1 Introduction

PacketCrypt is a bandwidth hard proof of work (PoW) random oracle that requires I rounds of encryption, I rounds of memory lookups and a final step to produce a digest. A fair prover presents the verifier with the digest, a merkle tree commitment of the full input set, and the sparse input subset of inputs used to produce the digest. PacketCrypt derives its data dependent memory hardness from the N blocks of memory required to produce a proof. Though not all memory hard functions are bandwidth hard, PacketCrypt gains its bandwidth hardness from its lack of locality in its memory access pattern.

2 Packet Crypt Memory Hardness

For a fair prover, PacketCrypt can be computed under the following definition.

Definition 1.

$$PacketCrypt\ can\ be\ fairly\ computed\ in\ I\ +\ 1\ steps,\ where\ I\ =\ 4$$

Thus, PacketCrypt can be computed in exactly 5 steps for a fair prover. However, for an adversary who wishes to compute PacketCrypt with a cache m where $m = N - x$, and x is the number of blocks of memory discarded such that $x > 0$, the probability of a successful step in a given round i is given by:

$$Pr(S) = (N - x / N)\ = p$$
$$where\ S = Successful\ Step$$

$$Pr(F) = 1 - p = q$$
$$where\ F\ =\ Failed\ Step$$

Let D be the number of trials until a successful step in the i^{th} round . Then the probability of achieving a successful step after $D = d$ trials is given by:

$$Pr(D = d)\ =\ Pr_{D}(FF...FS)\ =\ (1\ -\ p)(1\ -\ p)\ ...\ (p)\ =\ (q)(q)\ ...\ (p)\ =(q)^{d-1}p$$
$$where\ d \in\ \{1\ ...\ N\}$$

This series of probabilities is a geometric distribution. Thus, the expected number of attempts to achieve successful step in a given round i is:

$$E[D] = p + 2(1-p)p + 3(1-p)^2 p + \ldots + d(1-p)^{d-1} p + \ldots = \sum_{d=1}^{N} d(1-p)^{d-1} p = \frac{1}{p}$$

$$Hence,\ E[D] = \frac{N}{N-x}$$

Since PacketCrypt runs for I rounds and 1 additional step, the total number of computational steps are:

Definition 2.

$$Total\ computational\ steps = 1 + \sum_{i=1}^{I} \left(\frac{N}{N-x}\right)^i$$

or

$$1 + \sum_{i=1}^{4} \left(\frac{N}{N-x}\right)^i \ for\ I = 4$$

From **definition 1**, a fair prover can compute PacketCrypt in $I + 1$ steps, thus we compare this value to the number of steps the adversary needs to compute the function. In other words we wish to show:

$$1 + \sum_{i=1}^{4} \left(\frac{N}{N-x}\right)^i \geq I + 1$$

This reduces to:

$$\sum_{i=1}^{4} \left(\frac{N}{N-x}\right)^i \geq 4 \quad for\ I = 4$$

Since, by construction, $\sum_{i=1}^{I} \left(\frac{N}{N-x}\right)^i$ is a divergent geometric series for large I, it is easy to see that this inequality holds true for $N > 0,\ x \geq 0\ and\ N > x$

Though it is possible for an adversary to produce a proof with cache reduction x for $0 < x < N$, it is not possible to produce a proof with less computational steps than the fair prover.

For example, if an adversary wishes to discard half of their memory they will be attempting to compute a PacketCrypt proof with $m = x = \frac{N}{2}$ space. Applying **definition 2** we get:

$$1 + \sum_{i=1}^{4} \left(\frac{N}{N-\frac{N}{2}}\right)^i = 1 + 2^1 + 2^2 + 2^3 + 2^4 = 31\ steps$$

Since the fair prover can complete the proof in just 5 steps, we get $\frac{31}{5} = 6.2$ times more computational complexity to compute PacketCrypt with half of the memory.

Now, let $T =$ the time to produce a PacketCrypt proof, we then define the following.

Definition 3.

We define T as the time required to produce a PacketCrypt proof. Therefore we have:

$$T = (time / step)\,(total\ steps) = \Phi\left(1 + \sum_{i=1}^{4} \left(\frac{N}{N-x}\right)^{i}\right)$$

$$where\ \Phi = \frac{time}{step}$$

To analyze the memory hardness of PacketCrypt we use the classical approach and assess the space time complexity of the function. Space time complexity as defined by Percival [1] is a function which uses S space and T time operations such that:

1. $S = O(T^{1-\varepsilon})$
2. $S' = O(T'^{2-\varepsilon})\ for\ S' < S,\ T' > T,\ and\ \varepsilon > 0$

In other words, PacketCrypt can be computed in time $S \cdot T$ but not $S' \cdot T'$ where $S \cdot T \leq S' \cdot T'$.

Computing the space time complexity ($S \cdot T$) of PacketCrypt we get:

$$S \cdot T = (N)\Phi(I + 1)$$
$$where\ S = space = N$$
$$and\ T = time = \Phi(I + 1)$$

Thus for a fair prover, $I = 4$, the space time complexity is given by the following.

$$S \cdot T = (N)\Phi 5 = 5N\Phi$$

Alternately, for the adversary, the space time complexity ($S' \cdot T'$) is given as follows.

$$S' \cdot T' = (N - x)\Phi\left(1 + \sum_{i=1}^{4} \left(\frac{N}{N-x}\right)^{i}\right)$$
$$where\ S' = space = N' = N - x,\ and > 0,$$
$$and\ T' = time = \Phi\left(1 + \sum_{i=1}^{4} \left(\frac{N}{N-x}\right)^{i}\right)$$

Under strict constraints, we let $I = N$ rounds then space time complexity can be calculated as:

$$S \cdot T = (N)\Phi(N + 1)$$
$$thus\ S \cdot T \in O(N^2),$$
$$and\ S \cdot T \notin O(N'N)\ given\ N' = N - x$$

Which satisfies the requirements for PacketCrypt's memory hardness. However, The $I = N$ rounds condition is relaxed in practice to compute the function in adequate time to meet the requirements of a high-speed blockchain. Thus, we define a sufficient condition for memory hardness as follows.

Definition 4.

A function $f(\cdot)$ is sufficiently memory hard if a fair prover with space time complexity $= S \cdot T$ and a adversary with space time $= S' \cdot T'$ satisfies

$$\frac{S \cdot T}{S' \cdot T'} \leq 1$$

For PacketCrypt $S \cdot T = 5N\Phi$ and $S' \cdot T' = (N - x)\Phi\left(1 + \sum_{i=1}^{4} \left(\frac{N}{N-x}\right)^i\right)$

And it therefore follows that.

$$\frac{S \cdot T}{S' \cdot T'} = \frac{5N\Phi}{(N-x)\Phi\left(1 + \sum_{i=1}^{4} \left(\frac{N}{N-x}\right)^i\right)}$$

Thus **definition 4** is satisfied if and only if:

$$S \cdot T \leq S' \cdot T'$$
or
$$5N\Phi \leq (N - x)\Phi\left(1 + \sum_{i=1}^{4} \left(\frac{N}{N-x}\right)^i\right)$$

After algebraic simplification we get the following rearrangement of the inequality.

$$\frac{3N-x}{N-x} \leq \left(\frac{N}{N-x}\right)^2 + \left(\frac{N}{N-x}\right)^3 + \left(\frac{N}{N-x}\right)^4$$

Which is true for $N \geq 0,\ x \geq 0,\ N > x$ and thus PacketCrypt is proven to be sufficiently memory hard.

However, the memory hard function analysis is incomplete in its representation of the ASIC adversary's advantage. Specifically, memory hard function analysis, or time memory trade off (TMTO), only addresses the ASIC's space advantage, derived from its capacity for using less space to compute a hash. This analytic approach compels the design of a function such that it

will force an ASIC to spend maximal space to compute the digest. But, this fails to account for the ASIC's significant energy advantage. Thus, it is necessary to evaluate PacketCrypt's bandwidth hardness to properly compute the total advantage an adversary may have in practice.

3 PacketCrypt Bandwidth Hardness

It has been observed that many memory hard functions are also bandwidth hard [2], and PacketCrypt is consistent with this observation. PacketCrypt derives its bandwidth hardness by requiring a lot of off-chip memory accesses which occur in a non-local manner. It would be difficult for an ASIC adversary to derive an on-chip cache to filter out many off-chip memory accesses. Thus, the ASIC adversary would have little advantage due to its large number of cache misses and additional amortized energy costs due to super-linear computation.

The following is a generalization of the adversary's energy advantage defined by Ren and Devadas [2]. The generalized form accounts for differences in the computational steps (rounds) required for the fair prover versus the adversary.

Definition 5.

$$A_{ec} = \frac{jC_B B + kC_R R}{j'C'_B B' + k'C'_R R'}$$

C_B = *fair prover's memory access cost per byte per round*
B = *fair prover's # of bytes transferred per round*
C_R = *fair prover's computational cost per byte per round*
R = *fair prover's # of bytes computed per round*
C'_B = *adversary's memory access cost per byte per round*
B' = *adversary's # of bytes transferred per round*
C'_R = *adversaries computational cost per byte per round*
R' = *adversary's # of bytes computed per round*
j = *fair prover's number of rounds of memory accesses*
k = *fair prover's number of rounds of computes*
j' = *adversary's number of rounds of memory accesses*
k' = *adversary's number of rounds of computes*

For a function to be bandwidth hard it must necessarily satisfy the following condition.

Lemma 1.

$$A_{ec} = O(1)$$

Satisfying this condition implies that the adversary's advantage will be greatly limited.
The following table describes the energy costs for the fair prover (CPU) and the adversary (ASIC).

Table 1. Energy costs in nJ per Byte.

Operation	Memory access	SHA-256	AES-NI
Energy, CPU	.5	30	1.5
Energy, ASIC	.3	.0012	/

PacketCrypt uses chacha20/poly1305 for hashing which can do no worse than AES-NI, thus the energy costs of AES-NI acts as a lower bound for PacketCrypt's hash function.

Given that for a fair prover, ASIC or CPU, j = k = j'= k' = 4, **definition 5** reduces to the following lemma first defined by Ren and Devadas [1].

Lemma 2.

$$A_{ec} = \frac{C_B B + C_R R}{C'_B B' + C'_R R'}$$

For the fair prover, we have B = B' and R = R'. Furthermore since, each round of PacketCrypt requires a transfer of 1024 bytes from memory and 2048 bytes of processing, we have the following.

$$B = R/2$$

Thus, given **table 1**, for the fair prover, **lemma 2** can be reduced to:

$$A_{ec} = \frac{(.5)R/2 + (1.5)R}{(.3)R/2 + (.0012)R} = \frac{1.75R}{.4518R} = 3.87$$

Note that for C'_R we have no metrics for the AES-NI, thus the SHA-256 computational cost was used as a worst case substitution. Hence the ASIC adversary's energy cost advantage is about

3.9. Therefore, to achieve bandwidth hardness, it is necessary to ensure that no greater advantage can be achieved by the adversary's use of a cache of size m. Thus, we wish the memory requirement $N >> m$ such that it is great enough that most of N cannot fit in the cache.

Utilizing **definition 5** and **table 1,** we can compute the adversary's energy advantage as :

$$\overline{A_{ec}} = \frac{4(C_B B + C_R R)}{j' C'_B B' + (\sum_{i=1}^{4} (\frac{N}{N-x})^i) C'_R R'}$$

$$\overline{A_{ec}} = \frac{4(.5R/2 + 1.5R)}{f(x)(.3)R/2 + (\sum_{i=1}^{4} (\frac{N}{N-x})^i)(.0012)R}$$

$$Given\ j' = f(x)\ such\ that\ f(x) = \{4 :\ if\ x = 0;\ 1 :\ x > 0\}$$

$$And,\ B' = B = R/2,\ and\ j = k = I = 4,\ and\ k' = \sum_{i=1}^{4} \left(\frac{N}{N-x}\right)^i$$

To satisfy the upper bound requirement necessary to provide bandwidth hardness we wish to limit the impact of the adversary's cache m such that:

Definition 6.

If a function satisfies the inequality :

$\overline{A_{ec}} \leq A_{ec}$
It is bandwidth hard.

Applying **definition 6**, we seek the following:

$$\frac{4(.5R/2 + 1.5R)}{f(x)(.3)R/2 + (\sum_{i=1}^{4} (\frac{N}{N-x})^i)(.0012)R} \leq 3.87$$

It is obvious that this inequality is satisfied for x = 0. However, for x > 0 we get:

$$\frac{4(.5R/2 + 1.5R)}{(.3)R/2 + (\sum_{i=1}^{4} (\frac{N}{N-x})^i)(.0012)R} \leq 3.87$$

With algebraic rearrangement we arrive at:

$$1382.32 \leq (\frac{N}{N-x}) + (\frac{N}{N-x})^2 + (\frac{N}{N-x})^3 + (\frac{N}{N-x})^4$$

Since $m = N - x$, and for some z , $m = N/z$, we substitute N/z for $N - x$ and to derive the following polynomial inequality.

$$1382.32 \leq (z) + (z)^2 + (z)^3 + (z)^4$$

Solving for z using high order polynomial factorization we get one of the solutions as $z = 5.82 \approx 6$. Thus, to satisfy bandwidth hardness N must be maximized sufficiently large such that m can be no greater than $\frac{N}{6}$

$$m \leq \frac{N}{6}$$

Given that a miner will wish to optimize their chances of creating a PoW which meets the difficulty requirements they will seek to maximize N. This is equivalent to collecting the most announcements with which their memory can store in order to max(N). Since announcements amortize over time, the block miners will consistently seek to max(N) on each new block by collecting as many announcements which they can download to mine. With this dynamic at play if max(N) satisfies the criteria $m \leq \frac{N}{6}$ or $6m \leq N$ then PacketCrypt achieves bandwidth hardness.

References

[1] Colin Percival. Stronger Key Derivation Via Sequential Memory-hard Functions. https://www.tarsnap.com/scrypt/scrypt.pdf

[2] Ling Ren and Srinivas Devadas. Bandwidth Hard Functions for ASIC Resistance. https://eprint.iacr.org/2017/225.pdf