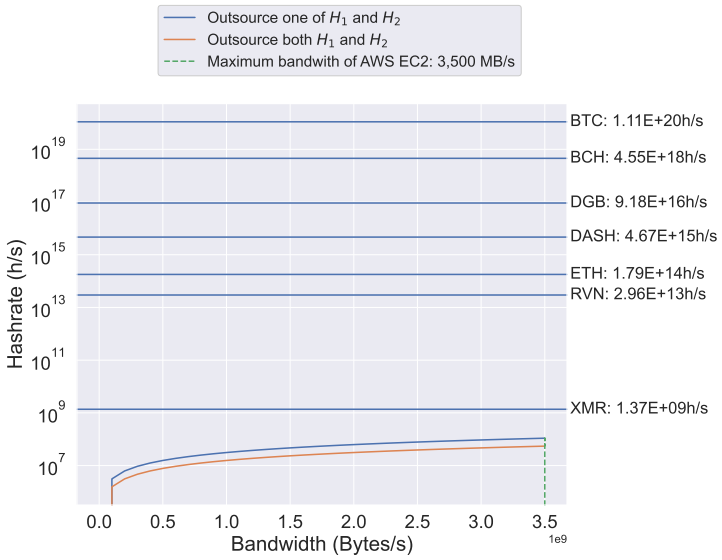


Fig. 2: Server bandwidth v.s. maximum hashrate. Data of server bandwidth and hashrate were fetched from AWS [1] and CoinWarz [2] at 01/03/2020, respectively.

8.1 Weaker security guarantee of PoW-based consensus

The reason of miners joining mining pools is that, miners may not obtain stable income via solo mining. Miners may be discouraged to mine if their rewards are highly volatile. With weaker incentive of mining, fewer miners will join the blockchain. The blockchain will then attract less mining power, which weakens the security of PoW-based consensus.

This problem can be addressed by making the mining reward fine-grained. Miller et al. [26] 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 [30] introduces the notion of *Collaborative PoW*, where miners are encouraged to mine blocks together and the mining reward is distributed in proportion to miners' contributions. Bobtail [14] and HotPoW [22] further explore the Collaborative PoW approach.

Another solution 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, although of independent interest, protocols for scaling blockchains such as sharding [34] and DAG [23] can also stabilise the mining reward variance.

Table 1: Comparison between mining protocols. NSP is short for Non-outsourceable scratch-off puzzle.

	VRF-based mining	NSP-1	NSP-2	2P-PoW
Punish-mining-reward	✓	✓	✓	✓
Stealing-unlinkability	✗†	✗	✓	✗
No partial outsourcing	✓	✓	✓	✗
Support randomised signatures	✓	✓	✓	✗
No complex cryptography	✓	✓	✗	✓

† The pool operator should take non-negligible effort to deanonymise stealing behaviours.

8.2 Secret key leakage in memory

VRF-mining requires the secret key, so a miner should keep its secret key in plaintext in memory *all the time*. This gives adversaries opportunity to steal the secret key from the memory. For example, if the mining software has a bug that enables hackers to access the memory space of the mining software, then the hacker can easily steal the secret key.

For VRF-based mining, keeping secret keys in memory is inevitable. To achieve *non-outsourceability*, the secret key should be combined to the mining process. As long as mining does not stop, the secret key should be kept in plaintext in memory. This applies to all protocols that execute frequently and require secret keys, such as TLS.

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. Second, the process of the mining software can destroy itself once detecting anomalous memory access attempts. Third, the miner can isolate mining from the blockchain node. For example, it can run the mining process independently with its blockchain node process. In this way, if an adversary compromises its node but not the operating system, the adversary still cannot read the secret key in another process. Fourth, it can run the mining process in an offline machine that can only communicate with its blockchain node. Fifth, 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. Last, the miner can use a new secret key for each block, so that leaking a secret key only affects the reward of a single block.

9 Related work

We review related research on preventing mining pools, and compare them with VRF-based mining. We classify related research to two types, namely pooled-mining-unfriendly mining protocols and decentralised mining pools.

Table 2: Comparison with decentralised mining pools.

	VRF-based mining	P2Pool	SmartPool	BetterHash
Complexity	-	Blockchain	Smart contract	-
Decentralisation	Mining	Mining	Mining	Select txs

9.1 Mining protocols

There are two mining protocols aiming at discouraging or breaking mining pools: the *non-outsourceable scratch-off puzzle* [26] and the *Two Phase Proof-of-Work* (*2P-PoW*) [21]. Table 1 summarises our comparisons.

Non-outsourceable scratch-off puzzle. Miller et al. [26] formalises mining as *scratch-off puzzles*, defines *non-outsourceability*, and proposes two *non-outsourceable scratch-off puzzles*. One achieves *weak non-outsourceability* — miners can steal the mining reward, and the other achieves *strong non-outsourceability* — miners can steal the mining reward anonymously.

The weak non-outsourceable scratch-off puzzle works as follows. First, the miner randomly generates a Merkle tree. Second, the miner randomly chooses a nonce, samples some leaves of the Merkle tree according to the nonce, and hashes their Merkle proofs together to a single hash. If this hash meets the difficulty requirement, then the nonce is valid. Last, the miner binds the valid nonce and its block template together to a valid block.

In order to outsource the mining process, the mining pool should distribute the search space of nonces to miners. A miner can steal the mining reward by binding valid nonces it finds with its own block template. However, this stealing process is not anonymous. Unlike existing PoW-based consensus where miners can choose both nonces and block templates, the weak non-outsourceable scratch-off puzzle only allows miners to choose nonces, so all miners share the same search space of nonces. Thus, the pool operator can link the nonce in the stolen block with the miner who is assigned with the search space covering this nonce. To achieve *strong non-outsourceability*, 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 §4, 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 [21], a mining protocol that discourages pooled mining. In 2P-PoW, mining consists of 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 secret key, pool operators cannot outsource the second phase.

Compared to VRF-based mining that makes pooled mining impossible, 2P-PoW is partially outsourceable by outsourcing the first phase. In addition, 2P-PoW requires the digital signature algorithm to be deterministic, while commonly used digital signatures such as ECDSA rely on randomisation. If the

signature is randomised, the pool operator can make use of all nonces from miners that meet the first phase but fail the second phase. 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.

9.2 Decentralised mining pools

Decentralised mining pool is a type of mining pools that work in a decentralised way: miners mine in the name of themselves and share reward in a fine-grained way. In this way, miners are rewarded stably while mining power is not controlled by pool operators. Table 2 summarises the comparison between VRF-based mining with decentralised mining pools.

P2Pool [33] is a decentralised mining pool for Bitcoin. Instead of using a centralised server, P2Pool uses a blockchain called *share-chain* to receive shares from miners. All miners in P2Pool participate in the PoW-based consensus of share-chain. Once a miner finds a share on Bitcoin, it appends a block to the share-chain. The block consists of the share and a coinbase transaction that rewards miners in proportion to their shares. Once a miner mines a block that also satisfies Bitcoin's difficulty, it submits this block to the Bitcoin blockchain, and and miners are rewarded according to the coinbase transaction. P2Pool suffers from several limitations. First, handling the difficulty of mining shares is hard. If the difficulty is high, miners' reward will still be volatile. If the difficulty is low, there will be numerous low-difficulty shares, which introduces huge overhead on broadcasting and receiving shares. Second, shares on share-chain are much more frequent than blocks on Bitcoin blockchain, and the share-chain suffers from high orphan rate. Last, existing research shows that P2Pool is vulnerable to temporary dishonest majority [37].

SmartPool [24] is another decentralised mining pool. Instead of using a blockchain, SmartPool employs smart contracts to track shares from miners. This implies that SmartPool cannot work on blockchains without smart contracts. In addition, as blockchains achieve limited throughput, blockchains can only handle a limited number of shares for each time unit. In this way, distributing mining reward may also 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 entire block — including transactions — to the SmartPool. Verifying blocks introduces non-negligible computing overhead and transaction fee, and storing blocks in smart contracts also introduces non-negligible overhead on storage.

BetterHash [16] is another decentralised mining protocol, which has been integrated into *Stratum V2* [8], the next generation of the *Stratum* [7] pooled mining protocol. In *BetterHash*, the block operator allows miners to choose transactions and construct blocks in its name, rather than constructing block templates by himself. Thus, *BetterHash* only contributes to the decentralisation of constructing blocks, but does not improve the decentralisation of mining power.

10 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 digests of blocks using VRFs rather than hash functions, so that a pool operator should reveal its secret key to outsource the mining process to other miners. We show that VRF-based mining achieves strong *non-outsourceability* guarantee. In addition, we discuss considerations of instantiating VRF-based mining. Moreover, we experimentally proves that VRF-based mining is simple to implement and introduces negligible overhead.

References

1. Amazon ebs–optimized instances, `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-optimized.html`
2. Cryptocurrency mining profitability calculator, `https://www.coinwarz.com/cryptocurrency`
3. Home — ethereum.org, `https://ethereum.org/`
4. Monero - secure, private, untraceable, `https://www.getmonero.org`
5. Pool stats - btc.com, `https://btc.com/stats/pool`, last accessed on 24/11/19
6. Tendermint, `https://tendermint.com/`
7. Stratum mining protocol - bitcoin wiki (2015), `https://en.bitcoin.it/wiki/Stratum\_mining\_protocol`, last accessed on 08/12/19
8. Stratum v2 — the next generation protocol for pooled mining (2019), `https://stratumprotocol.org/`, last accessed on 08/12/19
9. Aumasson, J.P., Henzen, L., Meier, W., Phan, R.C.W.: Sha-3 proposal blake. Submission to NIST **92** (2008)
10. Bernstein, D.J.: Curve25519: new diffie-hellman speed records. In: International Workshop on Public Key Cryptography. pp. 207–228. Springer (2006)
11. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. Journal of Cryptographic Engineering **2**(2), 77–89 (2012)
12. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: Elliptic-curve points indistinguishable from uniform random strings. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 967–980. ACM (2013)
13. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 313–314. Springer (2013)
14. Bissias, G., Levine, B.N.: Bobtail: Improved blockchain security with low-variance mining. In: ISOC Network and Distributed System Security Symposium (2020)
15. Brier, E., Coron, J.S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient indifferentiable hashing into ordinary elliptic curves. In: Annual Cryptology Conference. pp. 237–254. Springer (2010)
16. Corallo, M.: Betterhash mining protocol(s) (2019), `https://github.com/TheBlueMatt/bips/blob/betterhash/bip-XXXX.mediawiki`, last accessed on 08/12/19
17. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Annual International Cryptology Conference. pp. 139–147. Springer (1992)

18. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. Communications of the ACM **61**(7), 95–102 (2018)
19. Goldberg, S., Papadopoulos, D., Vcelak, J.: draft-goldbe-vrf: Verifiable random functions.(2017) (2017)
20. Icart, T.: How to hash into elliptic curves. In: Annual International Cryptology Conference. pp. 303–316. Springer (2009)
21. Ittay, E., Sirer, Gün, E.: How to disincentivize large bitcoin mining pools (2014), `http://hackingdistributed.com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools/`
22. Keller, P., Böhme, R.: Hotpow: Finality from proof-of-work quorums. arXiv preprint arXiv:1907.13531 (2019)
23. Li, C., Li, P., Zhou, D., Xu, W., Long, F., Yao, A.: Scaling nakamoto consensus to thousands of transactions per second. arXiv preprint arXiv:1805.03870 (2018)
24. Luu, L., Velner, Y., Teutsch, J., Saxena, P.: Smartpool: Practical decentralized pooled mining. In: 26th {USENIX} Security Symposium ({USENIX} Security 17). pp. 1409–1426 (2017)
25. Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. In: 40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039). pp. 120–130. IEEE (1999)
26. Miller, A., Kosba, A., Katz, J., Shi, E.: Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 680–691. ACM (2015)
27. Nakamoto, S., et al.: Bitcoin: A peer-to-peer electronic cash system (2008)
28. Scott, S., Sullivan, N., Wood, C.A.: Hashing to elliptic curves. Tech. rep., Internet-Draft draft-irtf-cfrg-hash-to-curve-03. Internet Engineering Task ... (2019)
29. Seigen, M.J., Nieminen, T.: Neocortex, and antonio m. juarez. cryptonight hash function. cryptonote, march 2013
30. Szalachowski, P., Reijsbergen, D., Homoliak, I., Sun, S.: Strongchain: Transparent and collaborative proof-of-work consensus. arXiv preprint arXiv:1905.09655 (2019)
31. Ulas, M.: Rational points on certain hyperelliptic curves over finite fields. arXiv preprint arXiv:0706.1448 (2007)
32. Van Saberhagen, N.: Cryptonote v 2.0 (2013)
33. Voight, F.: p2pool: Decentralized, dos-resistant, hop-proof pool (2011)
34. Wang, J., Wang, H.: Monoxide: Scale out blockchains with asynchronous consensus zones. In: 16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19). pp. 95–112 (2019)
35. Wiki, E.: Ethash. GitHub Ethereum Wiki. https://github.com/ethereum/wiki/wiki/Ethash (2017)
36. Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper **151**(2014), 1–32 (2014)
37. Zamyatin, A.: Decentralized mining pools: Security and attacks, `https://www.alexeizamyatin.me/files/Decentralized\_Mining-Security\_and\_Attacks.pdf`

A The standardised elliptic-curve-based VRF

Algorithm 3 describes the Elliptic-curve-based VRF (EC-VRF) construction standardised in draft-goldbe-vrf [19].

Algorithm 3: The Elliptic-curve-based VRF (EC-VRF) construction standardised in draft-goldbe-vrf [19].

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