Verus Proof of Power

A Provable Hybrid Solution to 51% Hash Attacks

Michael Joseph Toutonghi, Michael Filip Toutonghi, & John Westbrook

Table of Contents

I. Introduction

In 2008, the original Bitcoin white paper described an algorithm called Proof of Work (PoW) [1] used to secure a publicly notarized, decentralized ledger system, independent of any centralized service. Because of the cost of participating in the PoW algorithm by all parties, a majority of participating parties could be assumed, when acting in self-interest, to correctly observe the network protocol, resulting in continuous and accurate processing of public ledger transactions, as long as attackers of the system did not attack with more than 50% of the computing power applied to processing on the network for any extended amount of time. This same PoW algorithm described in 2008 is now widely used to secure most existing cryptocurrencies today. While PoW has served Bitcoin and cryptocurrencies quite well, generally keeping billions of dollars of transactions processing worldwide on a daily basis, its use has also exposed weaknesses for some use cases and to specific kinds of targeted attacks. In fact, as we finish final edits on the first release of this proof sketch, one of the largest attempted 51% attacks in history is underway in the "Hash War" between Bitcoin Cash - ABC and Bitcoin Cash - SV. This paper describes an evolutionary alternative to pure proof of work that supports public validation with PoW, making the choice to participate in the network available to anyone willing to contribute work, while at the same time anchoring a significant portion of the network validation in the self-interest of non-mining network participants, notably holders of the currency for which the network processes transactions.

Verus Proof of Power (PoP) is a hybrid consensus algorithm which uses a statistical function that combines Proof of Work (PoW) and Proof of Stake (PoS) to validate each block by either PoW or PoS, while averaging to a target percentage of blocks being validated by each form of proof. In this document we describe a PoW component of PoP and assume use of the VerusHash algorithm, a long-input hash function based on a Haraka512 V2 [2] core. PoP can, however, be implemented as a hybrid consensus algorithm with VerusPoS combined with virtually any PoW hash algorithm.

Verus Coin is an open source community project working towards Public Blockchains as a Service that leverages the latest advances in zero knowledge privacy and public blockchain security. Verus Coin benefits from lineage in the latest Komodo platform technologies, such as enhanced Komodo Crypto-Conditions technology as well as the latest Zcash Sapling support. In addition to its advanced base security, which will be described herein and proven to be resistant to 51% or even 99% PoW hash attacks, Verus is backed by the dPoW (delayed proof of work) notarization of Komodo and the Bitcoin blockchain. Furthermore, its unique, enhanced implementation of Crypto Conditions also enables Verus Coin to also provide a solution to the PoS issue of nothing-at-stake.

II. VerusHash PoW

VerusHash was developed to deliver a competitive advantage for CPUs with GPUs. It is an exceedingly CPU-friendly long input hash function that uses the quantum-secure, short input Haraka512 V2 as its core compression algorithm. The result is the fastest known cryptocurrency hash algorithm available to

modern CPUs and the only hash algorithm which enables today's CPUs and GPUs to compete on an economically comparable level.

Haraka512 V2 is designed as a short input hash to exclusively consume one chunk of 512 bits and produce 256 bits of a hash result. Utilizing Haraka512 V2 VerusHash takes any length of input and produces a 256 bit hash result, unique to VerusHash, that also provides the same security guarantees as Haraka512 V2. This makes VerusHash 256 bit secure for classical computing attacks and 128 bit secure against quantum computers for pre-image and second pre-image attacks [2].

To understand the VerusHash algorithm it helps to first separate the digest from the core. We then consider the Haraka512 V2 core as an abstract digest function that takes 512 bits (64 bytes) of input and produces 256 bits (32 bytes) of output. Given such a digest function, referred to as haraka512256, the most concise implementation of VerusHash, in any language to-date, is the following Python code for the VerusHash hash digest as follows:

```
# verus_hash
def verus_hash(msg):
    buf = [0] * 64
    length = len(msg)
    for i in range(0, length, 32):
        clen = min(32, length - i)
        buf[32:64] = [b for b in msg[i:i + clen]] + [0] * (32 - clen)
        buf[0:32] = haraka512256(buf)
    return bytes(buf[0:32])
```

As its PoW algorithm, Verus Coin calculates a block hash by applying the VerusHash algorithm to a Verus block header (essentially a Zcash and Komodo compatible header), with a reserved space in place of the Equihash solution. The result of this hash is then compared to the network-wide difficulty target and if the result is less than or equal to the target, the underlying block is submitted.

III. VerusPoS

Part I: General Overview

VerusPoS is a proof-of-stake network block validation algorithm. VerusPoS defines a decentralized competition which operates network-wide in selecting validators for the next block on the Verus blockchain. This is done according to consensus rules, using a network-wide contest similar to Verus PoW. The biggest difference between the VerusPoS contest and the PoW contest is that the winner selection is statistically weighted by the amount of Verus Coin under the control of the staking party, as a ratio of the entire network's staking supply, rather than individual hash power as a ratio of network hash power. The process of determining whether a staker wins the opportunity to submit a block to the network is quite similar to the PoW process in the sense that a mathematical operation involving hashes and

specific inputs produces a random value, the Raw PoS Hash, for each unspent transaction output (UTXO) being staked. This Raw PoS Hash is then divided by the amount of currency in that particular UTXO to produce the PoS Hash for that UTXO. The resultant PoS Hash for that UTXO is then compared to a network-wide PoS difficulty value, which is itself adjusted at each block to maintain an average percentage of blocks being validated by PoS at 50%.

In summary, VerusPoS goes through the process of creating block candidates with transactions, running the hashing and division operation across all UTXOs on all staking wallets across the network, comparing those results against the PoS difficulty and submitting winners at each block to the network for further validation and selection of best chain by consensus. This process creates a statistical contest across all UTXOs staking on the network with one objective winner across the entire network in virtually all cases that a PoS block wins. Once submitted, the winning submission for that block height, whether PoS or PoW, is chosen network-wide based on the consensus of the chain with most cumulative Chain Power, a combined value of PoS and PoW, explained in more detail in "Section V: Proof of Power" below.

VerusPoS is not a complete and independent consensus algorithm by itself, as it lacks a network-wide clock function. This must be supplied by either an oracle or, as in the case of the Verus Coin blockchain, the total progression of block validation, including PoW, which ensures a predictable, monotonically increasing blockchain progression, even in the 50% of cases in which no staking UTXO on the network validates a block. Since the PoW and VerusPoS algorithms operate in parallel on the Verus Coin blockchain, it is critical to understand the relationship between PoW block validation and VerusPoS block validation in order to understand how VerusPoS strengthens security during potential attacks, rather than create additional instabilities with which to contend.

VerusPoS uses the validation of a block on the blockchain as its only measure of the passage of time. While it could adapt to a variety of difficulty adjustment algorithms (DAAs) for PoS, the VerusPoS DAA increases its measure of the passage of time for each block when there are consecutive PoW blocks validated and decreases this measure for each block when there are consecutive VerusPoS blocks validated. This ensures that consecutive blocks of either form of validation, PoW or VerusPoS, are statistically discouraged.

Since VerusPoS has no independent measure of time besides the validation of blocks on the blockchain and since the time between blocks of PoW blocks on the blockchain is determined by the hash rate, attacks that may speed up block validation by instantaneous addition of hash rate then slow down upon a hash rate decrease do not affect PoS ratio or progress relative to PoW in an ideal VerusPoS implementation. This is important, as this property ensures that PoS block validation remains stable and independent during any attempted hash attack, continuing to validate its target percentage of PoS blocks, whether at a faster or slower real-time rate. PoW difficulty is managed separately with a target block time for the total of all validated blocks on the network, since every block could conceivably be either PoW or PoS. Adjustment of the PoW difficulty is therefore the only adjustment necessary to compensate for increases or decreases in block time as measured in units of time, due to changes in network hash rate.

To prevent any ability of the staking party to influence their chances of successfully staking a block by generating UTXOs or affect other input values that could influence their outcome, VerusPoS defines specific rules of eligibility for UTXOs as well as hashed entropy sources that ensure the staker cannot intentionally influence current or future staking results in order to gain unfair advantage over other miners or stakers on the network. Specifically, the rules are:

1. Staking UTXOs must be transparent pay-to-public-key or pay-to-public-key-hash.
2. UTXOs become valid for staking once they have 150 or more confirmations. While another number could have been chosen, the number should be greater than the number of confirmations required on any network using VerusPoP. This ensures that the UTXO used to qualify for a staking operation was finalized prior to additional entropy in the PoS hash result that is outside of the control of the staking party.
3. Valid staking UTXOs must have output values greater than the smallest transaction amount allowed by the network.

To determine if an eligible UTXO also qualifies as the winning stake required to successfully submit a block of transactions to the network, the data from the transaction is first combined through hash operations with other related data as well as unrelated entropy into a random, yet deterministic Nonce value, as described in *fig. 1,* below. The Nonce, as well as additional data is further combined to produce the PoS Hash for that UTXO, which ultimately determines the eligibility of that UTXO as being the stake source for a block qualified to be submitted to the network. Once it is determined that a specific UTXO qualifies as a winning stake, proof-of-control over that stake is added to a PoS validated block by spending the UTXO back to its original address as the last transaction of the block, resetting the age of the qualifying stake and containing specific additional meta-data which identifies the transaction as a proof-of-stake transaction and binds that stake to the prior block of the blockchain, which also contributes to the VerusPoS solution to the nothing-at-stake problem.

Part II: PoS Hash and PoS Nonce Construction

To define the construction of the PoS Hash of any UTXO, we must first define the following values. Each of the values below contribute to the proof of stake Nonce value and/or the final PoS hash used in the network wide competition. Each has an important role in contributing to the PoS Hash and the quality of its random, deterministic value that cannot be manipulated to a participant's advantage. Where the function Hash(input) is used, VerusHash may be assumed:

Chain magic - *magic* = 32 *bit pseudorandom number assigned to chain*

UTXO TxId - *txid* = *Sha*256 *hash of transaction that holds UTXO*

UTXO voutNum - *vout* = *integer index of UTXO*

UTXO Value - *value* = *integer index of UTXO*

ZeroBytes - $zeros = f : I_n \rightarrow O,\ i \mapsto o_i,\ \{0 < i < 17,\ o_i = 0\}$

PoW target - $Target_{pow}$ = 32 *bit compact representation of* 256 *bit PoW target*

PoS target - $Target_{pos}$ = 32 *bit compact representation of* 256 *bit PoS target*

New block height - $Height_{new}$ = *integer index of new potential block in blockchain*

Block type - $Type_{block}(Height)$ = "PoW" *or* "PoS"

PoW hash - $Hash_{pow}$ = $Hash(blockheader)$

Past hash - $Hash_{past}$ = *If* $Type_{block}(Height_{new} - 100)$ *is* "PoW", $Hash_{pow}$, *else* $Hash_{raw}$

Entropy hash - $Hash_{entropy}$ = *bytes* 5 *through* 16 *of* $Hash(Hash_{past}, txid, vout)$

PoS Intermediate - $Intermediate_{pos}(byte\ array) = M : I_n \rightarrow B,\ i \mapsto b_i,\ \{i \in I_n,\ b_i \in B\}$

- $\{b_1 ... b_4\}$ = $Target_{pos}$
- $\{b_5 ... b_{16}\}$ = $Hash_{entropy}$
- $\{b_{17} ... b_{32}\}$ = *zeros*

PoS Nonce - $Nonce_{pos}(byte\ array)$ = $N : I_n \rightarrow C,\ i \mapsto c_i,\ \{i \in I_n,\ c_i \in C\}$

- $\{c_1 ... c_4\}$ = $Target_{pos}$
- $\{c_5 ... c_{16}\}$ = $Hash_{entropy}$
- $\{c_{17} ... c_{32}\}$ = *least significant* 16 *bytes of* $Hash(Intermediate_{pos})$

PoS Hash (Raw) - $Hash_{raw}$ = $Hash(magic, Nonce_{pos}, Height_{new})$

PoS Hash (Final) - $Hash_{pos} = \frac{Hash_{raw}}{value}$

$Hash_{pos}$ is the value used to compare to the network-wide PoS target to determine if a spend of that UTXO as proof-of-stake in a block will be accepted as a validated block under the network consensus rules. In addition, the random value of consecutive zeros in the most significant bits of the nonce itself, limited to ½ the value of the $posTarget$, is also considered as additional stake value of that block submission when used to determine the Chain Power value of that block. Because in almost all cases this results in one single block with the largest stake value among all entries, if there are multiple entries for a particular block height, all peers in the network with the same information can still objectively agree upon the single best chain, based on consensus rules.

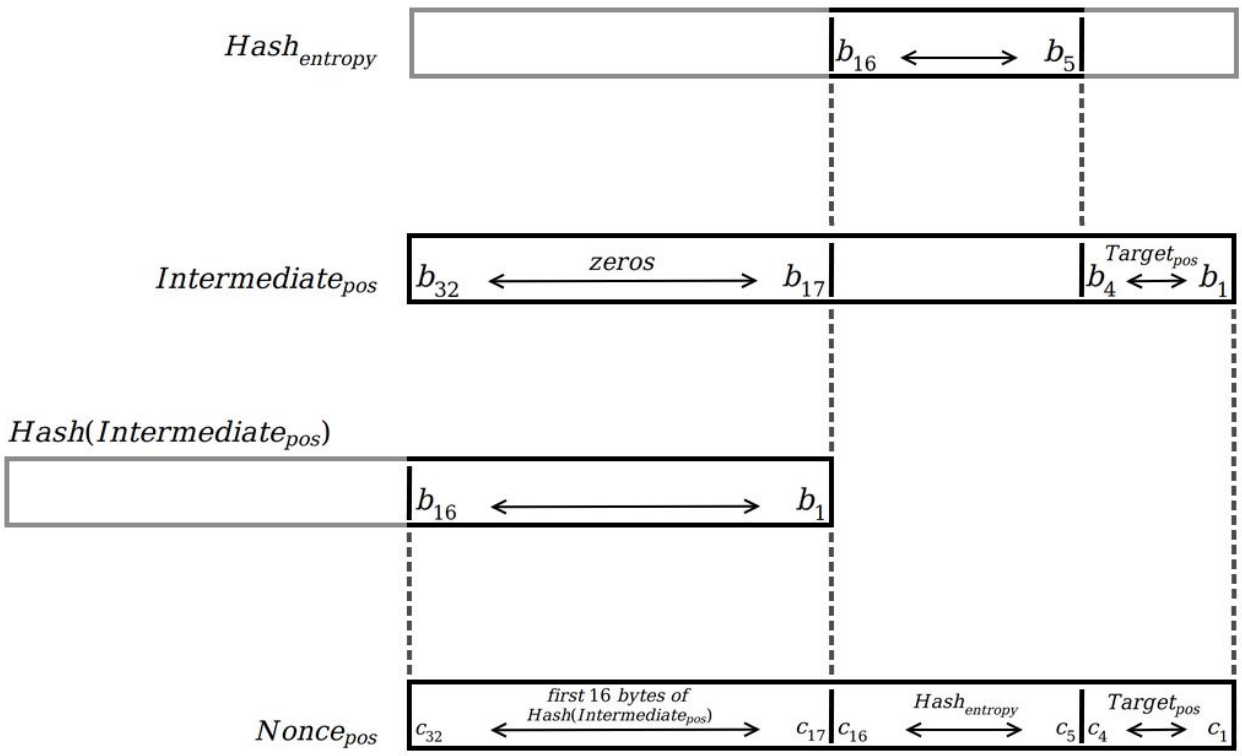

fig. 1

IV. Verus Proof of Power

Verus Coin implements Proof of Power (PoP) by combining VerusHash PoW and VerusPoS in a unique proof-of-work / proof-of-stake hybrid algorithm, in which both staking and mining compete simultaneously on the network for each and every block. The staking algorithm operates entirely in parallel, yet relative to, PoW progress which is used as the basis for the proof-of-stake block timer and difficulty adjustment.

This approach targets proof-of-stake validation for a statistical percentage of the validated blocks, without any specific foreknowledge of which block shall be validated by which method, using a difficulty algorithm entirely relative to PoW progress and ensuring independence from fluctuations in network hash rate. PoP includes a modification to the PoW “most-work” rule, used to select the best chain among forks that replaces a comparison with the amount of work represented by each chain with a composite measure, known as “Chain Power”, which is the total validation power of the network comprised of both cumulative chain work as well as cumulative chain stake.

To further reduce weak subjectivity in determining best chain, the PoS competition is augmented by an additional random, yet individually deterministic, stake value added to each block, which comes from the $Nonce_{pos}$ directly and ensures one objective winner across the entire network in all but a statistically

insignificant number of PoS contests won. The proof sketch under heading "Sketch II" of section "VI. Informal Proof Sketches" demonstrates how mounting a 51% hash attack against a PoP blockchain would effectively require much more than 51% of the network hash power as well as a significant amount of the coin supply.

Proof of Power also contributes to further decentralization by leveraging different, yet often intersecting, populations of interested network parties: those who mine; those who both mine and stake; and those who only stake. This enhanced decentralization further reduces risk of potentially centralized hardware advances while contributing to security by requiring a combination of attack vectors to be successful in order to mount a 51% attack on the network.

V. Stake Guard

In addition to the network-wide PoS contest and Chain Power selection, VerusPoP also includes an algorithm known as Stake Guard, which addresses the problem of "nothing-at-stake", a theoretical weakness of PoS over PoW that applies to proof of stake algorithms generally. VerusPoP uses the Stake Guard crypto-condition, a form of smart transaction that enables conditional transaction validation using an implementation of the Crypto-Condition IETF draft [3], to penalize double-staking of the same UTXO on two fork versions of a chain.

The Stake Guard algorithm enabling this penalty is a decentralized activity, where mining and staking participants on the network can look for double-staking protocol violations and use the signed stake transaction for the coins in question on one of the chain forks as the authorization, or proof of violation, which allows them to spend the double staker's reward on the main chain to their own address. Stake Guard is a component of VerusPoP, which is independent of and layered over the base protocol. As such, while it is introduced as part of VerusPoP here, its proof is beyond the scope of this sketch.

VI. Informal Proof Sketches

The following informal proof sketches detail the effectiveness of VerusPoP, both at solving a number of established practical and theoretical issues with modern day PoW and PoS systems, and at avoiding potential problems which may arise by incorrectly combining PoS and PoW systems.

Sketch I: Independence of VerusPoS from Network Hashrate

Part I: Establishing Independence

To truly benefit from the security aspects of both its PoW and VerusPoS components and to increase, rather than decrease, the security of the system as a whole by combining them, VerusPoP needs to ensure that changes to real network hashrate do not affect the statistical probability of PoW vs. PoS participation

in validating blocks. If the network is set to be 50% PoW and 50% VerusPoS, it must be able to maintain that ratio in the event a malicious actor were able to take control of the vast majority of network hashrate.

To prove the independence of the relative percentage of PoS vs. PoW in the presence of greatly fluctuating hashrate, we will first describe what inputs are used to determine the proof-of-stake difficulty, which determines whether or not a PoS block will win by statistically calculating at least one PoS hash for one UTXO across the entire network staking supply and achieve a result that is less than or equal to the PoS difficulty target.

We begin with the assumptions that 1) VerusPoS validations do not use any significant amount of computing power and will therefore always statistically occur more quickly than PoW blocks, even in the presence of significantly increasing hash power...and 2) there is at least one (1) VerusPoS staker and a total and constant staking supply at each block of S...and 3) the difficulty target $Target_{pos}$, which is a 32 bit compact form of an unsigned 256 bit value, will be a function of its max value, denoted by MAX_{C256}, and S, such that:

$$\frac{MAX_{C256}}{(2*S)} = Target_{pos}$$

We can conclude that if $Target_{pos}$ is kept at the this ratio during operation, statistically, VerusPoS should qualify to validate a block 50% of the time, independent of the network hashrate. Furthermore, if we were to modify the function to be as follows:

$$\frac{MAX_{C256}}{(n*S)} = Target_{pos}$$

We can see that VerusPoS would continue to statistically validate the number of blocks indicated by the mathematical ratio $(\frac{100}{n})\%$, independent of the network hashrate. It then follows that regardless of whether there is an instantaneous doubling, tripling, or more of network hashrate, as long as eligible staking calculations can be performed, compared to the target, and submitted to the network before a PoW block is solved, the percentage of blocks will depend on the staking difficulty, independent of the network hashrate for any constant staking difficulty. We have therefore established that given ideal staking performance, with respect to any specific, constant staking difficulty, the percentage of blocks that may be validated by VerusPoS is independent of hashrate.

Part II: Accounting for Real World Fluctuations

In the real world, the staking supply fluctuates, as does the staking difficulty. This difficulty is controlled by the VerusPoS difficulty algorithm in a manner similar to a typical PoW difficulty algorithm, through a form of moving averages, but using specific input variables which isolate VerusPoS difficulty calculations from actual block-time or other inputs that may be influenced by hashrate. If we do not use inputs that can be influenced by changes in network hashrate while recalculating the difficulty target in order to raise or lower the percentage of blocks validated by PoS, we can safely assume that PoS percentage and difficulty adjustment are independent of hashrate fluctuations.

With this established, we denote a function F for the difficulty target $Target_{pos}$ of any block B_n, given block B_{n-1}, that takes in two input values. One is the difficulty target $LastTarget_{pos}$ of B_{n-1}, and the other is a sequence Q_n, for the last n blocks, where each value in Q_n is either a 1, to represent a block validated by VerusPoS, or a 0, to represent a block validated by PoW. The function D can be denoted as:

$$F(Q_n,\ LastTarget_{pos}) = Target_{pos}$$

Since we have already established that all of the inputs are independent of the network hashrate at any given hashrate, we can conclude that for difficulty adjustment algorithm F, the PoS difficulty target is independent of network hashrate. Therefore, we can further conclude that for a VerusPoS blockchain with a PoS difficulty target determined solely by a function, $F(Q_n,\ LastTarget_{pos})$, the percentage of PoS blocks validated by the network is independent of network hash rate.

Sketch II: The Futility of 51% PoW Hash Attacks

Part I: The 51% Hash Attack

Using the above conclusion we can further define a VerusPoS Difficulty Adjustment Algorithm (POSDAA) as a function of the form $F(Q_n,\ LastTarget_{pos})$. We can then reason that whatever average percentage of blocks a VerusPoS DAA maintains as the PoS average, that percentage is independent of network hashrate. If we assume a POSDAA statistically maintains an average of 50% of blocks validated through VerusPoS and the other 50% through PoW, we can further conclude that in order for a malicious actor to gain majority validating control of a VerusPoP chain they would need to accumulate more than 50% of the blockchain's total validating power, otherwise referred to as Chain Power, including both PoW and VerusPoS validation. Following, we will prove that this is statistically impossible using solely PoW hash power or staking supply alone as long as the POSDAA keeps the average PoS validation percentage at 50% or above.

If an attacker possessed more than 50% of the network PoW hashpower there are two scenarios in which they could attempt an attack on the network. First is the scenario in which they are conducting an attack on the main chain, in public, while there is a staking supply controlled by other peers. The second is the scenario in which they conduct an attack in private, on their own version of the chain, where there is no staking supply in an attempt to then propagate their chain and overtake the network.

In the first scenario, and the assumption of 50% average PoS validation percentage, the statistical probability that the next block will be validated through PoW is 50%, due to the presence of a staking supply, therefore the probability of a miner mining more than a given amount of PoW blocks k in a given amount of blocks n can be modeled by the binomial distribution function, in which h represents their percentage of the PoW network hashpower:

$$P(k \geq x) = \sum_{k=x}^{n} \left(\frac{n!}{k!(n-k)!} * \left(\frac{h}{200}\right)^{k} * \left(\frac{h}{200}\right)^{n-k}\right)$$

We continue with this function to determine that the expected value, $E(k)$, of blocks mined can be defined as:

$$E(k) = n * \frac{h}{200}$$

Using the previous function we can show that in a theoretical sample of 100 blocks ($n = 100$) the probability of a miner with 51% of the network hashpower ($h = 51$) winning more than 50% of the block contests ($x = 50$), equates to about $4.27 * 10^{-6}\%$ and is, therefore, statistically improbable and nearly impossible. Furthermore, we can show that the expected value of blocks won, given these parameters, would be ~25.

Part II: Determining the Maximum Impact of a Hash Attack

It is possible, however, that a determined malicious actor on the network could get more than 51% of the total PoW hashpower. To prove that this will not allow said actor to have any statistically possible opportunity to launch a successful attack we can calculate the probability of them winning more than 50% of the blocks on the network, given they control 100% of the PoW hashrate ($h = 100$) in that same sample to be about 46%. We can further show that, given these parameters, the expected value of blocks won would be 50. Due to the probability of a single actor having 100% of the network PoW hash power being zero unless they are the only miner on the network, we can deduce that the maximum probability any one actor has to take control of more than 51% of the network is less than 46%, given other miners are present. Moreover, any miner with 100% of the PoW hashrate is expected to be limited to launching, at most, a 50% attack.

In the second scenario, in which a miner mines in private without any stakers, creating a 100% PoW chain which they will then try and propagate to overtake the main chain, we can prove a 51% attack to be statistically improbable and nearly impossible. We prove this by first defining how Chain Power evaluates if a chain is more or less powerful than the chain against which it is being compared, by taking the power P for each chain with length n, C_n to be a function of four values. 1) The cumulative work of C_n, $work_{C_n}$ 2) the cumulative stake of C_n, $stake_{C_n}$, 3) the largest work value out of both chains being compared, $work_{MAX}$ and 4) the largest stake value out of both chains being compared, $stake_{MAX}$.

The $work_{C_n}$ value for C_n represents the sum of all work difficulty values from W_0 to W_n. Any given block's difficulty value is derived from, and inversely proportional to that block's $Target_{pow}$. Therefore, $work_{C_n}$ can be described as:

$$work_{C_n} = \sum_{i=0}^{n} W_i$$

The $stake_{C_n}$ value for C_n represents the sum of all stake difficulty values from S_0 to S_n where each block's stake value is supplemented a deterministic, random, imaginary amount of additional stake. This imaginary stake difficulty is calculated by taking the PoS Nonce value for the block and treating it as if it were a stake difficulty target, while limiting its contribution to ½ of the real stake difficulty value. Any given block's difficulty value is derived from, and inversely proportional to that block's $Target_{pos}$ in the same way that work difficulty values are derived from $Target_{pow}$. Therefore, $stake_{C_n}$ can be described as:

$$f(x) = \begin{cases} x, & if\, x < \frac{S_i}{2} \\ \frac{S_i}{2} & if\, x \geq \frac{S_i}{2} \end{cases}$$

$$stake_{C_n} = \sum_{i=0}^{n} (S_i + f(R_i)),$$

Therefore, the power P for chain n can be evaluated as follows. If either $stake_{MAX}$ or $work_{MAX}$ are equal to 0, the denominator used in their place for function for P_n will be 1. The value P for chain n can be described as:

$$f(x) = \begin{cases} x, & if\, x \neq 0 \\ 1 & if\, x = 0 \end{cases}$$

$$P_n = ((stake_n)/(g(stake_{MAX}))) + ((work_n)/(g(work_{MAX})))$$

Hence if an attackers chain a has 100% of all network PoW hashpower and no staking power, then $stake_a = 0$ and $work_a = work_{MAX}$. We can compare it to a chain b in which there is no PoW hashpower and all of the network staking power, then $stake_b = stake_{MAX}$ and $work_b = 0$. Therefore:

$$P_a = 0/(stake_{MAX})) + ((work_{MAX})/(work_{MAX})) = 1$$

And

$$P_b = ((stake_{MAX})/(stake_{MAX})) + (0/(work_{MAX})) = 1$$

$$\therefore P_a = P_b$$

Therefore, the maximum Chain Power an attacker can receive through PoW hashing alone on chain a is at most equal to the Chain Power on chain b, given there are any stakers at all. This is also only possible if the only miner on the network is the attacker, the moment one miner begins to add PoW hashpower to chain b, $P_a < P_b$.

VII. Conclusion

From the reasoning above we can conclude that VerusPoP does in fact solve the 51% hash attack problem seen in many blockchains today. Although taking over a chain that uses VerusPoP is not impossible, doing so with a 51% attack would require a combined value of over 50% of both the chain's hashpower and its coin supply, making the success of any attack subject to acquisition of substantial amounts of validating stake as well as losses from any required stake participation and purchases thereof.

VIII. References

[1] Nakamoto, Satoshi. “Bitcoin: A Peer-to-Peer Electronic Cash System.” Bitcoin.org, http://bitcoin.org/bitcoin.pdf

[2] Kölbl, Stefan, et al. “Haraka v2 – Efficient Short-Input Hashing for Post-Quantum Applications.” International Association for Cryptologic Research, 24 Dec. 2016.

[3] Thomas, S., et al. “Crypto-Conditions.” IETF Tools, 24 Oct. 2016, tools.ietf.org/html/draft-thomas-crypto-conditions-01