A More Elaborate Computation Model for Analyzing Balloon Hashing

Cui Hongrui, Sha Jinrui

November 25, 2017

1 An improved big-spread lemma

Room for improvement The *big spread lemma* plays an important role in the security proof of balloon hashing algorithm [1], however, the theorem require that the subset being considered, $V\prime$ have the same size of the pebbles available by the adversary. This is also reflected in Lemma 30, which bound the probability of the graph being too "dense". Recall in Lemma 30, for $\delta = 3$, every sandwich graph is an (m, m, n/16)-consecutively-avoiding graph, for $n_0 > 16$ and all $n_0 \leq m < n/6$ except with probability $P_{consec}(n, d, n_0) \leq 2 * 8 \cdot d \cdot n \cdot 2^{-n_0/2}$; And for $\delta = 7$, such stack of sandwich graph is (n/64, n/32)-everywhere-avoiding graph when $n > 2^{11}$, except with probability $P_{every} \leq 128 \cdot d \cdot 2^{-n/50}$. Note that in the later part of the lemma, the size of V' is defined as $m = \frac{n}{2^{\omega+1}}$, and thus m grows with n.

This limitation that the subset of V begin fixed by the number of pebbles of the adversary is somehow inconvenient, in that to make the probability of bad event small, one need the number of pebbles available grow, which is the contrary of natural instinct, as the more pebble the adversary acquires, the easier the attack (re-computation) should be. Although the result in [1] is absolutely correct, as we can view the growth of m as the result of the growth of n, an if the subset V' being considered too small, the variables sampled independently uniform at random from the last layer would have a bigger probability to being not so well-spread. Still, we want to separate the size of subset (m) and the number of pebbles ($|V'|$).

For the sake of clarity, and also because of the nature of an internal manuscript, I will replicate the content and the proof of big spread lemma.

Lemma 1 (Big Spread Lemma Variation)**.** *For all positive integer $\delta \geq 3$, $\omega \geq 2$, n_0, and n, and for all positive integers m such that $2^\omega < m < 2^{2-\omega+\frac{2\omega}{\delta e}}$, a list of δm elements sampled independently and uniformly at random from $1, \ldots, n$ is an $(n_0, n/2^\omega)$-well-spread set with probability at least $1-2^{(1-\omega)\frac{\delta}{2}m} \cdot (2^{\omega m} + 2^{\omega+1}2^{\omega n_0})$*

Proof. The Strategy to bound the bad event that such property does not exist is the same as that used in [1]. Let $R = (R_1, \ldots, R_{\delta m})$ be integers sampled independently and uniformly at random from $1, \ldots, n$, we want to prove that for all subset $S \subseteq R$ of size at most n_0, $spread_S(R) \geq n/2^\omega$. To do so, we first define a bad event B, and then show that bounding Pr[B] is enough to prove the lemma, then bound Pr[B] to complete the lemma.

The Bad Event B Write the integers in R in non-decreasing order as $X_1, \ldots, X_{\delta m}$, then define $X_0 = 0$, $X_{\delta m+1} = n$. Let bad event B be the event that there exists a set $S' \subseteq X_1, \ldots, X_{\delta m+1}$ of size at most (n_0+1), such that $\sum_{X_i \in S'} X_i - X_{i-1} \geq (1 - 2^{-\omega})n$.

Whenever there exists a set $S \subseteq R$ of size at most n_0 that cause $spread_S(R) < n/2^{\omega}$, then bad event B must occur. Assuming that such a bad set S exists, construct a set $S' = S \bigcup X_{\delta m+1}$ of size at most $n_0 + 1$, Then we compute

$$\sum_{X_i \in S'} (X_i - X_{i-1}) = n - X_{\delta m} + \sum_{X_i \in S} (X_i - X_{i-1}) = n - \sum_{X_i \notin S} (X_i - X_{i-1}) \quad (1)$$

The last inequality holds because $X_{\delta m} = \sum_{X_i \in R} (X_i - X_{i-1})$. Now that $Spread_S(R) < n/2^{\omega}$, so we have bad event B occurs. Thus, $\Pr[Spread_S(R)] \leq \Pr[\mathrm{B}]$, and therefore bounding a upper bound of bad event B is sufficient to prove the lemma.

Strategy to bound Pr[B] Let D be a random variable denoting the number of distinct integers in the list of random integers R. For any fixed integer $D^* \in \{1, \ldots, n\}$, we can write:

$$\begin{aligned} Pr[B] &= Pr[B|D < d^*] \cdot Pr[D < d^*] + Pr[B|D \geq d^*] \cdot Pr[D \geq d^*] \quad (2) \\ &\leq Pr[D < d^*] + Pr[B, D \geq d^*] \quad (3) \end{aligned}$$

In the following proof, we take $d^* = \delta m/2$.

Bounding $Pr[D < d^*]$ The probability that this event occurs is at most the probability when we throw δm balls into n bins, all the balls fall into a set of $\delta m/2$ bins.

$$Pr[D < d^*] \leq \binom{n}{d^*} (\frac{d^*}{n})^{\delta m} \leq (\frac{n \cdot e}{d^*})^{d^*} (\frac{d^*}{n})^{\delta m} = (\frac{d^*}{n})^{\delta m - d^*} e^{d^*} \quad (4)$$

By the hypothesis of the lemma, $m < 2^{(1-\omega+\frac{2\omega}{\delta})} \frac{2n}{\delta e}$. Therefore, we have

$$Pr[D < d^*] \leq (\frac{\delta m e}{2n})^{\delta m/2} \leq 2^{[(1-\omega)\frac{\delta}{2}+\omega]m} \quad (5)$$

Bounding $Pr[B, D \geq d^*]$ First I would like to re-state the lemma 8 in the original work [1], because this lemma is useful in the proof of big spread lemma.

Lemma 2. *Let $(R_1, \ldots, R_d)$ be random variables respecting integers sampled uniformly, but without replacement, from $\{1, \ldots, n\}$. Write the Rs in ascending order as $(Y_1, \ldots, Y_d)$, and let $Y_0 = 0$, and $Y_{d+1} = n$. Next define $L = (L_1, \ldots, L_{d+1})$, where $L_i = Y_i - Y_{i-1}$. Then for all functions $f : \mathbb{Z}^{d+1} \to \{0, 1\}$, and for all permutations π on $d+1$ elements,*

$$Pr[f(L_1, \ldots, Ld + 1) = 1] = Pr[f(L_{\pi(1)}, \ldots, L_{\pi(d+1)}) = 1]. \quad (6)$$

Particularly, this lemma shows that if we define a function that maps a list of segments (separated from $\{1, \ldots, n\}$ by the δm random integers) in to $\{0, 1\}$, such that if bad event B happens, then it is one. Using this lemma, we can define a permutation on those integers that if it appears in a bad set, then we

shift it with the first $n_0 + 1$ integers in the list, and thus we can only focus on the first few elements on the list without changing the result. As lemma 2 require the list to be mutually distinct, we have to transform the random list in to a distinct one first.

Step 1: Bad event B occurs whenever there is a subset $S' \subseteq X = (X_1, \ldots, X_{\delta m+1})$ of size at most (n_0+1), such that $\sum_{X_i \in S'} (X_i - X_{i-1}) \geq (1-2^{-\omega})n$. Whenever bad event B occurs, there is also a subset S'_{dist} of distinct integers that also satisfy the sum equation, and also has the size (n_0+1). (This is because if duplicate exists, we can always delete duplicates, without changing the sum, and then add new integers to make the sum even bigger.) thus we can use lemma 2 to simplify calculation.

Step 2: we consider the case there is explicitly d distinct integers in the list R, and then use union bound to deduce the final probability. Write the d distinct integers in R is ascending order as $Y = (Y_1, \ldots, Y_d)$. We want to bound the probability that some subset Y'_{dist} of size n_0 is bad. (Note that the only change from the original lemma is that I changed m into n_0.) Then define L_i as in lemma 2. Let the set of indices $I \subseteq \{1, \ldots, d+1\}$ of size n_0+1 be the set of index of bad subset. That is, $S'_{dist} = \{Y_i | i \in I\}$. Then define the permutation $\pi : \mathbb{Z}^{d+1} \to \mathbb{Z}^{d+1}$, such that if $i \in I$, then L_i appears in the first n_0+1 elements of $(L_{\pi(1)}, \ldots, L_{\pi(d+1)})$. Define function $f : \mathbb{Z}^{d+1} \to \{0,1\}$ such that if its first n_0+1 arguments sums to at least $(1-2^{-\omega})n$, returns 1 and it returns 0 otherwise. Therefore, we have:

$$Pr[\sum_{i \in I} L_i \geq (1-2^{-\omega})n | D = d] = Pr[f(L_{\pi(1)}, \ldots, L_{\pi(d+1)}) = 1 | D = d] \tag{7}$$

$$= Pr[f(L_1, \ldots, Ld+1) = 1 | D = d] \tag{8}$$

$$= Pr[(L_1 + \cdots + L_{n_0+1}) \geq (1-2^{-\omega})n | D = d] \tag{9}$$

Step 3: the event defined above can only happen when the other $d-(n_0+1)$ integers falls into the right most $2^{-\omega}n$ bins, and then we have:

$$Pr[(L_1 + \cdots + L_{n_0+1}) \geq (1-2^{-\omega})n | D = d] \leq (\frac{1}{2})^{\omega(d-n_0-1)} \tag{10}$$

Above is the probability that a single set is bad, we then sum it over all $\binom{d+1}{n_0+1}$ possible size n_0+1 subsets, to get:

$$Pr[B|D = d] \leq \binom{d+1}{n_0+1} (\frac{1}{2})^{\omega(d-n_0-1)} \tag{11}$$

$$\leq 2^{(1-\omega)d + \omega n_0 + \omega + 1} \tag{12}$$

Equipped with $Pr[B|D=d]$, we can then proceed to calculate $Pr[B, D \geq d^*]$. Note when $\omega > 1$, $Pr[B|D=d]$ is non-increasing in d. And therefore we have:

$$Pr[B, D \geq d^*] \leq 2^{(1-\omega)d^* + \omega n_0 + \omega + 1} \cdot \sum_{d=d^*}^{\delta m+1} Pr[D = d] \tag{13}$$

$$\leq 2^{1+\omega} \cdot 2^{(1-\omega)\frac{\delta m}{2} + \omega n_0} \tag{14}$$

Completing the proof By the calculation above, we have

$$Pr[B] \leq Pr[D < d^*] + Pr[B, D \geq d^*] \tag{15}$$

$$\leq 2^{(1-\omega)\frac{\delta m}{2}} \cdot (2^{\omega m} + 2^{1+(n_0+1)\omega}) \tag{16}$$

□

2 Everywhere avoiding property

An analysis of original strategy In the original work [1], the author proves a *consecutive avoiding property* over all possible size of m, and then for m of fixed size $(n/64)$, proves a tighter *everywhere avoiding property*. The reason of combining these two property is that the first one allows the number pebbles to be arbitrary within a given range. In particular, one can consider m to have the same number of the pebbles the adversary owns. In order to prove a tighter bound on $S \cdot T$ (like in part b of Lemma 31), one needs to first apply consecutive avoiding property and then use the everywhere avoiding property. What we want to do is to acquire the probability that given at most n_0 pebbles, any m pebbles on a standard sandwich graph is an $(n_0, m, 2/2^{\omega})$ avoiding set.

Lemma 3 (Everywhere Avoiding)**.** *Let $G = (U \bigcup V, E)$, be a δ-random sandwich graph on 2n vertices. Then for all positive integer m that satisfy the assumption of lemma 1, and n_0 such that n_0 is no bigger than the upper bound of m, we have that given n_0 pebbles at most, every subset $V' \subseteq V$ of size m, is a $(n_0, n/2^{\omega})$ avoiding set with probability at least $1 - (\frac{n}{m})^m \cdot 2^{(1-\omega)\frac{\delta m}{2}} \cdot (2^{\omega m} + 2^{1+(n_0+1)\omega})$*

Proof. Fix the size of n_0 and m, all we need to do is to apply union bound to all possible choice of V', which has $\binom{n}{m}$ in total. Apply this inequality to simply the bound:

$$\binom{n}{m} = \frac{n \times (n-1) \times \cdots \times (n-m+1)}{m \times (m-1) \times \ldots 1} \leq \frac{n^m}{m^m} = fracnm^m \tag{17}$$

And thus by lemma 1, we have the probability that a single bad event occurs is at most $2^{(1-\omega)\frac{\delta}{2}m} \cdot (2^{\omega m} + 2^{\omega+1}2^{\omega n_0})$, and thus the probability that for all choice of V', the bad event occurs is at most $(\frac{n}{m})^m \cdot 2^{(1-\omega)\frac{\delta m}{2}} \cdot (2^{\omega m} + 2^{1+(n_0+1)\omega})$. And thus the lemma is proved. □

3 A bound on the pebbling moves of random sandwich graph

Bounding pebbling moves And now for the sake of our own objective, we would like to calculate the bound of re-computation of a subset of vertices on any layer. To do this, we need to first compute the moves needed to pebble a subset of vertices. First define a variable to denote the pebbling moves needed to pebble a subset of m vertices on the condition that except on those m vertices, there can be at most n_0 pebbles on any vertices of the graph.

Definition 1. *Let G be a stack of random sandwich graphs. Then define T_r be the pebbling moves needed to pebble a subset V' of vertices on layer r, which has not received any pebble on the beginning of the moves, and at the end of the moves, the topologically last one receives a pebble. The condition is that there can only be at most n_0 pebbles on the graph.*

Now that we have defined the amount to calculate, we can use induction reasoning to calculate T_r.

Lemma 4. *Given the average case that $T_0 = \frac{m+n}{2}$, we have:*

$$T_r \leq -\frac{m}{\frac{n}{m2^\omega} - 1} + (\frac{n}{m2^\omega})^r(\frac{m+n}{2} + \frac{m}{\frac{n}{m2^\omega} - 1}) \tag{18}$$

where $0 \leq r \leq d$, d is the layer of the stack of random sandwich graph.

Proof. By the definition of T_r, we have that $T_r \leq m + \frac{n}{m2^\omega}T_{r-1}$. And thus we have:

$$T_r \leq m + \frac{n}{m2^\omega}T_{r-1} \tag{19}$$

$$T_r + \frac{m}{\frac{n}{m2^\omega} - 1} \leq \frac{n}{m2^\omega}(T_{r-1} + \frac{m}{\frac{n}{m2^\omega} - 1}) \tag{20}$$

$$\tag{21}$$

If we define $Y_r = T_r + \frac{m}{\frac{n}{m2^\omega}-1}$, and change the inequality into equality, we can have

$$Y_r = (\frac{n}{m2^\omega})^r \cdot Y_0 \tag{22}$$

And thus $T_r \leq -\frac{m}{\frac{n}{m2^\omega}-1} + (\frac{n}{m2^\omega})^r(\frac{m+n}{2} + \frac{m}{\frac{n}{m2^\omega}-1})$. □

Then we define the lower bound of re-computation needed in the process discussed above.

Definition 2. *Let Recompute(r) be the pebbling moves needed in the process to pebble m vertices on layer r. More specifically, the pebbling moves needed to pebble the direct predecessors of those m vertices.*

By the definition of *Recompute*(r) and lemma 4, one can see that *Recompute*$(r) \geq -m + T_r$, and it grows exponentially with r.

4 A comparison on re-computation and DRAM fetching

Different strategy The pebbling diagram intends to model the computation on ASIC circuits, that is, the computation of underlying cryptographic hashing function, like SHA-256, is relatively very fast[1], and those logic takes the major size of the circuit. The memory section, on the other hand, is relatively small. And thus in the model, we can model the adversary to have small amount of pebbles, and try to bound the pebbling moves of the adversary.

[1]Some pipelining circuits can reach the frequency as fast as 1THz

Layer	Ratio
1	0.01
2	0.04
3	0.15
4	0.62
5	2.46

Table 1: The ratio of re-computation over DRAM fetching

However, this model cannot extend to honest users, whose considerable amount of time is used in fetching previously computed blocks from DRAM, and this operation takes constant time. Intuitively, if the adversary, or the manufacturer of ASIC can combine those two advantages, could a more devastating attack be constructed? Especially when DRAM circuits are getting faster and relatively cheaper. On the other hand, when such case occurs, then total computational time depends not only on the speed of hashing, but also on the DRAM latency. In this case, will the increase in hashing power (non-memory hard hashing power) yields a less attractive increase in the throughput, and therefore render the solution of ASIC as a cracking machine useless?

Our preliminary work shows that given contemporary parameter settings, when the layer of the graph gets greater than 4, the DRAM strategy will definitely start to have an advantage.

5 Comparison on common parameter settings

Parameter setting In the following comparison section, we take the block size, which the output length of the underlying cryptographic hashing function to be 256 bit, and the block in one layer of the graph to be $2^2 2^2$ [2]. We take $\omega = 3$, and the ratio of the latency of fetching a DRAM block over the ratio of the time needed to output a block from Random Oracle, $c = 1000$.

Comparison As shown in table 1, given our parameter setting, the ratio of re-computation becomes bigger than 1 when the graph get deeper than 4 layers. However, this is a loose bound. Firstly, the DRAM latency is computed as δmc, where $c = 1000$, this does not take into account the case when the δm predecessors 'collide' with each other, that is, when one fetch benefit another pebbling process. Secondly, the re-computation was take the upper bound. Thirdly, an optimal strategy may combine these two means, using fetching for some vertices, and re-computation for others. Nevertheless, this rather loose bound shows that if the layer of the graph gets deeper, pure re-computation will never be a useful solution, no matter what.

References

[1] Henry Corrigan-Gibbs, Dan Boneh, and Stuart E Schechter. Balloon hashing: Provably space-hard hash functions with data-independent access pat-

[2] The authors of Balloon Hashing recommended the space parameter to be 256MB for authentication servers.

terns. *IACR Cryptology ePrint Archive*, 2016:27, 2016.