Fast Hashing with Strong Concentration Bounds[*]

Anders Aamand[†]   Jakob B. T. Knudsen[†]   Mathias B. T. Knudsen[‡]

Peter M. R. Rasmussen[†]   Mikkel Thorup[†]

August 11, 2020

Abstract

Previous work on tabulation hashing by Pătraşcu and Thorup from STOC'11 on simple tabulation and from SODA'13 on twisted tabulation offered Chernoff-style concentration bounds on hash based sums, e.g., the number of balls/keys hashing to a given bin, but under some quite severe restrictions on the expected values of these sums. The basic idea in tabulation hashing is to view a key as consisting of $c = O(1)$ characters, e.g., a 64-bit key as $c = 8$ characters of 8-bits. The character domain Σ should be small enough that character tables of size $|\Sigma|$ fit in fast cache. The schemes then use $O(1)$ tables of this size, so the space of tabulation hashing is $O(|\Sigma|)$. However, the concentration bounds by Pătraşcu and Thorup only apply if the expected sums are $\ll |\Sigma|$.

To see the problem, consider the very simple case where we use tabulation hashing to throw n balls into m bins and want to analyse the number of balls in a given bin. With their concentration bounds, we are fine if $n = m$, for then the expected value is 1. However, if $m = 2$, as when tossing n unbiased coins, the expected value $n/2$ is $\gg |\Sigma|$ for large data sets, e.g., data sets that do not fit in fast cache.

To handle expectations that go beyond the limits of our small space, we need a much more advanced analysis of simple tabulation, plus a new tabulation technique that we call *tabulation-permutation* hashing which is at most twice as slow as simple tabulation. No other hashing scheme of comparable speed offers similar Chernoff-style concentration bounds.

---

i

Contents

1 Introduction

Chernoff's concentration bounds [12] date back to the 1950s but bounds of this types go even further back to Bernstein in the 1920s [7]. Originating from the area of statistics they are now one of the most basic tools of randomized algorithms [36]. A canonical form considers the sum $X = \sum_{i=1}^{n} X_i$ of independent random variables $X_1, \ldots, X_n \in [0, 1]$. Writing $\mu = \mathbb{E}[X]$ it holds for every $\varepsilon \geq 0$ that

$$\Pr[X \geq (1 + \varepsilon)\mu] \leq \exp(-\mu\, \mathcal{C}(\varepsilon)) \qquad \left[\leq \exp(-\varepsilon^2 \mu/3) \text{ for } \varepsilon \leq 1 \right], \qquad (1)$$

$$\Pr[X \leq (1 - \varepsilon)\mu] \leq \exp(-\mu\, \mathcal{C}(-\varepsilon)) \qquad \left[\leq \exp(-\varepsilon^2 \mu/2) \text{ for } \varepsilon \leq 1 \right]. \qquad (2)$$

Here $\mathcal{C} : (-1, \infty) \to [0, \infty)$ is given by $\mathcal{C}(x) = (x + 1)\ln(x + 1) - x$, so $\exp(-\mathcal{C}(x)) = \frac{e^x}{(1+x)^{(1+x)}}$. Textbook proofs of (1) and (2) can be found in [36, §4][1]. Writing $\sigma^2 = \text{Var}[X]$, a more general bound is

$$\Pr[|X - \mu| \geq t] \leq 2\exp(-\sigma^2 \mathcal{C}(t/\sigma^2)) \qquad \left[\leq 2\exp(-(t/\sigma)^2/3) \text{ for } t \leq \sigma^2 \right]. \qquad (3)$$

Since $\sigma^2 \leq \mu$ and $\mathcal{C}(-\varepsilon) \leq 1.5\, \mathcal{C}(\varepsilon)$ for $\varepsilon \leq 1$, (3) is at least as good as (1) and (2), up to constant factors, and often better. In this work, we state our results in relation to (3), known as Bennett's inequality [6].

Hashing is another fundamental tool of randomized algorithms dating back to the 1950s [23]. A random hash function, $h : U \to R$, assigns a hash value, $h(x) \in R$, to every key $x \in U$. Here both U and R are typically bounded integer ranges. The original application was hash tables with chaining where x is placed in bin $h(x)$, but today, hash functions are ubiquitous in randomized algorithms. For instance, they play a fundamental role in streaming and distributed settings where a system uses a hash function to coordinate the random choices for a given key. In most applications, we require concentration bounds for one of the following cases of increasing generality.

1. Let $S \subseteq U$ be a set of balls and assign to each ball, $x \in S$, a weight, $w_x \in [0, 1]$. We wish to distribute the balls of S into a set of bins $R = [m] = \{0, 1, \ldots, m-1\}$. For a bin, $y \in [m]$, $X = \sum_{x \in S} w_x \cdot [h(x) = y]$ is then the total weight of the balls landing in bin y.

2. We may instead be interested in the total weight of the balls with hash values in the interval $[y_1, y_2)$ for some $y_1, y_2 \in [m]$, that is, $X = \sum_{x \in S} w_x \cdot [y_1 \leq h(x) < y_2]$.

3. More generally, we may consider a fixed *value function* $v : U \times R \to [0, 1]$. For each key $x \in U$, we define the random variable $X_x = v(x, h(x))$, where the randomness of X_x stems from that of $h(x)$. We write $X = \sum_{x \in U} v(x, h(x))$ for the sum of these values.

To exemplify applications, the first case is common when trying to allocate resources; the second case arises in streaming algorithms; and the third case handles the computation of a complicated statistic, X, on incoming data. In each case, we wish the variable X to be concentrated around its mean, $\mu = \mathbb{E}[X]$, according to the Chernoff-style bound of (3). If we had fully random hashing, this would indeed be the case. However, storing a fully random hash function is infeasible. The goal of this paper is to obtain such concentration with a practical constant-time hash function. More specifically, we shall construct hash functions that satisfy the following definition when X is a random variable as in one of the three cases above.

Definition 1 (Strong Concentration). Let $h : [u] \to [m]$ be a hash function, $S \subseteq [u]$ be a set of hash keys of size $n = |S|$, and $X = X(h, S)$ be a random variable, which is completely determined by h and S. Denote by $\mu = \mathbb{E}[X]$ and $\sigma^2 = \text{Var}[X]$ the expectation and variance of X. We say that X is *strongly concentrated with added error probability* $f(u, n, m)$ if for every $t > 0$,

$$\Pr\left[|X - \mu| \geq t\right] \leq O\left(\exp\left(-\Omega(\sigma^2 \mathcal{C}(t/\sigma^2))\right)\right) + f(u, n, m). \qquad (4)$$

Throughout the paper we shall prove properties of random variables that are determined by some hash function. In many cases, we would like these properties to continue to hold while conditioning the hash function on its value on some hash key.

---

[1] The bounds in [36, §4] are stated as working only for $X_i \in \{0, 1\}$, but the proofs can easily handle any $X_i \in [0, 1]$.

Definition 2 (Query Invariant)**.** Let $h\colon [u] \to [m]$ be a hash function, let $X = X(h)$ be a random variable determined by the outcome of h, and suppose that some property T is true of X. We say that the property is *query invariant* if whenever we choose $x \in [u]$ and $y \in [m]$ and consider the hash function $h' = (h|h(x) = y)$, i.e., h conditioned on $h(x) = y$, property T is true of $X' = X(h')$.

Remark. For example, consider the case (1) from above. We are interested in the random variable $X = \sum_{x \in S} w_x \cdot [h(x) = y]$. Suppose that for every choice of weights, $(w_x)_{x \in S}$, X is strongly concentrated and that this concentration is query invariant. Let $x_0 \in [u]$ be a distinguished query key. Then since for every $y_0 \in [m]$, the hash function $h' = (h|h(x_0) = y_0)$ satisfies that $X' = \sum_{x \in S} w_x \cdot [h'(x) = y_0]$ is strongly concentrated, it follows that $X'' = \sum_{x \in S} w_x \cdot [h(x) = h(x_0)]$ is strongly concentrated. Thus, h allows us to get Chernoff-style concentration on the weight of the balls landing in the same bin as x_0.

This may be generalized such that in the third case from above, the weight function may be chosen as a function of $h(x_0)$. Thus, the property of being query invariant is very powerful. It is worth noting that the constants of the asymptotics may change when conditioning on a query. Furthermore, the expected value and variance of X' may differ from that of X, but this is included in the definition.

One way to achieve Chernoff-style bounds in all of the above cases is through the classic k-independent hashing framework of Wegman and Carter [48]. The random hash function $h : U \to R$ is k-independent if for any k distinct keys $x_1, \dots, x_k \in U$, $(h(x_1), \dots, h(x_k))$ is uniformly distributed in R^k. Schmidt and Siegel [43] have shown that with k-independence, the above Chernoff bounds hold with an added error probability decreasing exponentially in k. Unfortunately, a lower bound by Siegel [44] implies that evaluating a k-independent hash function takes $\Omega(k)$ time unless we use a lot of space (to be detailed later).

Pătrașcu and Thorup have shown that Chernoff-style bounds can be achieved in constant time with tabulation based hashing methods; namely simple tabulation [38] for the first case described above and twisted tabulation [41] for all cases. However, their results suffer from some severe restrictions on the expected value, μ, of the sum. More precisely, the speed of these methods relies on using space small enough to fit in fast cache, and the Chernoff-style bounds [38, 41] all require that μ is much smaller than the space used. For larger values of μ, Pătrașcu and Thorup [38, 41] offered some weaker bounds with a deviation that was off by several logarithmic factors. It can be shown that some of these limitations are inherent to simple and twisted tabulation. For instance, they cannot even reliably distribute balls into $m = 2$ bins, as described in the first case above, if the expected number of balls in each bin exceeds the space used.

In this paper, we construct and analyse a new family of fast hash functions *tabulation-permutation* hashing that has Chernoff-style concentration bounds like (3) without any restrictions on μ. This generality is important if building a general online system with no knowledge of future input. Later, we shall give concrete examples from streaming where μ is in fact large. Our bounds hold for all of the cases described above and all possible inputs. Furthermore, tabulation-permutation hashing is an order of magnitude faster than any other known hash function with similar concentration bounds, and almost as fast as simple and twisted tabulation. We demonstrate this both theoretically and experimentally. Stepping back, our main theoretical contribution lies in the field of analysis of algorithms, and is in the spirit of Knuth's analysis of linear probing [29], which shows strong theoretical guarantees for a very practical algorithm. We show that tabulation-permutation hashing has strong theoretical Chernoff-style concentration bounds. Moreover, on the practical side, we perform experiments, summarized in Table 1, demonstrating that it is comparable in speed to some of the fastest hash functions in use, none of which provide similar concentration bounds.

When talking about hashing in constant time, the actual size of the constant is of crucial importance. First, hash functions typically execute the same instructions on all keys, in which case we always incur the worst-case running time. Second, hashing is often an inner-loop bottle-neck of data processing. Third, hash functions are often applied in time-critical settings. Thus, even speedups by a multiplicative constant are very impactful. As an example from the Internet, suppose we want to process packets passing through a high-end Internet router. Each application only gets very limited time to look at the packet before it is forwarded. If it is not done in time, the information is lost. Since processors and routers use some of the same technology, we never expect to have more than a few instructions available. Slowing down the Internet is typically not an option. The papers of Krishnamurthy et al. [30] and Thorup and Zhang [47] explain in more detail how high speed hashing is necessary for their Internet traffic analysis. Incidentally, our hash

2

function is a bit faster than the ones from [30, 47], which do not provide Chernoff-style concentration bounds.

Concrete examples of the utility of our new hash-family may be found in [1]. In [1] it is shown that some classic streaming algorithms enjoy very substantial speed-ups when implemented using tabulation-permutation hashing; namely the original similarity estimation of Broder [8] and the estimation of distinct elements of Bar-Yossef et al. [5]. The strong concentration bounds makes the use of independent repetitions unnecessary, allowing the implementations of the algorithms to be both simpler and faster. We stress that in high-volume streaming algorithms, speed is of critical importance.

Tabulation-permutation hashing builds on top of simple tabulation hashing, and to analyse it, we require a new and better understanding of the behaviour and inherent limitations of simple tabulation, which we proceed to describe. Afterwards we break these limitations by introducing our new powerful tabulation-permutation hashing scheme.

1.1 Simple Tabulation Hashing

Simple tabulation hashing dates back to Zobrist [49]. In simple tabulation hashing, we consider the key domain U to be of the form $U = \Sigma^c$ for some character alphabet Σ and $c = O(1)$, such that each key consists of c characters of Σ. Let $m = 2^\ell$ be given and identify $[m] = \{0, 1, \ldots, m-1\}$ with $[2]^\ell$. A simple tabulation hash function, $h\colon \Sigma^c \to [m]$, is then defined as follows. For each $j \in \{1, \ldots, c\}$ store a fully random character table $h_j\colon \Sigma \to [m]$ mapping characters of the alphabet Σ to ℓ-bit hash values. To evaluate h on a key $x = (x_1, \ldots, x_c) \in \Sigma^c$, we compute $h(x) = h_1(x_1) \oplus \cdots \oplus h_c(x_c)$, where $\oplus$ denotes bitwise XOR – an extremely fast operation. With character tables in cache, this scheme is the fastest known 3-independent hashing scheme [38]. We will denote by $u = |U|$ the size of the key domain, identify $U = \Sigma^c$ with $[u]$, and always assume the size of the alphabet, $|\Sigma|$, to be a power of two. For instance, we could consider 32-bit keys consisting of four 8-bit characters. For a given computer, the best choice of c in terms of speed is easily determined experimentally once and for all, and is independent of the problems considered.

Let $S \subseteq U$ and consider hashing $n = |S|$ weighted balls or keys into $m = 2^\ell$ bins using a simple tabulation function, $h\colon [u] \to [m]$, in line with the first case mentioned above. We shall prove the theorem below.

Theorem 1. *Let $h\colon [u] \to [m]$ be a simple tabulation hash function with $[u] = \Sigma^c$, $c = O(1)$. Let $S \subseteq [u]$ be given of size $n = |S|$ and assign to each key/ball $x \in S$ a weight $w_x \in [0, 1]$. Let $y \in [m]$, and define $X = \sum_{x \in S} w_x \cdot [h(x) = y]$ to be the total weight of the balls hashing to bin y. Then for any constant $\gamma > 0$, X is strongly concentrated with added error probability n/m^γ, where the constants of the asymptotics are determined solely by c and γ. Furthermore, this concentration is query invariant.*

In Theorem 1, we note that the expectation, $\mu = \mathbb{E}[X]$, and the variance, $\sigma^2 = \mathrm{Var}[X]$, are the same as if h were a fully random hash function since h is 3-independent. This is true even when conditioning on the hash value of a query key having a specific value. The bound provided by Theorem 1 is therefore the same as the variance based Chernoff bound (3) except for a constant delay in the exponential decrease and an added error probability of n/m^γ. Since $\sigma^2 \leq \mu$, Theorem 1 also implies the classic one-sided Chernoff bounds (1) and (2), again with the constant delay and the added error probability as above, and a leading factor of 2.

Pătrașcu and Thorup [38] proved an equivalent probability bound, but without weights, and, more importantly, with the restriction that the number of bins $m \geq n^{1-1/(2c)}$. In particular, this implies the restriction $\mu \leq |\Sigma|^{1/2}$. Our new bound gives Chernoff-style concentration with high probability in n for any $m \geq n^\varepsilon$, $\varepsilon = \Omega(1)$. Indeed, letting $\gamma' = (\gamma + 1)/\varepsilon$, the added error probability becomes $n/m^{\gamma'} \leq 1/n^\gamma$.

However, for small m the error probability n/m^γ is prohibitive. For instance, unbiased coin tossing, corresponding to the case $m = 2$, has an added error probability of $n/2^\gamma$ which is useless. In Section 8, we will show that it is inherently impossible to get good concentration bounds using simple tabulation hashing when the number of bins m is small. To handle all instances, including those with few bins, and to support much more general Chernoff bounds, we introduce a new hash function: tabulation-permutation hashing.

1.2 Tabulation-Permutation Hashing

We start by defining *tabulation-permutation hashing* from Σ^c to Σ^d with $c, d = O(1)$. A tabulation-permutation hash function $h \colon \Sigma^c \to \Sigma^d$ is given as a composition, $h = \tau \circ g$, of a simple tabulation hash function $g \colon \Sigma^c \to \Sigma^d$ and a permutation $\tau \colon \Sigma^d \to \Sigma^d$. The permutation is a coordinate-wise fully random permutation: for each $j \in \{1, \dots, d\}$, pick a uniformly random character permutation $\tau_j : \Sigma \to \Sigma$. Now, $\tau = (\tau_1, \dots, \tau_d)$ in the sense that for $z = (z_1, \dots, z_d) \in \Sigma^d$, $\tau(z) = (\tau_1(z_1), \dots, \tau_d(z_d))$. In words, a tabulation-permutation hash function hashes c characters to d characters using simple tabulation, and then randomly permutes each of the d output characters. As is, tabulation-permutation hash functions yield values in Σ^d, but we will soon see how we can hash to $[m]$ for any $m \in \mathbb{N}$.

If we precompute tables $T_i \colon \Sigma \to \Sigma^d$, where

$$T_i(z_i) = \left(\overbrace{0, \dots, 0}^{i-1}, \tau_i(z_i), \overbrace{0, \dots, 0}^{d-i} \right), \quad z_i \in \Sigma,$$

then $\tau(z_1, \dots, z_d) = T_1(z_1) \oplus \dots \oplus T_d(z_d)$. Thus, τ admits the same implementation as simple tabulation, but with a special distribution on the character tables. If in particular $d \leq c$, the permutation step can be executed at least as fast as the simple tabulation step.

Our main result is that with tabulation-permutation hashing, we get high probability Chernoff-style bounds for the third and most general case described in the beginning of the introduction.

Theorem 2. *Let $h \colon [u] \to [r]$ be a tabulation-permutation hash function with $[u] = \Sigma^c$ and $[r] = \Sigma^d$, $c, d = O(1)$. Let $v \colon [u] \times [r] \to [0, 1]$ be a fixed value function that to each key $x \in [u]$ assigns a value $X_x = v(x, h(x)) \in [0, 1]$ depending on the hash value $h(x)$ and define $X = \sum_{x \in [u]} X_x$. For any constant $\gamma > 0$, X is strongly concentrated with added error probability $1/u^\gamma$, where the constants of the asymptotics are determined solely by c, d, and γ. Furthermore, this concentration is query invariant.*

Tabulation-permutation hashing inherits the 3-independence of simple tabulation, so as in Theorem 1, $\mu = \mathbb{E}[X]$ and $\sigma^2 = \text{Var}[X]$ have exactly the same values as if h were a fully-random hash function. Again, this is true even when conditioning on the hash value of a query key having a specific value.

Tabulation-permutation hashing allows us to hash into m bins for any $m \in \mathbb{N}$ (not necessarily a power of two) preserving the strong concentration from Theorem 2. To do so, simply define the hash function $h^m \colon [u] \to [m]$ by $h^m(x) = h(x) \bmod m$. Relating back to Theorem 1, consider a set $S \subseteq U$ of n balls where each ball $x \in S$ has a weight $w_x \in [0, 1]$ and balls x outside S are defined to have weight $w_x = 0$. To measure the total weight of the balls landing in a given bin $y \in [m]$, we define the value function $v(x, z) = w_x \cdot [z \bmod m = y]$. Then

$$X = \sum_{x \in [u]} v(x, h(x)) = \sum_{x \in S} w_x \cdot [h^m(x) = y]$$

is exactly the desired quantity and we get the concentration bound from Theorem 2. Then the big advantage of tabulation-permutation hashing over simple tabulation hashing is that it reduces the added error probability from n/m^γ of Theorem 1 to the $1/u^\gamma$ of Theorem 2, where u is the size of the key universe. Thus, with tabulation-permutation hashing, we actually get Chernoff bounds with high probability regardless of the number of bins.

Pătrașcu and Thorup [41] introduced twisted tabulation that like our tabulation-permutation achieved Chernoff-style concentration bounds with a general value function v. Their bounds are equivalent to those of Theorem 2, but only under the restriction $\mu \leq |\Sigma|^{1-\Omega(1)}$. To understand how serious this restriction is, consider again tossing an unbiased coin for each key x in a set $S \subseteq [u]$, corresponding to the case $m = 2$ and $\mu = |S|/2$. With the restriction from [41], we can only handle $|S| \leq 2 |\Sigma|^{1-\Omega(1)}$, but recall that Σ is chosen small enough for character tables to fit in fast cache, so this rules out any moderately large data set. We are going to show that for certain sets S, twisted tabulation has the same problems as simple tabulation when hashing to few bins. This implies that the restrictions from [41] cannot be lifted with a better analysis.

4

Pătraşcu and Thorup [41] were acutely aware of how prohibitive the restriction $\mu \leq |\Sigma|^{1-\Omega(1)}$ is. For unbounded μ, they proved a weaker bound; namely that with twisted tabulation hashing, $X = \mu \pm O(\sigma(\log u)^{c+1})$ with probability $1 - u^{-\gamma}$ for any $\gamma = O(1)$. With our tabulation-permutation hashing, we get $X = \mu \pm O(\sigma(\log u)^{1/2})$ with the same high probability, $1 - u^{-\gamma}$. Within a constant factor on the deviation, our high probability bound is as good as with fully-random hashing.

More related work, including Siegel's [44] and Thorup's [45] highly independent hashing will be discussed in Section 1.7.

1.3 Tabulation-1Permutation

Above we introduced tabulation-permutation hashing which yields Chernoff-style bounds with an arbitrary value function. This is the same general scenario as was studied for twisted tabulation in [41]. However, for almost all applications we are aware of, we only need the generality of the second case presented at the beginning of the introduction. Recall that in this case we are only interested in the total weight of the balls hashing to a certain interval. As it turns out, a significant simplification of tabulation-permutation hashing suffices to achieve strong concentration bounds. We call this simplification *tabulation-1permutation*. Tabulation-permutation hashing randomly permutes each of the d output characters of a simple tabulation function $g \colon \Sigma^c \to \Sigma^d$. Instead, tabulation-1permutation only permutes the most significant character.

More precisely, a tabulation-1permutation hash function $h \colon \Sigma^c \to \Sigma^d$ is a composition, $h = \tau \circ g$, of a simple tabulation function, $g \colon \Sigma^c \to \Sigma^d$, and a random permutation, $\tau \colon \Sigma^d \to \Sigma^d$, of the most significant character, $\tau(z_1, \ldots, z_d) = (\tau_1(z_1), z_2, \ldots, z_d)$ for a random character permutation $\tau_1 \colon \Sigma \to \Sigma$.

To simplify the implementation of the hash function and speed up its evaluation, we can precompute a table $T \colon \Sigma \to \Sigma^d$ such that for $z_1 \in \Sigma$,

$$T(z_1) = \left(z_1 \oplus \tau_1(z_1), \overbrace{0, \ldots, 0}^{d-1} \right).$$

Then if $g(x) = z = (z_1, \ldots, z_d)$, $h(x) = z \oplus T(z_1)$.

This simplified scheme, needing only $c+1$ character lookups, is powerful enough for concentration within an arbitrary interval.

Theorem 3. *Let $h \colon [u] \to [r]$ be a tabulation-1permutation hash function with $[u] = \Sigma^c$ and $[r] = \Sigma^d$, $c, d = O(1)$. Consider a key/ball set $S \subseteq [u]$ of size $n = |S|$ where each ball $x \in S$ is assigned a weight $w_x \in [0, 1]$. Choose arbitrary hash values $y_1, y_2 \in [r]$ with $y_1 \leq y_2$. Define $X = \sum_{x \in S} w_x \cdot [y_1 \leq h(x) < y_2]$ to be the total weight of balls hashing to the interval $[y_1, y_2)$. Then for any constant $\gamma > 0$, X is strongly concentrated with added error probability $1/u^\gamma$, where the constants of the asymptotics are determined solely by c, d, and γ. Furthermore, this concentration is query invariant.*

One application of Theorem 3 is in the following sampling scenario: We set $y_1 = 0$, and sample all keys with $h(x) < y_2$. Each key is then sampled with probability y_2/r, and Theorem 3 gives concentration on the number of samples. In [1] this is used for more efficient implementations of streaming algorithms.

Another application is efficiently hashing into an arbitrary number $m \leq r$ of bins. We previously discussed using hash values modulo m, but a general mod-operation is often quite slow. Instead we can think of hash values as fractions $h(x)/r \in [0, 1)$. Multiplying by m, we get a value in $[0, m)$, and the bin index is then obtained by rounding down to the nearest integer. This implementation is very efficient because r is a power of two, $r = 2^b$, so the rounding is obtained by a right-shift by b bits. To hash a key x to $[m]$, we simply compute $h^m(x) = (h(x) * m) \gg b$. Then x hashes to bin $d \in [m]$ if and only if $d \in [y_1, y_2) \subseteq [r]$ where $y_1 = \lfloor rd/m \rfloor$ and $y_2 = \lfloor r(d+1)/m \rfloor$, so the number of keys hashing to a bin is concentrated as in Theorem 3. Moreover, h^m uses only $c+1$ character lookups and a single multiplication in addition to some very fast shifts and bit-wise Boolean operations.

1.4   Subpolynomial Error Probabilities

In Theorem 2 and 3, we have $\Pr[|X - \mu| \geq t] = O(\exp(-\Omega(\sigma^2 \mathcal{C}(t/\sigma^2)))) + 1/u^\gamma$ which holds for any fixed γ. The value of γ affects the constant hidden in the Ω-notation delaying the exponential decrease. In Section 8, we will show that the same bound does not hold if γ is replaced by any slow-growing but unbounded function. Nevertheless, it follows from our analysis that for every $\alpha(u) = \omega(1)$ there exists $\beta(u) = \omega(1)$ such that whenever $\exp(-\sigma^2 \mathcal{C}(t/\sigma^2)) < 1/u^{\alpha(u)}$, $\Pr[|X - \mu| \geq t] \leq 1/u^{\beta(u)}$.

1.5   Generic Remarks on Universe Reduction and Amount of Randomness

The following observations are fairly standard in the literature. Suppose we wish to hash a set of keys S belonging to some universe $\mathcal{U}$. The universe may be so large compared to S that it is not efficient to directly implement a theoretically powerful hashing scheme like tabulation-permutation hashing. A standard first step is to perform a *universe reduction*, mapping $\mathcal{U}$ randomly to "signatures" in $[u] = \{0, 1, \ldots, u-1\}$, where $u = n^{O(1)}$, e.g. $u = n^3$, so that no two keys from S are expected to get the same signature [9]. As the only theoretical property required for the universe reduction is a low collision probability, this step can be implemented using very simple hash functions as described in [46]. In this paper, we generally assume that this universe reduction has already been done, if needed, hence that we only need to deal with keys from a universe $[u]$ of size polynomial in n. For any small constant $\varepsilon > 0$ we may thus pick $c = O(1/\varepsilon)$ such that the space used for our hash tables, $\Theta(|\Sigma|)$, is $O(n^\varepsilon)$. Practically speaking, this justifies focusing on the hashing of 32- and 64-bit keys.

When we defined simple tabulation above, we said the character tables were fully random. However, for the all the bounds in this paper, it would suffice if they were populated with a $O(\log u)$-independent pseudo-random number generator (PNG), so we only need a seed of $O(\log u)$ random words to be shared among all applications who want to use the same simple tabulation hash function. Then, as a preprocesing for fast hashing, each application can locally fill the character tables in $O(|\Sigma|)$ time [13]. Likewise, for our tabulation permutation hashing, our bounds only require a $O(\log u)$-independent PNG to generate the permutations. The basic point here is that tabulation based hashing does not need a lot of randomness to fill the tables, but only space to store the tables as needed for the fast computation of hash values.

1.6   Techniques

The paper relies on three main technical insights to establish the concentration inequality for tabulation-permutation hashing of Theorem 2. We shall here describe each of these ideas and argue that each is in fact necessary towards an efficient hash function with strong concentration bounds.

1.6.1   Improved Analysis of Simple Tabulation

The first step towards proving Theorem 2 is to better understand the distribution of simple tabulation hashing. We describe below how an extensive combinatorial analysis makes it possible to prove a generalised version of Theorem 1.

To describe the main idea of this technical contribution, we must first introduce some ideas from previous work in the area. This will also serve to highlight the inherent limitations of previous approaches. A simplified account is the following. Let $h \colon \Sigma^c \to [m]$ be a simple tabulation hash function, let $y \in [m]$ be given, and for some subset of keys $S \subseteq \Sigma^c$, let $X = \sum_{x \in S}[h(x) = y]$ be the random variable denoting the number of elements $x \in S$ that have hash value $h(x) = y$. Our goal is to bound the deviation of X from its mean $\mu = |S|/m$. We first note that picking a random simple tabulation hash function $h : \Sigma^c \to [m]$ amounts to filling the c character tables, each of size Σ, with uniformly random hash values. Thus, picking a simple tabulation hash function $h : \Sigma^c \to [m]$ corresponds to picking a uniformly random hash function $h \colon [c] \times \Sigma \to [m]$. We call $[c] \times \Sigma$ the set of *position characters*. Viewing a key $x = (x_1, \ldots, x_c) \in \Sigma^c$ as a set of position characters, $x = \{(1, x_1), \ldots, (c, x_c)\}$, and slightly abusing notation, it then holds that $h(x) = \bigoplus_{\alpha \in x} h(\alpha)$. Now let $\alpha_1, \ldots, \alpha_r$ be a (for the sake of the proof) well-chosen ordering of the position

characters. For each $k \in [r+1]$, we define the random variable $X_k = \mathbb{E}\left[X \mid h(\alpha_1), \ldots, h(\alpha_k)\right]$, where $h(\alpha_i)$ is the value of the entry of the lookup table of h corresponding to α_i. The process $(X_k)_{k=0}^r$ is then a martingale. We can view this as revealing the lookup table of h one entry at a time and adjusting our expectation of the outcome of X accordingly. Defining the martingale difference $Y_k = X_k - X_{k-1}$, we can express X as a sum $X = \mu + \sum_{k=1}^{c \cdot |\Sigma|} Y_k$. Previous work has then bounded the sum using a Chernoff inequality for martingales as follows. Due to the nature of the ordering of $\{\alpha_i\}_{i=1}^r$, we can find $M > 0$ such that with high probability, $|Y_k| \leq M$ for every k. Then conditioned on each of the Y_ks being bounded, X satisfies the Chernoff bounds of (1) and (2) except the exponent is divided by M. As long as the expectation, μ, satisfies $\mu = O(|\Sigma|)$, it is possible[2] that $M = O(1)$, yielding Chernoff bounds with a constant in the delay of the exponential decrease. However, since there are only $c \cdot |\Sigma|$ variables, Y_k, it is clear that $M \geq \mu/(c \cdot |\Sigma|)$. Thus, whenever $\mu = \omega(|\Sigma|)$, the delay of the exponential decrease is super-constant, meaning that we do not get asymptotically tight Chernoff-style bounds. This obstacle has been an inherent issue with the previous techniques in analysing both simple tabulation [38] as well as twisted tabulation [41]. Being unable to bound anything beyond the absolute deviation of each variable Y_k, it is impossible to get good concentration bounds for large expectations, μ.

Going beyond the above limitation, we dispense with the idea of bounding absolute deviations and instead bound the sum of variances, $\sigma^2 = \sum_{k=1}^{c \cdot |\Sigma|} \text{Var}\left[Y_k\right]$. This sum has a combinatorial interpretation relating to the number of collisions of hash keys, i.e., the number of pairs $y_1, y_2 \in \Sigma^c$ with $h(y_1) = h(y_2)$.

An extensive combinatorial analysis of simple tabulation hashing yields high-probability bounds on the sum of variances that is tight up to constant factors. This is key in establishing an induction that allows us to prove Theorem 1. Complementing our improved bounds, we will show that simple tabulation hashing inherently does not support Chernoff-style concentration bounds for small m.

1.6.2   Permuting the Hash Range

Our next step is to consider the hash function $h = \tau \circ g \colon \Sigma^c \to \Sigma$ where $g \colon \Sigma^c \to \Sigma$ is a simple tabulation hash function and $\tau \colon \Sigma \to \Sigma$ is a uniformly random permutation. Our goal is to show that h provides good concentration bounds for any possible value function. To showcase our approach, we consider the example of hashing to some small set, $[m]$, of bins, e.g., with $m = 2$ as in our coin tossing example. This can be done using the hash function $h^m \colon \Sigma^c \to [m]$ defined by $h^m(x) = (h(x) \mod m)$. For simplicity we assume that m is a power of two, or equivalently, that m divides $|\Sigma|$. We note that the case of small m was exactly the case that could not be handled with simple tabulation hashing alone.

Let us look at the individual steps of h^m. First, we use simple tabulation mapping into the "character bins", Σ. The number of balls in any given character bin is nicely concentrated, but only because $|\Sigma|$ is large. Next, perform a permutation followed by the mod m operation. The last two steps correspond to the way we would deal a deck of $|\Sigma|$ cards into m hands. The cards are shuffled by a random permutation, then dealt to the m players one card at a time in cyclic order. The end result is that each of the final m bins is assigned exactly $|\Sigma|/m$ random character bins. An important point is now that because the partitioning is *exact*, the error in the number of balls in a final bin stems solely from the errors in the $|\Sigma|/m$ character bins, and because the partitioning is *random*, we expect the positive and negative errors to cancel out nicely. The analysis, which is far from trivial, requires much more than these properties. For example, we also need the bound described in Section 1.6.1 on the sum of variances. This bound ensures that not only is the number of balls in the individual character bins nicely concentrated around the mean, but moreover, there is only a small number of character bins for which the error is large. That these things combine to yield strong concentration, not only in the specific example above, but for general value functions as in Theorem 2, is quite magical.

We finish the discussion by mentioning two approaches that do not work and highlight how a permutation solves the issues of these strategies.

First, one may ask why we need the permutation at all. After all, the mod m operation also partitions the $|\Sigma|$ character bins into groups of the same size, $|\Sigma|/m$. The issue is that while a simple tabulation hash

---

[2]In [38], the actual analysis of simple tabulation using this approach achieves $\mu = O(\sqrt{|\Sigma|})$.

function, $g : \Sigma^c \to \Sigma$, has good concentration in each of the individual character bins, the $|\Sigma|/m$ character bins being picked out by the mod m operation constitute a very structured subset of Σ, and the errors from this set of bins could be highly correlated. We indeed show that the structure of simple tabulation causes this to happen for certain sets of keys, both theoretically (Section 8) and experimentally (Appendix A).

Second, the reader may wonder why we use a permutation, $\tau \colon \Sigma \to \Sigma$, instead of a random hash function as in double tabulation [45]. In terms of the card dealing analogy, this would correspond to throwing the $|\Sigma|$ cards at the m astonished card players one at a time with a random choice for each card, not guaranteeing that the players each get the same number of cards. And this is exactly the issue. Using a fully random hash function τ', we incur an extra error stemming from τ' distributing the $|\Sigma|$ character bins unevenly into the final bins. This is manifested in the variance of the number of balls hashing to a specific bin: Take again the coin tossing example with $n \geq |\Sigma|$ balls being distributed into $m = 2$ bins. With a permutation τ the hash function becomes 2-independent, so the variance is the same as in the fully random setting, $n/4$. Now even if the simple tabulation hash function, g, distributes the n keys into the character bins evenly, with exactly n/Σ keys in each, with a fully random hash function, τ', the variance becomes $(n/|\Sigma|)^2 \cdot |\Sigma|/4 = n^2/(4|\Sigma|)$, a factor of $n/|\Sigma|$ higher.

1.6.3 Squaring the Hash Range

The last piece of the puzzle is a trick to extend the range of a hash function satisfying Chernoff-style bounds. We wish to construct a hash function $h \colon \Sigma^c \to [m]$ satisfying Chernoff-style bounds for m arbitrarily large as in Theorem 2. At first sight, the trick of the previous subsection would appear to suffice for the purpose. However, if we let $g = \tau \circ h$ be the composition of a simple tabulation hash function $h \colon \Sigma^c \to [m]$ and τ a random permutation of $[m]$, we run into trouble if for instance $[m] = \Sigma^c$. In this case, a random permutation of $[m]$ would require space equal to that of a fully random function $f \colon \Sigma^c \to [m]$, but the whole point of hashing is to use less space. Hence, we instead prove the following. Let $a \colon C \to D$ and $b \colon C \to D$ be two independent hash functions satisfying Chernoff-style bounds for general value functions. Then this property is preserved up to constant factors under "concatenation", i.e., if we let $c \colon C \to D^2$ be given by $c(x) = (a(x), b(x))$, then c is also a hash function satisfying Chernoff-style bounds for general value functions, albeit with a slightly worse constant delay in the exponential decrease than a and b. Thus, this technique allows us to "square" the range of a hash function.

With this at hand, let $h_1, h_2 \colon \Sigma^c \to \Sigma$ be defined as $h_1 = \tau_1 \circ g_1$ and $h_2 = \tau_2 \circ g_2$, where $g_1, g_2 \colon \Sigma^c \to \Sigma$ are simple tabulation hash functions and $\tau_1, \tau_2 \colon \Sigma \to \Sigma$ are random permutations. Then the concatenation $h \colon \Sigma^c \to \Sigma^2$ of h_1 and h_2 can be considered a composition of a simple tabulation function $g \colon \Sigma^c \to \Sigma^2$ given by $g(x) = (g_1(x), g_2(x))$ and a coordinate-wise permutation $\tau = (\tau_1, \tau_2) \colon \Sigma^2 \to \Sigma^2$, where the latter is given by $\tau(x_1, x_2) = (\tau_1(x_1), \tau_2(x_2)), x_1, x_2 \in \Sigma$. Applying our composition result, gives that g also satisfies Chernoff-style bounds. Repeating this procedure $\lceil \log(d) \rceil = O(1)$ times, yields the desired concentration bound for tabulation-permutation hashing $h \colon \Sigma^c \to \Sigma^d$ described in Theorem 2.

1.7 Related Work – Theoretical and Experimental Comparisons

In this section, we shall compare the performance of tabulation-permutation and tabulation-1permutation hashing with other related results. Our comparisons are both theoretical and empirical. Our goal in this paper is fast constant-time hashing having strong concentration bounds with high probability, i.e., bounds of the form

$$\Pr[|X - \mu| \geq t] \leq 2 \exp(-\Omega(\sigma^2 \mathcal{C}(t/\sigma^2))) + u^{-\gamma},$$

as in Definition 1 and Theorems 2 and 3, or possibly with σ^2 replaced by $\mu \geq \sigma^2$. Theoretically, we will only compare with other hashing schemes that are relevant to this goal. In doing so, we distinguish between the hash functions that achieve Chernoff-style bounds with restrictions on the expected value and those that, like our new hash functions, do so without such restrictions, which is what we want for all possible input. Empirically, we shall compare the practical evaluation time of tabulation-permutation and permutation-1permutation to the fastest commonly used hash functions and to hash functions with similar

| | Running time (ms) | | | |
| Hash function | Computer 1 | | Computer 2 | |
	32 bits	64 bits	32 bits	64 bits
Multiply-Shift	4.2	7.5	23.0	36.5
2-Independent PolyHash	14.8	20.0	72.2	107.3
Simple Tabulation	13.7	17.8	53.1	55.9
Twisted Tabulation	17.2	26.1	65.6	92.5
Mixed Tabulation	28.6	68.1	120.1	236.6
Tabulation-1Permutation	16.0	19.3	63.8	67.7
Tabulation-Permutation	27.3	43.2	118.1	123.6
Double Tabulation	1130.1	–	3704.1	–
"Random" (100-Independent PolyHash)	2436.9	3356.8	7416.8	11352.6

Table 1: The time for different hash functions to hash 10^7 keys of length 32 bits and 64 bits, respectively, to ranges of size 32 bits and 64 bits. The experiment was carried out on two computers. The hash functions written in italics are those without general Chernoff-style bounds. Hash functions written in bold are the contributions of this paper. The hash functions in regular font are known to provide Chernoff-style bounds. Note that we were unable to implement double tabulation from 64 bits to 64 bits since the hash tables were too large to fit in memory.

Hash function	Time	Space	Concentration Guarantee	Restriction
Multiply-Shift	$O(1)$	$O(1)$	Chebyshev's inequality	None
k-Independent PolyHash	$O(k)$	$O(k)$	Chernoff-style bounds	Requires $k = \Omega(\log u)$ for added error probability $O(1/u^\gamma)$
Simple Tabulation	$O(c)$	$O(u^{1/c})$	Chernoff-style bounds	Added error probability: $O(n/m^\gamma)$
Twisted Tabulation	$O(c)$	$O(u^{1/c})$	Chernoff-style bounds	Requires: $\mu \leq |\Sigma|^{1-\Omega(1)}$
Mixed Tabulation	$O(c)$	$O(u^{1/c})$	Chernoff-style bounds	Requires: $\mu = o(|\Sigma|)$
Tabulation-Permutation	$O(c)$	$O(u^{1/c})$	Chernoff-style bounds	Added error probability: $O(1/u^\gamma)$
Double Tabulation	$O(c^2)$	$O(u^{1/c})$	Chernoff-style bounds	Added error probability: $O(1/u^\gamma)$

Table 2: Theoretical time and space consumption of some of the hash functions discussed.

theoretical guarantees. A major goal of algorithmic analysis is to understand the theoretical behavior of simple algorithms that work well in practice, providing them with good theoretical guarantees such as worst-case behavior. For instance, one may recall Knuth's analysis of linear probing [29], showing that this very practical algorithm has strong theoretical guarantees. In a similar vein, we not only show that the hashing schemes of tabulation-permutation and tabulation-1permutation have strong theoretical guarantees, we also perform experiments, summarized in Table 1, demonstrating that in practice they are comparable in speed to some of the most efficient hash functions in use, none of which have similar concentration guarantees. Thus, with our new hash functions, hashing with strong theoretical concentration guarantees is suddenly feasible for time-critical applications.

1.7.1 High Independence and Tabulation

Before this paper, the only known way to obtain unrestricted Chernoff-style concentration bounds with hash functions that can be evaluated in constant time was through k-independent hashing. Recall that a hash function $h : U \to R$ is k-independent if the distribution of $(h(x_1), \ldots, h(x_k))$ is uniform in R^k for every choice of distinct keys $x_1, \ldots, x_k \in U$. Schmidt, Siegel, and Srinivasan [43] have shown that with k-independent hashing, we have Chernoff-style concentration bounds in all three cases mentioned at the beginning of the

introduction up to an added error probability decreasing exponentially in k. With $k = \Theta(\gamma \log u)$, this means Chernoff-style concentration with an added error probability of $1/u^\gamma$ like in Theorem 2 and 3. However, evaluating any k-independent hash function takes time $\Omega(k)$ unless we use a lot of space. Indeed, a cell probe lower bound by Siegel [44] states that evaluating a k-independent hash function over a key domain $[u]$ using $t < k$ probes, requires us to use at least $u^{1/t}$ cells to represent the hash function. Thus, aiming for Chernoff concentration through k-independence with $k = \Omega(\log u)$ and with constant evaluation time, we would have to use $u^{\Omega(1)}$ space like our tabulation-permutation. Here it should be mentioned that k-independent PolyHash modulo a prime p can be evaluated at k points in total time $O(k \log^2 k)$ using multipoint evaluation methods. Then the average evaluation time is $O(\log^2 k)$, but it requires that the hashing can be done to batches of k keys at a time. We can no longer hash one key at a time, continuing with other code before we hash the next key. This could be a problem for some applications. A bigger practical issue is that it is no longer a black box implementation of a hash function. To understand the issue, think of Google's codebase where thousands of programs are making library calls to hash functions. A change to multipoint evaluation would require rewriting all of the calling programs, checking in each case that batch hashing suffices — a huge task that likely would create many errors. A final point is that multipoint evaluation is complicated to implement yet still not as fast as our tabulation-permutation hashing. Turning to upper bounds, Siegel designed a $u^{\Omega(1/c^2)}$-independent hash function that can be represented in tables of size $u^{1/c}$ and evaluated in $c^{O(c)}$ time. With $c = O(1)$, this suffices for Chernoff-style concentration bounds by the argument above. However, as Siegel states, the hashing scheme is "far too slow for any practical application".

In the setting of Siegel, Thorup's double tabulation [45] is a simpler and more efficient construction of highly independent hashing. It is the main constant-time competitor of our new tabulation-permutation hashing, and yet it is 30 times slower in our experiments. In the following, we describe the theoretical guarantees of double tabulation hashing and discuss its concrete parameters in terms of speed and use of space towards comparing it with tabulation-permutation hashing.

A *double tabulation* hash function, $h : \Sigma^c \to \Sigma^c$ is the composition of two independent simple tabulation hash functions $h_1 : \Sigma^c \to \Sigma^d$ and $h_2 : \Sigma^d \to \Sigma^c$, $h = h_2 \circ h_1$. Evaluating the function thus requires $c + d$ character lookups. Assuming that each memory unit stores an element from $[u] = \Sigma^c$ and $d \geq c$, the space used for the character tables is $(c(d/c) + d)u^{1/c} = 2du^{1/c}$. Thorup [45] has shown that if $d \geq 6c$, then with probability $1 - o(\Sigma^{2-d/(2c)})$ over the choice of h_1, the double tabulation hash function h is k-independent for $k = |\Sigma|^{1/(5c)} = u^{\Omega(1/c^2)}$. More precisely, with this probability, the output keys $(h_1(x))_{x \in \Sigma^c}$ are distinct, and h_2 is k-independent when restricted to this set of keys. If we are lucky to pick such an h_1, this means that we get the same high indepence as Siegel [44]. With $d = 6c$, the space used is $12cu^{1/c} = O(cu^{1/c})$ and the number of character lookups to compute a hash value is $7c = O(c)$. Tabulation-permutation hashing is very comparable to Thorup's double tabulation. As previously noted, it can be implemented in the same way, except that we fill the character tables of h_2 with permutations and padded zeros instead of random hash values. To compare, a tabulation-permutation hash function $h : \Sigma^c \to \Sigma^c$ requires $2c$ lookups and uses space $2cu^{1/c}$, which may not seem a big difference. However, in the following, we demonstrate how restrictions on double tabulation cost an order of magnitude in speed and space compared with tabulation-permutation hashing when used with any realistic parameters.

With Thorup's double tabulation, for $(\log u)$-independence, we need $\log u \leq |\Sigma|^{1/(5c)} = u^{1/(5c^2)}$. In choosing values for u and c that work in practice, this inequality is very restrictive. Indeed, even for $c = 2$, $\log u \leq u^{1/20}$, which roughly implies that $\log u \geq 140$. Combined with the fact that the character tables use space $12c|\Sigma|$, and that $|\Sigma| \geq (\log u)^{5c}$, this is an intimidating amount of space. Another problem is the error probability over h_1 of $1 - o(\Sigma^{2-d/(2c)})$. If we want this to be $O(1/u)$, like in the error bounds from Theorem 2 and 3, we need $d \geq 2(c^2 + 2c)$. Thus, while things work well asymptotically, these constraints make it hard to implement highly independent double tabulation on any normal computer. However, based on a more careful analysis of the case with 32-bit keys, Thorup shows that using $c = 2$ characters of 16 bits, and $d = 20$ derived characters, gives a 100-independent hash function with probability $1 - 1.5 \times 10^{-42}$. According to [45] we cannot use significantly fewer resources even if we just want 4-independence. For hashing 32-bit keys, this means making 22 lookups for each query and using tables of total size $40 \cdot 2^{16}$. In contrast, if we hash 32-bit keys with tabulation-permutation hashing, we may use 8-bit characters with $d = c = 4$, thus making

only 8 lookups in tables of total size $8 \cdot 2^8$. For this setting of parameters, our experiments (summarized in Table 1) show that double tabulation is approximately 30 times slower than tabulation-permutation hashing. For 64-bit keys, Thorup [45] suggests implementing double tabulation with $c = 3$ characters of 22 bits and $d = 24$. This would require 26 lookups in tables of total size $48 \cdot 2^{22}$. We were not able to implement this on a regular laptop due to the space requirement.

We finally mention that Christani et al. [14] have presented a hash family which obtains the even higher independence $u^{\Omega(1/c)}$. The scheme is, however, more complicated with a slower evaluation time of $\Theta(c \log c)$.

1.7.2 Space Bounded Independence and Chernoff Bounds

One of the earliest attempts of obtaining strong concentration bounds via hashing is a simple and elegant construction by Dietzfelbinger and Meyer auf der Heide [19]. For some parameters m, s, d, their hash family maps to $[m]$, can be represented with $O(s)$ space, and uses a $(d+1)$-independent hash function as a subroutine, where $d = O(1)$, e.g., a degree-d polynomial. In terms of our main goal of Chernoff-style bounds, their result can be cast as follows: Considering the number of balls from a fixed, but unknown, subset $S \subseteq U$, with $|S| = n$, that hashes to a specific bin, their result yields Chernoff bounds like ours with a constant delay in the exponential decrease and with an added error probability of $n \left(\frac{n}{ms} \right)^d$. The expected number of balls in a given bin is $\mu = n/m$, so the added error probability is $n(\mu/s)^d$. To compare with tabulation-permutation, suppose we insist on using space $O(|\Sigma|)$ and that we moreover want the added error probability to be $u^{-\gamma} = |\Sigma|^{-c\gamma}$ like in Theorems 2 and 3. With the hashing scheme from [19], we then need $\mu = O(|\Sigma|^{1-\gamma c/d})$. If we want to be able to handle expectations of order, e.g. $|\Sigma|^{1/2}$, we thus need $d \geq 2c\gamma$. For 64-bit key, $c = 8$, and $\gamma = 1$, say, this means that we need to evaluate a 16-independent hash function. In general, we see that the concentration bound above suffers from the same issues as those provided by Pătrașcu and Thorup for simple and twisted tabulation [38, 41], namely that we only have Chernoff-style concentration if the expected value is much smaller than the space used.

Going in a different direction, Dietzfelbinger and Rink [20] use universe splitting to create a hash function that is highly independent (building on previous works [21, 22, 25, 27]) but, contrasting double tabulation as described above, only within a fixed set S, not the entire universe. The construction requires an upper bound n on the size of S, and a polynomial error probability of $n^{-\gamma}$ is tolerated. Here $\gamma = O(1)$ is part of the construction and affects the evaluation time. Assuming no such error has occurred, which is not checked, the hash function is highly independent on S. As with Siegel's and Thorup's highly independent hashing discussed above, this implies Chernoff bounds without the constant delay in the exponential decrease, but this time only within the fixed set S. In the same setting, Pagh and Pagh [37] have presented a hash function that uses $(1 + o(1))n$ space and which is fully independent on any given set S of size at most n with high probability. This result is very useful, e.g., as part of solving a static problem of size n using linear space, since, with high probability, we may assume fully-random hashing as a subroutine. However, from a Chernoff bound perspective, the fixed polynomial error probability implies that we do not benefit from any independence above $O(\log n)$, using the aforementioned results from [43]. More importantly, we do not want to impose any limitations to the size of the sets we wish to hash in this paper. Consider for example the classic problem of counting distinct elements in a huge data stream. The size of the data stream might be very large, but regardless, the hashing schemes of this paper will only use space $O(u^{1/c})$ with c chosen small enough for hash tables to fit in fast cache.

Finally, Dahlgaard et al. [16] have shown that on a given set S of size $|S| \leq |\Sigma|/2$ a double tabulation hash function, $h = h_2 \circ h_1$ as described above, is fully random with probability $1 - |\Sigma|^{1-\lfloor d/2 \rfloor}$ over the choice of h_1. For an error probability of $1/u$, we set $d = (2c + 2)$ yielding a hash function that can be evaluated with $3c + 2$ character lookups and using $(4c + 4)|\Sigma|$ space. This can be used to simplify the above construction by Pagh and Pagh [37]. Dahlgaard et al. [16] also propose mixed tabulation hashing which they use for statistics over k-partitions. Their analysis is easily modified to yield Chernoff-style bounds for intervals similar to our bounds for tabulation-1permuation hashing presented in Theorem 3, but with the restriction that the expectation μ is at most $|\Sigma|/\log^{2c}|\Sigma|$. This restriction is better than the earlier mentioned restictions $\mu \leq |\Sigma|^{1/2}$ for simple tabulation [38] and $\mu \leq |\Sigma|^{1-\Omega(1)}$ for twisted tabulation [41].

For mixed tabulation hashing, Dahlgaard et al. use $3c + 2$ lookups and $(5c + 4)|\Sigma|$ space. In comparison, tabulation-1permutation hashing, which has no restriction on μ, uses only $c+1$ lookups and $(c+1)|\Sigma|$ space.

1.7.3 Small Space Alternatives in Superconstant Time

Finally, there have been various interesting developments regarding hash functions with small representation space that, for example, can hash n balls to n bins such that the maximal number of balls in any bin is $O(\log n / \log \log n)$, corresponding to a classic Chernoff bound. Accomplishing this through independence of the hash function, this requires $O(\log n / \log \log n)$-independence and evaluation time unless we switch to hash functions using a lot of space as described above. However, [10, 33] construct hash families taking a random seed of $O(\log \log n)$ words and which can be evaluated using $O((\log \log n)^2)$ operations, still obtaining a maximal load in any bin of $O(\log n / \log \log n)$ with high probability. This is impressive as it only uses a small amount of space and a short random seed, though it does require some slightly non-standard operations when evaluating the hash functions. The running time however, is not constant, which is what we aim for in this paper.

A different result is by [26] who construct hash families which hash n balls to 2 bins. They construct hash families that taking a random seed of $O((\log \log n)^2)$ words get Chernoff bounds with an added error probability of $n^{-\gamma}$ for some constant γ, which is similar to our bounds. Nothing is said about the running time of the hash function of [26]. Since one of our primary goals is to design hash functions with constant running time, this makes the two results somewhat incomparable.

1.7.4 Experiments and Comparisons

To better understand the real-world performance of our new hash functions in comparison with well-known and comparable alternatives, we performed some simple experiments on regular laptops, as presented in Table 1. We did two types of experiments.

- On the one hand we compared with potentially faster hash functions with weaker or restricted concentration bounds to see how much we lose in speed with our theoretically strong tabulation-permutation hashing. We shall see that our tabulation-permutation is very competitive in speed.

- On the other hand we compared with the fastest previously known hashing schemes with strong concentration bounds like ours. Here we will see that we gain a factor of 30 in speed.

Concerning weaker, but potentially faster, hashing schemes we have chosen two types of hash functions for the comparison. First, we have the fast 2-independent hash functions multiply-shift (with addition) and 2-independent PolyHash. They are among the fastest hash functions in use and are commonly used in streaming algorithms. It should be noted that when we use 2-independent hash functions, the variance is the same as with full randomness, and it may hence suffice for applications with constant error probability. Furthermore, for data sets with sufficient entropy, Chung, Mitzenmacher, and Vadhan [15] show that 2-independent hashing suffices. However, as previously mentioned, we want provable Chernoff-style concentration bounds of our hash functions, equivalent up to constant factors to the behavior of a fully random hash function, for any possible input. Second, we have simple tabulation, twisted tabulation, and mixed tabulation, which are tabulation based hashing schemes similar to tabulation-1permutation and tabulation-permutation hashing, but with only restricted concentration bounds. It is worth noting that Dahlgaard, Knudsen, and Thorup [17] performed experiments showing that the popular hash functions MurmurHash3 [3] and CityHash [40] along with the cryptographic hash function Blake2 [4] all are slower than mixed tabulation hashing, which we shall see is even slower than permutation-tabulation hashing. These hash functions are used in practice, but given that our experiments show mixed tabulation to be slightly slower than tabulation-permutation hashing, these can now be replaced with our faster alternatives that additionally provide theoretical guarantees as to their effectiveness.

Concerning hashing schemes with previous known strong concentration bounds, we compared with double tabulation and 100-independent PolyHash, which are the strongest competitors that we are aware of using simple portable code.

The experiment measures the time taken by various hash functions to hash a large set of keys. Since the hash functions considered all run the same instructions for all keys, the worst- and best-case running times are the same, and hence choosing random input keys suffices for timing purposes. Further technical details of the experiments are covered in Appendix A. We considered both hashing 32-bit keys to 32-bit hash values and 64-bit keys to 64-bits hash values. We did not consider larger key domains as we assume that a universe reduction, as described in Section 1.5, has been made if needed. The results are presented in Table 1. Below, we comment on the outcome of the experiment for each scheme considered.

Multiply-Shift. The fastest scheme of the comparison is Dietzfelbinger's 2-independent Multiply-Shift [18]. For 32-bit keys it uses one 64-bit multiplication and a shift. For 64-bit keys it uses one 128-bit multiplication and a shift. As expected, this very simple hash function was the fastest in the experiment.

2-Independent PolyHash. We compare twice with the classic k-independent PolyHash [48]. Once with $k = 2$ and again with $k = 100$. k-independent PolyHash is based on evaluating a random degree $(k-1)$-polynomial over a prime field, using Mersenne primes to make it fast: $2^{61} - 1$ for 32-bit keys and $2^{89} - 1$ for 64-bit keys. The 2-independent version was 2-3 times slower in experiments than multiply-shift. It is possible that implementing PolyHash with a specialized carry-less multiplication [31] would provide some speedup. However, we do not expect it to become faster than multiply-shift.

Simple Tabulation. The baseline for comparison of our tabulation-based schemes is simple tabulation hashing. Recall that we hash using c characters from $\Sigma = [u^{1/c}]$ (in this experiment we considered $u = 2^{32}$ and $u = 2^{64}$). This implies c lookups from the character tables, which have total size $c |\Sigma|$. For each lookup, we carry out a few simple AC^0 operations, extracting the characters for the lookup and applying an XOR. Since the size of the character alphabet influences the lookup times, it is not immediately clear, which choice of c will be the fastest in practice. This is, however, easily checkable on any computer by simple experiments. In our case, both computers were fastest with 8-bit characters, hence with all character tables fitting in fast cache.

Theoretically, tabulation-based hashing methods are incomparable in speed to multiply-shift and 2-independent PolyHash, since the latter methods use constant space but multiplication which has circuit complexity $\Theta(\log w / \log \log w)$ for w-bit words [11]. Our tabulation-based schemes use only AC^0 operations, but larger space. This is an inherent difference, as 2-independence is only possible with AC^0 operations using a large amount of space [2, 32, 34]. As is evident from Table 1, our experiments show that simple tabulation is 2-3 slower than multiply-shift, but as fast or faster than 2-independent PolyHash. Essentially, this can be ascribed to the cache of the two computers used being comparable in speed to arithmetic instructions. This is not surprising as most computation in the world involves data and hence cache. It is therefore expected that most computers have cache as fast as arithmetic instructions. In fact, since fast multiplication circuits are complex and expensive, and a lot of data processing does not involve multiplication, one could imagine computers with much faster cache than multiplication [28].

Twisted Tabulation. Carrying out a bit more work than simple tabulation, twisted tabulation performs c lookups of entries that are twice the size, as well as executing a few extra AC^0 operations. It hence performs a little worse than simple tabulation hashing.

Mixed Tabulation. We implemented mixed tabulation hashing with the same parameters ($c = d$) as in [17]. With these parameters the scheme uses $2c$ lookups from $2c$ character tables, where c of the lookups are to table entries that are double as long as the output, which may explain its worse performance with 64-bit domains. In our experiments, mixed tabulation performs slightly worse than tabulation-permutation hashing. Recall from above that mixed tabulation is faster than many popular hash functions without theoretical guarantees, hence so is our tabulation-permutation.
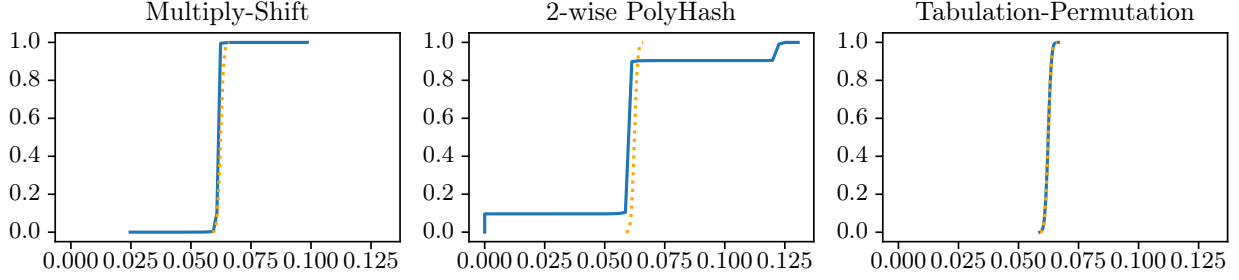
13

Figure 1: Hashing the arithmetic progression $\{a \cdot i \mid i \in [50000]\}$ to 16 bins for a random integer a. The dotted line is a 100-independent PolyHash.

Tabulation-1Permutation. Also only slightly more involved than simple tabulation, tabulation-1permutation performs $c+1$ lookups using $c+1$ character tables. In our experiments, tabulation-1permutation turns out to be a little bit faster than twisted tabulation, at most 30% slower than simple tabulation, and at most 4 times slower than multiply-shift. Recall that tabulation-1permutation is our hash function of choice for streaming applications where speed is critical.

Tabulation-Permutation. Tabulation-permutation hashing performs $2c$ lookups from $2c$ character tables. In our experiments, it is slightly more than twice as slow as simple tabulation, and at most 8 times slower than multiply-shift. It is also worth noting that it performs better than mixed tabulation.

Double Tabulation. Recall that among the schemes discussed so far, only tabulation-permutation and tabulation-1permutation hashing offer unrestricted Chernoff-style concentration with high probability. Double tabulation is the first alternative with similar guarantees and in our experiments it is 30 times slower for 32-bit keys. For 64-bit keys, we were unable to run it on the computers at our disposal due to the large amount of space required for the hash tables. As already discussed, theoretically, double tabulation needs more space and lookups. The 32-bit version performed 26 lookups in tables of total size $48 \cdot 2^{22}$, while tabulation-permutation only needs 8 lookups using $8 \cdot 2^8$ space. It is not surprising that double tabulation lags so far behind.

100-Independent PolyHash. Running the experiment with 100-independent PolyHash, it turned out that for 32-bit keys, it is slower than 100-independent double tabulation. A bit surprisingly, 100-independent PolyHash ran nearly 200 times slower than the 2-independent PolyHash, even though it essentially just runs the same code 99 times. An explanation could be that the 2-independent scheme just keeps two coefficients in registers while the 100-independent scheme would loop through all the coefficients. We remark that the number 100 is somewhat arbitrary. We need $k = \Theta(\log u)$, but we do not know the exact constants in the Chernoff bounds with k-independent hashing. The running times are, however, easily scalable and for k-independent PolyHash, we would expect the evaluation time to change by a factor of roughly $k/100$.

Bad instances for Multiply-Shift and 2-wise PolyHash We finally present experiments demonstrating concrete bad instances for the hash functions Multiply-Shift [18] and 2-wise PolyHash, underscoring what it means for them to not support Chernoff-style concentration bounds. In each case, we compare with our new tabulation-permutation hash function as well as 100-independent PolyHash, which is our approximation to an ideal fully random hash function. We refer the reader to Appendix A for bad instances for simple-tabulation [49] and twisted tabulation [41] as well as a more thorough discussion of our experiments.

Bad instances for Multiply-Shift and 2-independent PolyHash are analyzed in detail in [39, Appendix B]. The specific instance we consider is that of hashing the arithmetic progression $A = \{a \cdot i \mid i \in [50000]\}$ into 16 bins, where we are interested in the number of keys from A that hashes to a specific bin. We performed this

experiment 5000 times, with independently chosen hash functions. The cumulative distribution functions on the number of keys from A hashing to a specific bin is presented in Figure 1. We see that most of the time 2-independent PolyHash and Multiply-Shift distribute the keys perfectly with exactly 1/16 of the keys in the bin. By 2-independence, the variance is the same as with fully random hashing, and this should suggest a much heavier tail, which is indeed what our experiments show. For contrast, we see that the cumulative distribution function with our tabulation-permutation hash function is almost indistinguishable from that of 100-independent Poly-Hash. We note that no amount of experimentation can prove that tabulation-permutation (or any other hash function) works well for all possible inputs. However, given the mathematical concentration guarantee of Theorem 2, the strong performance of tabulation-permutation in the experiment is no surprise.

2 Technical Theorems and how they Combine

We now formally state our main technical results, in their full generality, and show how they combine to yield Theorems 1, 2, and 3. A fair warning should be given to the reader. The theorems to follow are intricate and arguably somewhat inaccessible at first read. Rather than trying to understand everything at once, we suggest that the reader use this section as a roadmap for the main body of the paper. We will, however, do our best to explain the contents of the results as well as disentangling the various assumptions in the theorems.

As noted in Section 1.6, the exposition is subdivided into three parts, each yielding theorems that we believe to be of independent interest. First, we provide an improved analysis of simple tabulation (Section 4). We then show how permuting the output of a simple tabulation hash function yields a hash function having Chernoff bounds for arbitrary value functions (Section 5). Finally, we show that concatenating the output of two independent hash functions preserves the property of having Chernoff bounds for arbitrary value functions (Section 6).

It turns out that the proofs of our results become a little cleaner when we assume that value functions take values in $[-1, 1]$, so from here on we state our results in relation to such value functions. Theorems 1, 2, and 3 will still follow, as the value functions in these theorems can also be viewed as having range $[-1, 1]$.

2.1 Improved Analysis of Simple Tabulation

Our new and improved result on simple tabulation is the subject of Section 4. It is stated as follows.

Theorem 4. *Let $h\colon \Sigma^c \to [m]$ be a simple tabulation hash function and $S \subseteq \Sigma^c$ be a set of keys of size $n = |S|$. Let $v\colon \Sigma^c \times [m] \to [-1, 1]$ be a value function such that the set $Q = \{i \in [m] \mid \exists x \in \Sigma^c : v(x, i) \neq 0\}$ satisfies $|Q| \leq m^\varepsilon$, where $\varepsilon < \frac{1}{4}$ is a constant.*

1. *For any constant $\gamma \geq 1$, the random variable $V = \sum_{x \in S} v(x, h(x))$ is strongly concentrated with added error probability $O_{\gamma,\varepsilon,c}(n/m^\gamma)$, where the constants of the asymptotics are determined by c and γ. Furthermore, this concentration is query invariant.*

2. *For $j \in [m]$ define the random variable $V_j = \sum_{x \in S} v(x, h(x) \oplus j)$ and let $\mu = \mathbb{E}[V_j]$, noting that this is independent of j. For any $\gamma \geq 1$,*

$$\Pr\left[\sum_{j \in [m]} (V_j - \mu)^2 > D_{\gamma,c} \sum_{x \in S} \sum_{k \in [m]} v(x, k)^2\right] = O_{\gamma,\varepsilon,c}(n/m^\gamma) \tag{5}$$

for some constant $D_{\gamma,c}$ and this bound is query invariant up to constant factors.

The technical assumption involving Q states that the value function has *bounded support* in the hash range: The value $v(x, h(x))$ can only possibly be non-zero if $h(x)$ lies in the relatively small set Q of size at most m^ε. In fact, when proving Theorem 1 it suffices to assume that $|Q| = 1$, as we shall see below,

15

but for our analysis of tabulation-permutation hashing we need the more general result above. Another nice illustration of the power of Theorem 4 holding with value functions of *any* bounded support will appear when we prove Theorem 3 in Section 2.4.

To see that Theorem 1 is implied by Theorem 4, one may observe that the latter is a generalization of the former. Let $y \in [m]$ be the bin and $(w_x)_{x \in S}$ be the weights of the balls from S in the setting of Theorem 1. Then defining the value function $v \colon \Sigma^c \times [m] \to [0, 1]$,

$$v(x, y') = \begin{cases} w_x \cdot [y' = y], & x \in S, \\ 0, & x \notin S, \end{cases}$$

we find that $X = \sum_{x \in S} w_x \cdot [h(x) = y] = \sum_{x \in S} v(x, h(x))$ is strongly concentrated by part 1 of Theorem 4 and the concentration is query invariant.

Finally, the bound (5) requires some explaining. For this, we consider the toy example of Theorem 1. Suppose we have a set $S \subseteq [u]$ of balls with weights $(w_x)_{x \in S}$ and we throw them into the bins of $[m]$ using a simple tabulation hash function. We focus on the total weight of balls landing in bin 0, defining the value function by $v(x, y) = w_x$ for $x \in S$ and $y = 0$, and $v(x, y) = 0$ otherwise. In this case, $\mu = \frac{1}{m} \sum_{x \in S} w_x$ denotes the expected total weight in any single bin and $V_j = \sum_{x \in S} w_x \cdot [h(x) = j]$ denotes the total weight in bin $j \in [m]$. Then (5) states that $\sum_{j \in [m]} (V_j - \mu)^2 = O(\|w\|_2^2)$ with high probability in m. This is exactly a bound on the *variance* of the weight of balls landing in one of the bins when each of the hash values of the keys of S are shifted by an XOR with a uniformly random element of $[m]$. Note that this example corresponds to the case where $|Q| = 1$. In its full generality, i.e., for general value functions of bounded support, (5) is similarly a bound on the variance of the value obtained from the keys of S when their hash values are each shifted by a uniformly random XOR. This variance bound turns out to be an important ingredient in our proof of the strong concentration in the first part of Theorem 4. As described in Section 1.6.1 the proof proceeds by fixing the hash values of the position characters $[c] \times \Sigma$ in a carefully chosen order, $\alpha_1 \prec \cdots \prec \alpha_r$. Defining G_i to be those keys that contain α_i as a position character but no α_j with $j > i$, the internal clustering of the keys of G_i is determined solely by $(h(\alpha_j))_{j < i}$ and fixing $h(\alpha_i)$ "shifts" each of these keys by an XOR with $h(\alpha_i)$. Now (5), applied with $S = G_i$, exactly yields a bound on the *variance* of the total value obtained from the keys from G_i when fixing the random XOR $h(\alpha_i)$. Thus, (5) conveniently bounds the variance of the martingale described in Section 1.6.1. As such, (5) is merely a technical tool, but we have a more important reason for including the bound in the theorem. As it turns out, for *any* hash function satisfying the conclusion of Theorem 4, composing with a uniformly random permutation yields a hash family having Chernoff-style concentration bounds for any value function as we describe next.

2.2 Permuting the Hash Range

Our next step in proving Theorem 2 is to show that, given a hash function with concentration bounds like in Theorem 4, composing with a uniformly random permutation of the entire range yields a hash function with Chernoff-style concentration for general value functions. The main theorem, proved in Section 5, is as follows.

Theorem 5. *Let $\varepsilon \in (0, 1]$ and $m \geq 2$ be given. Let $g \colon [u] \to [m]$ be a 3-independent hash function satisfying the following. For every $\gamma > 0$, and for every value function $v \colon [u] \times [m] \to [-1, 1]$ such that the set $Q = \{i \in [m] \mid \exists x \in [u] \colon v(x, i) \neq 0\}$ is of size $|Q| \leq m^\varepsilon$, the two conclusions of Theorem 4 holds with respect to g.*

Let $v' \colon [u] \to [-1, 1]$ be any value function, $\tau \colon [m] \to [m]$ be a uniformly random permutation independent of g, and $\gamma > 0$. Then the for the hash function $h = \tau \circ g$, the sum $\sum_{x \in [u]} v'(x, h(x))$ is strongly concentrated with added error probability $O_{\gamma, \varepsilon}(u/m^\gamma)$, where the constants of the asymptotics are determined solely by ε and γ. Furthermore, this concentration is query invariant.

We believe the theorem to be of independent interest. From a hash function that only performs well for value functions supported on an asymptotically small subset of the bins we can construct a hash function

performing well for any value function – simply by composing with a random permutation. Theorem 4 shows that simple tabulation satisfies the two conditions in the theorem above. It follows that if $m = |U|^{\Omega(1)}$, e.g., if $m = |\Sigma|$, then composing a simple tabulation hash function $g : \Sigma^c \to [m]$ with a uniformly random permutation $\tau : [m] \to [m]$ yields a hash function $h = \tau \circ g$ having Chernoff-style bounds for general value functions asymptotically matching those from the fully random setting up to an added error probability inversely polynomial in the size of the universe. In particular these bounds hold for tabulation-permutation hashing from Σ^c to Σ, that is, using just a single permutation, which yields the result of Theorem 2 in the case $d = 1$. If we desire a range of size $m \gg |\Sigma|$ the permutation τ becomes too expensive to store. Recall that in tabulation-permutation hashing from Σ^c to Σ^d we instead use d permutations $\tau_1, \ldots, \tau_d : \Sigma \to \Sigma$, hashing

$$\Sigma^c \xrightarrow{g^{\text{simple}}} \Sigma^d \xrightarrow{(\tau_1, \ldots, \tau_d)} \Sigma^d.$$

Towards proving that this is sensible, the last step in the proof of Theorem 2 is to show that concatenating the outputs of independent hash functions preserves the property of having Chernoff-style concentration for general value functions.

2.3 Squaring the Hash Range

The third and final step towards proving Theorem 2 is showing that concatenating the hash values of two independent hash functions each with Chernoff-style bounds for general value functions yields a new hash function with similar Chernoff-style bounds up to constant factors. In particular it will follow that tabulation-permutation hashing has Chernoff-style bounds for general value functions. However, as with Theorem 5, the result is of more general interest. Since it uses the input hash functions in a black box manner, it is a general tool towards constructing new hash functions with Chernoff-style bounds. The main theorem, proved in Section 6, is the following.

Theorem 6. *Let $h_1 \colon A \to B_1$ and $h_2 \colon A \to B_2$ be 2-wise independent hash functions with a common domain such that for every pair of value functions, $v_1 \colon A \times B_1 \to [-1, 1]$ and $v_2 \colon A \times B_2 \to [-1, 1]$, the random variables $X_1 = \sum_{a \in A} v_1(a, h_1(a))$ and $X_2 = \sum_{a \in A} v_2(a, h_2(a))$ are strongly concentrated with added error probability f_1 and f_2, respectively, and the concentration is query invariant. Suppose further that h_1 and h_2 are independent. Then the hash function $h = (h_1, h_2) \colon A \to B_1 \times B_2$, which is the concatenation of h_1 and h_2, satisfies that for every value function $v \colon A \times (B_1 \times B_2) \to [-1, 1]$, the random variable $X = \sum_{a \in A} v(a, h(a)) = \sum_{a \in A} v(a, h_1(a), h_2(a))$ is strongly concentrated with additive error $O(f_1 + f_2)$ and the concentration is query invariant.*

We argue that Theorem 6, combined with the previous results, leads to Theorem 2.

Proof of Theorem 2. We proceed by induction on d. For $d = 1$ the result follows from Theorem 4 and 5 as described in the previous subsection. Now suppose $d > 1$ and that the result holds for smaller values of d. Let $\gamma = O(1)$ be given. Let $d_1 = \lfloor d/2 \rfloor$ and $d_2 = \lceil d/2 \rceil$. A tabulation-permutation hash function $h : \Sigma^c \to \Sigma^d$ is the concatenation of two independent tabulation-permutation hash functions $h_1 : \Sigma^c \to \Sigma^{d_1}$ and $h_2 : \Sigma^c \to \Sigma^{d_2}$. Letting $A = \Sigma^c$, $B_1 = \Sigma^{d_1}$, $B_2 = \Sigma^{d_2}$, the induction hypothesis gives that the conditions of Theorem 6 are satisfied and the conclusion follows. Note that since $d = O(1)$, the induction is only applied a constant number of times. Hence, the constants hidden in the asymptotics of Definition 1 are still constant. $\square$

2.4 Concentration in Arbitrary Intervals.

We will now show how we can use our main result, Theorem 2, together with our improved understanding of simple tabulation Theorem 4 to obtain Theorem 3 which shows that the extra efficient tabulation-1permutation hashing provides Chernoff-style concentration for the special case of weighted balls and intervals. This section also serves as an illustration of how our previous results play in tandem, and it illustrates the importance of Theorem 4 holding, not just for single bins, but for any value function of bounded support.

17

Proof of Theorem 3. Let $S \subseteq [u]$ be a set of keys, with each key $x \in S$ having a weight $w_x \in [0, 1]$. Let $h = \tau \circ g \colon \Sigma^c \to \Sigma^d = [r]$ be a tabulation-1permutation hash function, with $g \colon \Sigma^c \to \Sigma^d$ a simple tabulation hash function and $\tau \colon \Sigma^d \to \Sigma^d$ a random permutation of the most significant character, $\tau(z_1, \ldots, z_d) = (\tau_1(z_1), z_2, \ldots, z_d)$ for a uniformly random permutation $\tau_1 \colon \Sigma \to \Sigma$. Let $y_1, y_2 \in \Sigma^d$ and X be defined as in Theorem 3, $X = \sum_{x \in S} w_x \cdot [y_1 \leq h(x) < y_2]$. Set $\mu = \mathbb{E}[X]$, and $\sigma^2 = \text{Var}[X]$. For simplicity we assume that $|I| \geq r/2$. Otherwise, we just apply the argument below with I replaced by $[r] \setminus I = [0, y_1) \cup [y_2, r)$, which we view as an interval in the cyclic ordering of $[r]$. We will partition $I = [y_1, y_2)$ into a constant number of intervals in such a way that our previous results yield Chernoff style concentration bound on the total weight of keys landing within each of these intervals. The desired result will follow.

To be precise, let $t > 0$ and $\gamma = O(1)$ be given. Let $P_1 = \{x \in \Sigma \mid \forall y \in \Sigma^{d-1} : (x, y) \in I\}$ and $I_1 = \{(x_1, \ldots, x_d) \in \Sigma^d \mid x_1 \in P_1\}$. Whether or not $h(x) \in I_1$ for a key $x \in \Sigma^c$ depends solely on the most significant character of $h(x)$. With $X_1 = \sum_{x \in S} w_x \cdot [h(x) \in I_1]$, $\mu_1 = \mathbb{E}[X_1]$, and $\sigma_1^2 = \text{Var}[X_1]$, we can therefore apply Theorem 2 to obtain that for any $t' > 0$ and $\gamma' = O(1)$,

$$\Pr[|X_1 - \mu_1| \geq t'] \leq C \exp(-\Omega(\sigma_1^2 \mathcal{C}(t'/\sigma_1^2))) + 1/u^{\gamma'} \leq C \exp(-\Omega(\sigma^2 \mathcal{C}(t'/\sigma^2))) + 1/u^{\gamma'}, \tag{6}$$

for some constant C. Here we used that $\sigma_1^2 \leq \sigma^2$ as $|I_1| \leq |I| \leq |\Sigma^d|/2$. Next, let $d_1 = \lg |\Sigma|$ and $d_2, \ldots, d_\ell \in \mathbb{N}$ be such that for $2 \leq i \leq \ell$, it holds that $2^{d_i} \leq (2^{d_1 + d_2 + \cdots + d_i})^{1/4}$, and further $2^{d_1 + d_2 + \cdots + d_\ell} = |\Sigma|^d$. We may assume that u and hence $|\Sigma|$ is larger than some constant as otherwise the bound in Theorem 3 is trivial. It is then easy to check that we may choose ℓ and the $(d_i)_{2 \leq i \leq \ell}$ such that $\ell = O(\log d) = O(1)$. We will from now on consider elements of Σ^d as numbers written in binary or, equivalently, bit strings of length $d' := d_1 + \cdots + d_\ell$. For $i = 1, \ldots, \ell$ we define a map $\rho_i \colon \Sigma^d \to [2]^{d_1 + \cdots + d_i}$ as follows. If $x = b_1 \ldots b_{d'} \in [2]^{d'}$, then $\rho_i(x)$ is the length $d_1 + \cdots + d_i$ bit string $b_1 \ldots b_{d_1 + \cdots + d_i}$ Set $J_1 = I$. For $i = 2, \ldots, \ell$ we define $J_i \subseteq I$ and $I_i \subseteq I$ recursively as follows. First, we let $J_i = J_{i-1} \setminus I_{i-1}$. Second, we define I_i to consist of those elements of $x \in J_i$ such that if $y \in \Sigma^c$ has $\rho_i(y) = \rho_i(x)$, then $y \in J_i$. In other words, I_i consists of those elements of J_i that remain in J_i when the least significant $d_{i+1} + \cdots + d_\ell$ bits of x are changed in an arbitrary manner. It is readily checked that for $i = 1, \ldots, \ell$, I_i is a disjoint union of two (potentially empty) intervals $I_i = I_i^{(1)} \cup I_i^{(2)}$ such that for each $j \in \{1, 2\}$ and $x, y \in I_i^{(j)}$, $\rho_i(x) = \rho_i(y)$. Moreover, the sets $(I_i)_{i=1}^\ell$ are pairwise disjoint and $I = \bigcup_{i=1}^\ell I_i$.

We already saw in (6) that we have Chernoff-style concentration for the total weight of balls landing in I_1. We now show that the same is true for $I_i^{(j)}$ for each $i = 2, \ldots, \ell$ and $j \in \{0, 1\}$. So let such an i and j be fixed. Note that whether or not $h(x) \in I_i^{(j)}$, for a key $x \in \Sigma^c$, depends solely on the most significant $d_1 + \cdots + d_i$ bits of $h(x)$. Let $h' \colon \Sigma^c \to [2]^{d_1 + \cdots + d_i}$ be defined by $h'(x) = \rho_i(h(x))$. Then h' is itself a simple tabulation hash function and $h'(x)$ is obtained by removing the $d_{i+1} + \cdots + d_\ell$ least significant bits of $h(x)$. Letting $I' = \rho_i(I_i^{(j)})$, it thus holds that $h(x) \in I_i^{(j)}$ if and only if $h'(x) \in I'$. Let now $X_i^{(j)} = \sum_{x \in S} w_x \cdot [h(x) \in I_i^{(j)}]$, $\mu_i^{(j)} = \mathbb{E}[X_i^{(j)}]$, and $\sigma_1^2 = \text{Var}[X_i^{(j)}] \leq \sigma^2$. As $|I'| \leq 2^{d_i} \leq (2^{d_1 + \cdots + d_i})^{1/4}$, we can apply Theorem 4 to conclude that for $t' > 0$ and $\gamma' = O(1)$,

$$\Pr[|X_i^{(j)} - \mu_i^{(j)}| \geq t'] \leq C \exp(-\Omega(\sigma_1^2 \mathcal{C}(t'/\sigma_1^2))) + 1/u^{\gamma'} \leq C \exp(-\Omega(\sigma^2 \mathcal{C}(t'/\sigma^2))) + 1/u^{\gamma'}. \tag{7}$$

Now applying (6) and (7) with $t' = t/(2\ell - 1)$ and $\gamma' = \gamma + \frac{\log(2\ell)}{\log u} = O(1)$, it follows that

$$\Pr[|X - \mu| \geq t] \leq \Pr[|X_1 - \mu_1| \geq t'] + \sum_{i=2}^{\ell} \sum_{j=1}^{2} \Pr[|X_i^{(j)} - \mu_i^{(j)}| \geq t'] \leq 2C\ell \exp(-\Omega(\sigma^2 \mathcal{C}(t'/\sigma^2))) + 2\ell/u^{\gamma'}$$

$$= O(\exp(-\Omega(\sigma^2\, \mathcal{C}(t/\sigma^2)))) + 1/u^\gamma,$$

as desired. $\qquad\square$

3 Preliminaries

Before proceeding, we establish basic definitions and describe results from the literature which we will use.

3.1 Notation

Throughout the paper, we use the following general notation.

- We let $[n]$ denote the set $\{0, 1, \ldots, n-1\}$.

- For a statement or event Q we let $[Q]$ be the indicator variable on Q, i.e.,

$$[Q] = \begin{cases} 1, & Q \text{ occurred or is true}, \\ 0, & \text{otherwise}. \end{cases}$$

- Whenever $Y_0, \ldots, Y_{n-1} \in \mathbb{R}$ are variables and $i \in [n+1]$, we shall denote by $Y_{<i}$ the sum $\sum_{j<i} Y_j$. Likewise, whenever $A_0, \ldots, A_{n-1}$ are sets and $i \in [n+1]$, we shall denote by $A_{<i}$ the set $\bigcup_{j<i} A_j$.

- Suppose we have a hash function $h \colon A \to B$ with domain A and range B. We shall often associate weight and value functions with h as follows.

  - A function $w \colon A \to \mathbb{R}$ is called a *weight function*, corresponding to the idea that every ball or key $x \in A$ has an associated weight, $w(x) \in \mathbb{R}$. Occasionally, we shall write w_x for $w(x)$.
  - A function $v \colon A \times B \to \mathbb{R}$ is called a *value function*, with the interpretation that a key $x \in A$ yields a value $v(x, h(x))$ depending on the bin/hash value $h(x) \in B$.

  For weight functions $w \colon A \to \mathbb{R}$, a subset of balls, $S \subset A$, and a bin $y_0 \in B$, we will be interested in sums of the form $W = \sum_{x \in S} w(x)[h(x) = y_0]$, i.e., the total weight of the balls in S that are hashed to bin y_0. Defining the value function $v \colon A \times B \to \mathbb{R}$ by $v(x, y) = w(x)[y = y_0]$, W is exactly equal to $\sum_{x \in S} v(x, h(x))$, i.e., the total value obtained by the balls in S. From this perspective, value functions are more general objects than weight functions.

3.2 Probability Theory and Martingales

In the following, we introduce the necessary notions of probability theory. A note of caution is in order. The paper at hand relies on results from the theory of martingales to arrive at its conclusion. Working with martingales, we shall require probability theoretic notions of a fairly general and abstract character. For an introduction to measure and probability theory, see, for instance, [42].

For the most basic notation, let $(\Omega, \mathcal{F}, \Pr)$ be a probability space.

- Let $X_1, \ldots, X_n : \Omega \to \mathbb{R}$ be $\mathcal{F}$-measurable random variables. We denote by $\mathcal{G} = \sigma(X_1, \ldots, X_n) \subset \mathcal{F}$ the smallest σ-algebra such that $X_1, \ldots, X_n$ are all $\mathcal{G}$-measurable. We say that $\mathcal{G}$ is the *sigma algebra generated by* $X_1, \ldots, X_n$. Intuitively, $\sigma(X_1, \ldots, X_n)$ represents the collective information regarding the outcome of the joint distribution $(X_1, \ldots, X_n)$.

- Let $X : \Omega \to \mathbb{R}$ be an $\mathcal{F}$-measurable random variable, and let $\mathcal{G}$ be a σ-algebra with $\mathcal{G} \subset \mathcal{F}$. If $\mathbb{E}[|X|] < \infty$, we may define the random variable $\mathbb{E}[X \mid \mathcal{G}]$ to be the *conditional expectation* of X given $\mathcal{G}$. It is important to note that $\mathbb{E}[X \mid \mathcal{G}]$ is $\mathcal{G}$-measurable. In the context of the above notation, $\mathbb{E}[X \mid \sigma(X_1, \ldots, X_n)] = \mathbb{E}[X \mid X_1, \ldots, X_n]$ is the expectation of X as a function of the outcomes of $X_1, \ldots, X_n$.

We proceed to discuss martingales and martingale differences. For convenience we shall assume all random variables to be bounded, i.e., whenever X is a random variable, we assume that there exists a constant $M \geq 0$ such that $|X| \leq M$ almost surely.

Definition 3 (Filtration). Let $(\Omega, \mathcal{P}(\Omega), \Pr)$ be a finite measure space. A sequence of σ-algebras, $(\mathcal{F}_i)_{i=0}^r$, is a *filtration* of $(\Omega, \mathcal{P}(\Omega), \Pr)$ if $\{\emptyset, \Omega\} = \mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}_r = \mathcal{P}(\Omega)$. We shall usually omit explicit reference to the background space.

Definition 4 (Adapted Sequence). Let $(\mathcal{F}_i)_{i=0}^r$ be a filtration. A sequence of random variables $(X_i)_{i=0}^r$ is *adapted* to $(\mathcal{F}_i)_{i=0}^r$ if for every $i \in [r+1]$, X_i is $\mathcal{F}_i$-measurable. In that case, we say that $(X_i, \mathcal{F}_i)$ is an *adapted sequence*.

Definition 5 (Martingale). A *martingale* is an adapted sequence, $(X_i, \mathcal{F}_i)$, satisfying that for every $i \in \{1, \ldots, r\}$, $\mathbb{E}[X_i \mid \mathcal{F}_{i-1}] = X_{i-1}$.

Definition 6 (Martingale Difference). A *martingale difference* is a an adapted sequence, $(Y_i, \mathcal{F}_i)_{0=1}^r$, such that $Y_0 = 0$ almost surely and for every $i \in \{1, \ldots, r\}$, $\mathbb{E}[Y_i \mid \mathcal{F}_{i-1}] = 0$.

If $(X_i, \mathcal{F}_i)_{i=0}^r$ is a martingale, we may define the sequence of random variables $(Y_i)_{i=0}^r$ by $Y_0 = 0$ and $Y_i = X_i - X_{i-1}$ for $i = 1, \ldots, r$. Then $(Y_i, \mathcal{F}_i)_{i=0}^r$ is a martingale difference. Conversely, if $(Y_i, \mathcal{F}_i)_{i=0}^r$ is a martingale difference, a martingale $(X_i, \mathcal{F}_i)_{i=0}^r$ can be constructed by letting $X_i = Y_{<i+1} = \sum_{j \leq i} Y_j$. Under this correspondence, martingales and martingale differences are in a sense two sides of the same coin.

Concluding the section, we describe canonical constructions of a martingale and a martingale difference, respectively, that we shall use later on.

- Let X be a random variable and consider a filtration $(\mathcal{F}_i)_{i=0}^r$. We may define a martingale from X with respect to $(\mathcal{F}_i)_{i=0}^r$ by defining the sequence of random variables $(X_i)_{i=0}^r$ by $X_i = \mathbb{E}[X \mid \mathcal{F}_i]$ for each $i \in [r+1]$. Clearly, $\mathbb{E}[X_i \mid \mathcal{F}_{i-1}] = \mathbb{E}[X \mid \mathcal{F}_{i-1}] = X_{i-1}$, so $(X_i, \mathcal{F}_i)_{i=0}^r$ is indeed a martingale.

  We shall apply this construction in the following situation. Suppose we have random variables $Z_1, \ldots, Z_r$ taking values in the measure spaces $A_1, \ldots, A_r$ and denote by Z the joint distribution $(Z_1, \ldots, Z_r)$. For some function $f : A_1 \times \ldots A_r \to \mathbb{R}$, we wish to assess the value of $f(Z)$. We may then define the filtration $\mathcal{F}_i = \sigma(Z_1, \ldots, Z_i)$ for $i \in [r+1]$ and set $X_i = \mathbb{E}[f(Z) \mid \mathcal{F}_i]$. This yields a martingale $(X_i, \mathcal{F}_i)_{i=0}^r$ with $X_0 = \mathbb{E}[f(Z)]$ and $X_r = f(Z)$. This is known as a Doob martingale and the construction will be used in Section 5 to prove Theorem 5.

- Let $(Z_i, \mathcal{F}_i)_{i=0}^r$ be an adapted sequence and define $Y_0 = 0$ and $Y_i = Z_i - \mathbb{E}[Z_i \mid \mathcal{F}_{i-1}]$ for $i \in \{1, \ldots, r\}$. Then $(Y_i, \mathcal{F}_i)_{i=1}^r$ is a martingale difference. This construction is applied in Section 4 to prove Theorem 4.

3.3 Martingale Concentration Inequalities

In applications of probability theory, we often consider a sequence of random variables $X_0, \ldots, X_r$. If we are lucky, the random variables are independent, pair-wise independent, or a derivative thereof. It is unfortunately often the case, however, that there is no such independence notion that apply to $X_0, \ldots, X_r$. One reason that martingales have been as successful as they are, is that frequently, one may instead impose a martingale structure on the variables, and martingales satisfy many of the same theorems that independent variables do. In this exposition, we shall consider sums of the form $X = \sum_{i=0}^r X_i$ where the X_i are far from independent, yet we would like X to satisfy Chernoff-style bounds.

To this end, we state a martingale version of Bennett's inequality due to Fan et al [24]. The reader may note the similarity to Eq. (3).

Definition 7. We denote by $\mathcal{C} : (-1, \infty) \to [0, \infty)$ the function given by $\mathcal{C}(x) = (x+1)\ln(x+1) - x$.

Theorem 7 (Fan et al. [24]). *Let $\sigma > 0$ be given. Let $(X_i, \mathcal{F}_i)_{i=0}^r$ be a martingale difference such that almost surely $|X_i| \leq 1$ for all $i \in \{1, \ldots, r\}$ and $\sum_{i=1}^r \mathbb{E}[X_i^2 \mid \mathcal{F}_{i-1}] \leq \sigma^2$. Writing $X = \sum_{i=1}^r X_i$, it holds for any $t \geq 0$ that*

$$\Pr[X \geq t] \leq e^t \cdot \left(\frac{\sigma^2}{\sigma^2 + t} \right)^{\sigma^2 + t}.$$

Simple calculations yield the following corollary.

Corollary 8. *Suppose that $(X_i, \mathcal{F}_i)_{i=0}^r$ is a martingale difference and there exist $M, \sigma \geq 0$ such that $|X_i| \leq M$ for all $i \in \{1, \ldots, r\}$ and $\sum_{i=1}^r \mathbb{E}\left[X_i^2 \mid \mathcal{F}_{i-1}\right] \leq \sigma^2$. Define $X = \sum_{i=1}^r X_i$. For any $t \geq 0$ it holds that*

$$\Pr\left[X \geq t\right] \leq \exp\left(-\frac{\sigma^2}{M^2}\mathcal{C}\left(\frac{tM}{\sigma^2}\right)\right),$$

where $\mathcal{C}(x) = (x+1)\ln(x+1) - x$.

Finally, we present three lemmas describing the asymptotic behavior of $\mathcal{C}$. We omit the proofs of the first two since the results are standard and follow by elementary calculus.

Lemma 9. *For any $x \geq 0$*

$$\frac{1}{2}x\ln(x+1) \leq \mathcal{C}(x) \leq x\ln(x+1) .$$

For any $x \in [0, 1]$

$$\frac{1}{3}x^2 \leq \mathcal{C}(x) \leq \frac{1}{2}x^2 ,$$

where the right hand inequality holds for all $x \geq 0$.

Lemma 10. *For any $a \geq 0$. If $b \geq 1$ then*

$$b\mathcal{C}(a) \leq \mathcal{C}(ab) \leq b^2\mathcal{C}(a) .$$

If $0 \leq b \leq 1$ then

$$b^2\mathcal{C}(a) \leq \mathcal{C}(ab) \leq b\mathcal{C}(a) .$$

Note that as a corollary, if $b = \Theta(1)$ and $a \geq 0$, then $\mathcal{C}(ba) = \Theta(\mathcal{C}(a))$. The final lemma shows that the bound of Corollary 8 only gets worse when σ^2 or M is replaced by some larger number.

Lemma 11. *Let $a \geq 0$ be given. On $\mathbb{R}^+$, the following two functions are decreasing*

$$x \mapsto x\mathcal{C}\left(\frac{a}{x}\right) ,$$

$$x \mapsto \frac{\mathcal{C}(ax)}{x^2} .$$

Proof. Let $0 < x \leq y$ be given. We then observe that the first function is indeed decreasing since by the first bound of Lemma 10, $x\mathcal{C}(a/x) = x\mathcal{C}\left((a/y) \cdot (y/x)\right) \geq y\mathcal{C}(a/y)$. That the second function is decreasing follows from a similar argument. $\qquad\square$

4 Analysis of Simple Tabulation

In this section, we analyze the simple tabulation hashing scheme. The section is divided in three parts. First, there will be an introductory section regarding simple tabulation hashing and associated notation. Second, we shall prove the sum of squares result (Eq. (5)). The final section presents a proof of Theorem 4. In order to make the exposition slightly simpler and more accessible, we postpone the argument that our concentration bounds are query invariant to Section 7.

4.1 Simple Tabulation Basics

Simple tabulation hashing as introduced by Zobrist [49] is defined as follows.

Definition 8 (Simple Tabulation Hashing)**.** Let Σ be an alphabet, $c \geq 1$ an integer, and $m = 2^k, k > 0$, a power of two. A simple tabulation hash function, $h \colon \Sigma^c \to [m]$, is a random variable taking values in the set of functions from Σ^c to $[m]$, chosen with respect to the following distribution. For each $j \in \{1, \ldots, c\}$, let $h_j \colon \Sigma \to [m]$ be a fully random hash function, in other words, a uniformly random function from Σ to $[m]$. We evaluate h on the key $x = (x_1, \ldots, x_c) \in \Sigma^c$ by computing $h(x) = \bigoplus_{j=1}^c h_j(x_j)$, where $\oplus$ denotes bitwise XOR.

Now, towards analyzing simple tabulation hashing, we add the following notation.

Definition 9 (Position Character)**.** Let Σ be an alphabet and $c \geq 1$ an integer. We call an element $\alpha = (a, y) \in \{1, \ldots, c\} \times \Sigma$ a *position character* of Σ^c.

Let $h \colon \Sigma^c \to [m]$ be a simple tabulation hash function. We may consider a key $x = (x_1, \ldots, x_c) \in \Sigma^c$ as a set of c position characters, $\{(1, x_1), \ldots, (c, x_c)\} \subseteq \{1, \ldots, c\} \times \Sigma$. Recall that $h(x) = \bigoplus_{i=1}^c h_i(x_i)$ for uniformly random functions $h_i \colon \Sigma \to [m]$. For a position character $\alpha = (a, y) \in \{1, \ldots, c\} \times \Sigma$, we may overload notation and write $h(\alpha) = h_a(y)$. Extending this, for a set of position characters $A = \{\alpha_1, \ldots, \alpha_n\} \subseteq \{1, \ldots, c\} \times \Sigma^c$, $h(A) = \bigoplus_{i=1}^n h(\alpha_i)$. Note that this agrees with the correspondence between keys of Σ^c and sets of position characters mentioned before, since for $x = (x_1, \ldots, x_c) \in \Sigma^c$, $h(x) = h(\{(1, x_1), \ldots, (c, x_c)\})$. If finally $A, B \subset \{1, \ldots, c\} \times \Sigma$ are sets of position characters we write $A \oplus B$ for the symmetric difference between A and B, i.e., $A \oplus B = (A \setminus B) \cup (B \setminus A)$. We note that for a simple tabulation hash function h, $h(A \oplus B) = h(A) \oplus h(B)$.

Definition 10 (Projection Onto an Index)**.** Let $c \geq 1$ be an integer and $i \in \{1, \ldots, c\}$ be given. We denote by $\pi_i \colon \Sigma^c \to \{1, \ldots, c\} \times \Sigma$ the projection onto the ith coordinate given by $\pi_i(x_1, \ldots, x_c) = (i, x_i)$, i.e., projecting a key x to its ith position character. We extend this to sets of keys, such that for $S \subseteq \Sigma^c$, $\pi_i(S) = \{\pi_i(x) \mid x \in S\}$.

The following lemma by Thorup and Zhang [47] describes the independence of sets of position characters of Σ^c under a simple tabulation function $h \colon \Sigma^c \to [2^r]$. We provide a proof for completeness.

Lemma 12 (Thorup and Zhang [47])**.** *Let $h \colon \Sigma^c \to [2^r]$ be a simple tabulation hash function. For each $i \in \{1, \ldots, t\}$, let $s_i \subseteq \{1, \ldots, c\} \times \Sigma$ be a set of position characters of Σ^c. Let $j \in \{1, \ldots, t\}$. If every subset of indices $B \subseteq \{1, \ldots, t\}$ containing j satisfies $\bigoplus_{i \in B} s_i \neq \emptyset$, then the distribution of $h(s_j)$ is independent of the joint distribution $(h(s_i))_{i \neq j}$.*

Proof. Let $\mathbb{F}_2$ be the field $\mathbb{Z}/2\mathbb{Z}$ and V the $\mathbb{F}_2$-vector space $\mathbb{F}_2^{\{1, \ldots, c\} \times \Sigma}$. For a set of position characters A, we define $v_A \in V$ as follows: For $(a, y) \in \{1, \ldots, c\} \times \Sigma$ we let $v_A(a, y) = 1$ if and only if $(a, y) \in A$, and $v_A(a, y) = 0$ otherwise. Picking a random simple tabulation hash function $h : \Sigma^c \to [2^r]$ is equivalent to picking a random *linear* function $h' : V \to [2^r]$. Here $[2^r]$ is identified with the $\mathbb{F}_2$-vector space $\mathbb{F}_2^r$. Indeed, $(v_{\{\alpha\}})_{\alpha \in \{1, \ldots, c\} \times \Sigma}$ forms a basis for V, and choosing a random linear map $h' : V \to [2^r]$ can be done by picking independent and uniformly random values for h' on the basis elements, and extending by linearity. To define h from h', we simply put $h(x) = \bigoplus_{\alpha \in x} h'(v_{\{\alpha\}})$ for a key $x \in \Sigma$ viewed as a set of position characters. Conversely, a simple tabulation hash function $h : \Sigma^c \to [2^r]$ uniquely extends to a linear map $h' : V \to [2^r]$. Now under this identification, the condition in the lemma is equivalent to v_{s_j} being linearly independent of the vectors $(v_{s_i})_{i \neq j}$. As h' is a random linear map, it follows by elementary linear algebra that $h'(v_{s_j}) = h(s_j)$ is independent of the joint distribution $(h'(v_{s_i}))_{i \neq j} = (h(s_i))_{i \neq j}$, as desired. $\square$

4.2   Bounding the Sum of Squared Deviations

In the following section we shall prove the bound (5) of Theorem 4 from Section 2.1, stated independently here as Theorem 16. It is a technical, albeit crucial, step on the way to proving Theorem 4 itself. The foundation of the proof of Theorem 16 is a series of combinatorial observations regarding simple tabulation hashing.

Recall from Section 1.6.1 our general proof strategy when proving concentration bounds for simple tabulation hashing. For a set of keys $S \subseteq \Sigma^c$ to be hashed, we fix an ordering of the position characters of Σ^c. We then fix the hash table entries corresponding to the position characters one at a time according to this ordering. Crucial to the success of this strategy is fixing an ordering where each position character "decides" only a small part of the final outcome.

Definition 11 (Group of Keys). Let $S \subseteq \Sigma^c$ be a set of keys and $A = \{\alpha \in x \mid x \in S\}$ be the set of position characters of the keys of S. For an enumeration or ordering of the position characters of A as $\{\alpha_1, \ldots, \alpha_r\} = A$, we denote by $G_i \subseteq S$ the ith *group of keys* with respect to S and the ordering of the position characters. The set is given by $G_i = \{x \in S \mid \{\alpha_i\} \subseteq x \subseteq \{\alpha_1, \ldots, \alpha_i\}\}$.

Put in other words, let $\prec$ denote the ordering on A, let x be a key of S, and let $\beta_1, \ldots, \beta_c$ be the position characters of x such that $\beta_1 \prec \beta_2 \prec \cdots \prec \beta_c$, i.e., β_c is last in the ordering of A. Then $x \in G_i$ if and only if $\alpha_i = \beta_c$. In relation to simple tabulation, this has the following meaning. In the proof, we shall fix the values $h(\alpha_j)$ one at a time starting at $j = 1$ and ending at $j = r$. For every $x \in G_i$, the value of $h(x)$ is then undecided before $h(\alpha_i)$ is known, but is known once $h(\alpha_1), \ldots, h(\alpha_i)$ are all fixed. In analyzing the contribution of each group to the final outcome of the process, we start by proving a generalization of a result from [38]. It says that if we assign each key a weight, it is always possible to choose the ordering of the position characters such that the total weight of each group is relatively small. The original lemma simply assigned weight 1 to every key.

Lemma 13. *Let $S \subseteq \Sigma^c$ be given and let $A = \{\alpha \in x \mid x \in S\}$ be the position characters of the keys of S. Let $w \colon \Sigma^c \to \mathbb{R}_{\geq 0}$ be a weight function. Then there exists an ordering of the position characters, $\{\alpha_1, \ldots, \alpha_r\} = A$ such that for every $i \in \{1, \ldots, r\}$, the group $G_i = \{x \in S \mid \{\alpha_i\} \subseteq x \subseteq \{\alpha_1, \ldots \alpha_i\}\}$ satisfies*

$$\sum_{x \in G_i} w(x) \leq \left(\max_{x \in S} w(x)\right)^{1/c} \left(\sum_{x \in S} w(x)\right)^{1-1/c} .$$

Proof. We define the ordering recursively and backwards as $\alpha_r, \ldots, \alpha_1$. Let $T_i = A \setminus \{\alpha_{i+1}, \ldots, \alpha_r\}$ and $S_i = \{x \in S \mid x \subseteq T_i\}$. We prove that we can find an $\alpha_i \in T_i$ such that

$$G_i = \{x \in S_i \mid \alpha_i \in x\} ,$$

satisfies

$$\sum_{x \in G_i} w(x) \leq \left(\max_{x \in S_i} w(x)\right)^{1/c} \left(\sum_{x \in S_i} w(x)\right)^{1-1/c} ,$$

which will establish the claim. Let B_k be the set of position characters at position k contained in T_i, i.e., $B_k = \{(k, y) \in T_i\} = \pi_k(T_i)$. Then as $\prod_{k=1}^c |B_k| \geq |S_i|$, we have $|B_k| \geq |S_i|^{1/c}$ for some k.

Since each key of S_i contains at most one position character from B_k, we can choose α_i such that

$$\sum_{x \in G_i} w(x) \leq \frac{\sum_{x \in S_i} w(x)}{|B_k|} \leq \frac{\sum_{x \in S_i} w(x)}{|S_i|^{1/c}} \leq \left(\max_{x \in S_i} w(x)\right)^{1/c} \left(\sum_{x \in S_i} w(x)\right)^{1-1/c} .$$

$\square$

Suppose we have keys $x_1, \ldots, x_t \in \Sigma^c$. It follows as a corollary of Lemma 12 that with a simple tabulation hash function $h \colon \Sigma^c \to [m]$, the values $h(x_1), \ldots, h(x_t)$ are completely independent if and only if there does not exist a subset of indices $B \subseteq \{1, \ldots, t\}$ with $\bigoplus_{i \in B} x_i = \emptyset$. In this vein, it turns out to be natural, given sets of keys $A_1, \ldots, A_\ell \subseteq \Sigma^c$, to bound the number of tuples $x_1 \in A_1, \ldots, x_\ell \in A_\ell$ with $\bigoplus_{i=1}^\ell x_i = \emptyset$. This is the content of Lemma 3 of [16]. We prove the following generalization of this result, which deals with weighted keys. Note that the statements would be identical if each key was assigned the weight 1.

Lemma 14. *Let $\ell \in \mathbb{N}$ be even, $w_1, \ldots, w_\ell \colon \Sigma^c \to \mathbb{R}$ be weight functions, and $A_1, \ldots, A_\ell \subseteq \Sigma^c$ be sets of keys. Then*

$$\sum_{\substack{x_1 \in A_1, \ldots, x_\ell \in A_\ell \\ \bigoplus_{k=1}^{\ell} x_k = \emptyset}} \prod_{k=1}^{\ell} w_k(x_k) \le ((\ell-1)!!)^c \cdot \prod_{k=1}^{\ell} \sqrt{\sum_{x \in A_k} w_k(x)^2}.$$

Proof. For every $(x_1, \ldots, x_\ell) \in A_1 \times \cdots \times A_\ell$ satisfying $\bigoplus_{k=1}^{\ell} x_k = \emptyset$ we have $\bigoplus_{k=1}^{\ell} \{\pi(x_k, c)\} = \emptyset$. This implies that each character in the c-th position occurs an even number of times in $(x_1, \ldots, x_\ell)$. Thus, for any such tuple we can partition the indices $1, \ldots, \ell$ into pairs $(i_1, j_1), \ldots, (i_{\ell/2}, j_{\ell/2})$ satisfying $\pi(x_{i_k}, c) = \pi(x_{j_k}, c)$ for every $k \in \{1, \ldots, \ell\}$. Fix such a partition and let $X \subseteq A_1 \times \cdots \times A_\ell$ be the set

$$X = \{(x_1, \ldots, x_\ell) \in A_1 \times \ldots \times A_\ell \mid \forall k \in \{1, \ldots, \ell/2\} \colon \pi(x_{i_k}, c) = \pi(x_{j_k}, c)\}.$$

We proceed by induction on c.

For $c = 1$, $\pi(x_{i_k}, c) = \pi(x_{j_k}, c)$ implies $x_{i_k} = x_{j_k}$ such that

$$X = \{(x_1, \ldots, x_\ell) \in A_1 \times \ldots \times A_\ell \mid \forall k \in \{1, \ldots, \ell/2\} \colon x_{i_k} = x_{j_k}\}.$$

Thus, by the Cauchy-Schwartz inequality,

$$\sum_{(x_1, \ldots, x_\ell) \in X} \prod_{k=1}^{\ell} w_k(x_k) = \prod_{k=1}^{\ell/2} \sum_{x \in A_{i_k} \cap A_{j_k}} w_{i_k}(x) w_{j_k}(x)$$

$$\le \prod_{k=1}^{\ell/2} \left(\sqrt{\sum_{x \in A_{i_k}} w_{i_k}(x)^2} \cdot \sqrt{\sum_{x \in A_{j_k}} w_{j_k}(x)^2} \right)$$

$$\le \prod_{k=1}^{\ell} \sqrt{\sum_{x \in A_k} w_k(x)^2}.$$

Since this is true for any partition into pairs, $(i_1, j_1), \ldots, (i_{\ell/2}, j_{\ell/2})$, there are exactly $(\ell-1)!!$ such partitions, and every term in the original sum is counted by some partition, we get the desired bound for $c = 1$.

Let $c > 1$ and assume that the statement holds when each key has $< c$ characters. For each $a \in \Sigma$ and $k \in \{1, \ldots, \ell\}$ define the set

$$A_k[a] = \{x \in A_k \mid \pi(x, c) = a\}.$$

Fixing the last character of each pair in our partition by picking $a_1, \ldots, a_{\ell/2} \in \Sigma$ and considering the sets $A_{i_k}[a_k]$ and $A_{j_k}[a_k]$, we can consider the keys of $\prod_{k=1}^{\ell/2} A_{i_k}[a_k] \times A_{j_k}[a_k]$ as only having $c-1$ characters,

which allows us to apply the induction hypothesis. This yields

$$
\sum_{\substack{(x_1,\ldots,x_\ell)\in X \\ \bigoplus_{k=1}^{\ell} x_k = \emptyset}} \prod_{k=1}^{\ell} w_k(x_k) = \sum_{(a_k)_{k=1}^{\ell/2}\in\Sigma^{\ell/2}} \left(\sum_{\substack{(x_{i_k},x_{j_k})_{k=1}^{\ell/2}\in\prod_{k=1}^{\ell/2} A_{i_k}[a_k]\times A_{j_k}[a_k] \\ \bigoplus_{k=1}^{\ell} x_k = \emptyset}} \prod_{k=1}^{\ell/2} w_{i_k}(x_{i_k})w_{j_k}(x_{j_k}) \right)
$$

$$
\leq \sum_{(a_k)_{k=1}^{\ell/2}\in\Sigma^{\ell/2}} \left(((\ell-1)!!)^{c-1}\cdot\prod_{k=1}^{\ell/2}\left(\sqrt{\sum_{x\in A_{i_k}[a_k]} w_{i_k}(x)^2}\cdot\sqrt{\sum_{x\in A_{j_k}[a_k]} w_{j_k}(x)^2} \right) \right)
$$

$$
= ((\ell-1)!!)^{c-1}\cdot\prod_{k=1}^{\ell/2}\left(\sum_{a\in\Sigma}\left(\sqrt{\sum_{x\in A_{i_k}[a]} w_{i_k}(x)^2}\cdot\sqrt{\sum_{x\in A_{j_k}[a]} w_{j_k}(x)^2} \right) \right)
$$

$$
\leq ((\ell-1)!!)^{c-1}\cdot\prod_{k=1}^{\ell/2}\left(\sqrt{\sum_{a\in\Sigma}\sum_{x\in A_{i_k}[a]} w_{i_k}(x)^2}\cdot\sqrt{\sum_{a\in\Sigma}\sum_{x\in A_{j_k}[a]} w_{j_k}(x)^2} \right)
$$

$$
= ((\ell-1)!!)^{c-1}\cdot\prod_{k=1}^{\ell/2}\left(\sqrt{\sum_{x\in A_{i_k}} w_{i_k}(x)^2}\cdot\sqrt{\sum_{x\in A_{j_k}} w_{j_k}(x)^2} \right),
$$

where the last inequality follows from the Cauchy-Schwartz inequality. Since the indices can be partitioned into pairs in $(\ell-1)!!$ ways, the same argument as in the induction start yields

$$
\sum_{\substack{x_1\in A_1,\ldots,x_\ell\in A_\ell \\ \bigoplus_{k=1}^{\ell} x_k = \emptyset}} \prod_{k=1}^{\ell} w_k(x_k) \leq ((\ell-1)!!)^c\cdot\prod_{k=1}^{\ell}\sqrt{\sum_{x\in A_k} w_k(x)^2},
$$

which was the desired conclusion. $\qquad\square$

The following rather technical lemma bounds the moments of collisions between sets of keys. However, we shall dwell on it for a moment as it reflects considerations that will come up repeatedly going forward. Consider a simple tabulation function $h\colon \Sigma^c \to [m]$ and a value function $v\colon \Sigma^c\times[m]\to\mathbb{R}$. Hashing the keys of some subsets $A_1,\ldots,A_n\subseteq\Sigma^c$ into $[m]$ using h, we are interested in the sums $X_i = \sum_{x\in A_i} v(x,h(x))$ for $1\leq i\leq n$ and, in particular, in properties of the joint distribution $(X_1,\ldots,X_n)$. Here, the actual values of X_i are not as important as how much X_i deviates form its mean. For $1\leq i\leq n$, We thus consider the variables

$$
Y_i = X_i - \mathbb{E}\left[X_i\right] = \sum_{x\in A_i}\sum_{b\in[m]} v(x,b)\left([h(x)=b] - \frac{1}{m} \right),
$$

and for a level of generality required for proving the main theorems of this section, we consider the *shifted* variables

$$
Y_i^{(j)} = \sum_{x\in A_i}\sum_{b\in[m]} v(x,b)\left([h(x)=j\oplus b] - \frac{1}{m} \right),
$$

for $j\in[m]$, corresponding to shifting the hash function h by $j\in[m]$.

Lemma 15. *Let $h\colon\Sigma^c\to[m]$ be a simple tabulation hash function and $v\colon\Sigma^c\times[m]\to\mathbb{R}$ a value function. Let $Q = \{i\in[m] \mid \exists x\in\Sigma^c : v(x,i)\neq 0\}$ be the support of v and write $\ell = |Q|$. Let $n\in\mathbb{N}$ and $A_1,\ldots,A_n\subseteq\Sigma^c$. For every $i\in\{1,\ldots,n\}$ and $j\in[m]$ define the random variable*

$$
Y_i^{(j)} = \sum_{x\in A_i}\sum_{b\in Q} v(x,b)\left([h(x)=j\oplus b] - \frac{1}{m} \right),
$$

and set

$$T = \sum_{\substack{j_1,\dots,j_n\in[m] \\ \bigoplus_{j=1}^n j_k=0}} \prod_{k=1}^n Y_k^{(j_k)}.$$

Then for every constant $t \in \mathbb{N}$,

$$\left|\mathbb{E}\left[T^t\right]\right| = O_{t,n,c}\left(\ell^{tn/2} \prod_{k=1}^n \left(\sum_{x\in A_k}\sum_{b\in Q} v(x,b)^2\right)^{t/2}\right).$$

Proof. We rewrite T as follows

$$T = \sum_{\substack{j_1,\dots,j_n\in[m] \\ \bigoplus_{k=1}^n j_k=0}} \prod_{k=1}^n Y_k^{(j_k)}$$

$$= \sum_{\substack{j_1,\dots,j_n\in[m] \\ \bigoplus_{k=1}^n j_k=0}} \left(\prod_{k=1}^n \sum_{x\in A_k}\sum_{b\in Q} v(x,b)\left([h(x)=j_k\oplus b]-\frac{1}{m}\right)\right)$$

$$= \sum_{(x_1,\dots,x_n)\in A_1\times\dots\times A_n}\sum_{b_1,\dots b_n\in Q}\left(\sum_{\substack{j_1,\dots,j_n\in[m] \\ \bigoplus_{k=1}^n j_k=0}}\left(\prod_{k=1}^n\left(v(x_k,b_k)\left([h(x_k)=j_k\oplus b_k]-\frac{1}{m}\right)\right)\right)\right)$$

$$= \sum_{(x_1,\dots,x_n)\in A_1\times\dots\times A_n}\sum_{b_1,\dots b_n\in Q}\left(\left(\prod_{k=1}^n v(x_k,b_k)\right)\cdot\left(\sum_{\substack{j_1,\dots,j_n\in[m] \\ \bigoplus_{k=1}^n j_k=0}}\left(\prod_{k=1}^n\left([h(x_k)=j_k\oplus b_k]-\frac{1}{m}\right)\right)\right)\right)$$

$$= \sum_{(x_1,\dots,x_n)\in A_1\times\dots\times A_n}\sum_{b_1,\dots b_n\in Q}\left(\left(\prod_{k=1}^n v(x_k,b_k)\right)\cdot\left(\left[\bigoplus_{k=1}^n h(x_k)=\bigoplus_{k=1}^n b_k\right]-\frac{1}{m}\right)\right)$$

Here the last equality is derived by observing that for fixed $(x_1,\dots,x_n)\in A_n\times\dots\times A_n$ and fixed $b_1,\dots b_n\in Q$,

$$\sum_{\substack{j_1,\dots,j_n\in[m] \\ \bigoplus_{k=1}^n j_k=0}}\left(\prod_{k=1}^n\left([h(x_k)=j_k\oplus b_k]-\frac{1}{m}\right)\right) = \sum_{\substack{j_1,\dots,j_n\in[m] \\ \bigoplus_{k=1}^n j_k=0}}\left(\sum_{B\subseteq\{1,\dots,n\}}(-m)^{-(n-|B|)}\prod_{k\in B}[h(x_k)=j_k\oplus b_k]\right)$$

and since for $\emptyset\subseteq B\subsetneq\{1,\dots,n\}$ there are exactly $m^{n-|B|-1}$ tuples $(j_1,\dots,j_n)\in[m]^n$ satisfying $j_k\oplus b_k = h(x_k)$ for every $k\in B$ and $\bigoplus_{k=1}^n j_k=0$, we get

$$\sum_{\substack{j_1,\dots,j_n\in[m] \\ \bigoplus_{k=1}^n j_k=0}}\left(\prod_{k=1}^n\left([h(x_k)=j_k\oplus b_k]-\frac{1}{m}\right)\right) = \sum_{\substack{j_1,\dots,j_n\in[m] \\ \bigoplus_{k=1}^n j_k=0}}\left(\prod_{k=1}^n[h(x_k)=j_k\oplus b_k]\right) + \frac{1}{m}\sum_{B\subsetneq\{1,\dots,n\}}(-1)^{n-|B|}$$

$$= \left[\bigoplus_{k=1}^n h(x_k)=\bigoplus_{k=1}^n b_k\right] + \frac{1}{m}\sum_{B\subsetneq\{1,\dots,n\}}(-1)^{n-|B|}.$$

By the principle of inclusion-exclusion, the last term is $-\frac{1}{m}$, which concludes the rearrangement.

Write $S = A_1 \times \ldots \times A_n$ and let $f \colon S \to \mathbb{R}$ be the function

$$f(x_1, \ldots, x_n) = \sum_{b_1, \ldots b_n \in Q} \left(\left(\prod_{k=1}^{n} v(x_k, b_k) \right) \cdot \left(\left[\bigoplus_{k=1}^{n} h(x_k) = \bigoplus_{k=1}^{n} b_k \right] - \frac{1}{m} \right) \right) .$$

By the above rearrangement, we have $T^t = \sum_{(s_i)_{i \in [t]} \in S^t} \prod_{i=1 \in [t]} f(s_i)$, such that,

$$\mathbb{E}\left[T^t \right] = \sum_{(s_i)_{i=1}^{t} \in S^t} \mathbb{E}\left[\prod_{i=1}^{t} f(s_i) \right] .$$

Now, for a t-tuple $(s_i)_{i=1}^{t} \in S^t$, we overload notation by for a subset $T \subseteq \{1, \ldots, t\}$ defining $\bigoplus_{i \in T} s_i = \bigoplus_{i \in T} \bigoplus_{j=1}^{n} (s_i)_j$, where we still think of the keys $(s_i)_j$ as sets of input characters, and where $\oplus$ is the symmetric difference. Let $(s_i)_{i=1}^{t} \in S^t$ and let $T_1, \ldots, T_r \subseteq \{1, \ldots, t\}$ be all subsets of indices satisfying $\bigoplus_{i \in T_j} s_i = \emptyset, 1 \leq j \leq t$. If for some $i \in \{1, \ldots, t\}$, $i \notin \bigcup_{j=1}^{r} T_i$ then by Lemma 12, $h(s_i)$ is independent of the joint distribution $(h(s_j))_{j \neq i}$ and uniformly distributed in $[m]$. It follows that $f(s_i)$ is independent of the joint distribution $(f(s_j))_{j \neq i}$. Since it further holds that $\mathbb{E}[f(s_i)] = 0$, this implies

$$\mathbb{E}\left[\prod_{j=1}^{t} f(s_j) \right] = \mathbb{E}[f(s_i)] \cdot \mathbb{E}\left[\prod_{j \neq i} f(s_j) \right] = 0 .$$

Hence, we shall only sum over the t-tuples $(s_i)_{i=1}^{t} \in S^t$ satisfying that there exist subsets of indices $T_1, \ldots, T_r \subseteq \{1, \ldots, t\}$ such that $\bigoplus_{i \in T_j} s_i = \emptyset$ for every $j \in \{1, \ldots, r\}$ and $\bigcup_{j=1}^{r} T_j = \{1, \ldots, t\}$.

Fix such subsets $T_1, \ldots, T_r \subseteq \{1, \ldots, t\}$ and for $i \in \{1, \ldots, r\}$ let $B_i = T_i \setminus \left(\bigcup_{j < i} T_j \right)$. Then we can write

$$\sum_{\substack{(s_i)_{i=1}^{t} \in S^t \\ \forall j \in \{1, \ldots, r\} \colon \ \bigoplus_{i \in T_j} s_i = \emptyset}} \prod_{i=1}^{t} f(s_i)$$

$$= \sum_{\substack{(s_i)_{i \in \{1, \ldots, t\} \setminus B_r} \in S^{t - |B_r|} \\ \forall j \in \{1, \ldots, r-1\} \colon \ \bigoplus_{i \in T_j} s_i = \emptyset}} \prod_{i \in \{1, \ldots, t\} \setminus B_r} f(s_i) \sum_{\substack{(s_i)_{i \in B_r} \in S^{|B_r|} \\ \bigoplus_{i \in B_r} s_i = \bigoplus_{i \in T_r \setminus B_r} s_i}} \prod_{i \in B_r} f(s_i) \tag{8}$$

Now fix $(s_i)_{i \in \{1, \ldots, t\} \setminus B_r} \in S^{t - |B_r|}$ such that for all $j \in \{1, \ldots, r-1\}$ it holds that $\bigoplus_{i \in T_j} s_i = \emptyset$. We wish to upper bound the inner sum in (8) for this choice of $(s_i)_{i \in \{1, \ldots, t\} \setminus B_r}$. In order to do this, observe that for $s = (x_1, \ldots, x_n) \in S$ we always have

$$|f(s)| \leq \sum_{b_1, \ldots b_n \in Q} \prod_{k \in [n]} |v(x_k, b_k)| = \prod_{k=1}^{n} \left| \sum_{b \in Q} v(x_k, b) \right| \leq \prod_{k=1}^{n} \sqrt{ \ell \sum_{b \in Q} v(x_k, b)^2 } ,$$

by the QA-inequality. We now wish to combine this bound with Lemma 14 to obtain a bound on the inner sum in (8). For this, we define let $\ell = |T_r| n$ and define sets of keys $F_1, \ldots, F_\ell$ and weight functions $w_1, \ldots, w_\ell \colon \Sigma^c \to \mathbb{R}$ as follows. Enumerate $T_r = \{i_1, \ldots, i_{|T_r|}\}$ such that $\{i_1, \ldots, i_{|B_r|}\} = B_r$. Now for $0 \leq k < |B_r|$ and $1 \leq j \leq n$ we define $F_{kn+j} = A_j$. We further define the weight function $w_{kn+j} \colon \Sigma^c \to \mathbb{R}$ by $w_{kn+j}(x) = \sqrt{\ell \sum_{b \in Q} v(x, b)^2}$ for $x \in \Sigma^c$. Observe that these weight functions are all identical. Secondly, for $|B_r| \leq k < |T_r|$ and $1 \leq j \leq n$, we define $F_{kn+j} = \{s_{i_k}(j)\}$, and $w_{kn+j} \colon \Sigma^c \to \mathbb{R}$ by $w_{kn+j}(x) = 1$ for all $x \in \Sigma^c$. Then,

$$\left| \sum_{\substack{(s_i)_{i \in B_r} \in S^{|B_r|} \\ \bigoplus_{i \in B_r} s_i = \bigoplus_{i \in T_r \setminus B_r} s_i}} \prod_{i \in B_r} f(s_i) \right| \leq \sum_{\substack{x_1 \in B_1, \ldots, x_\ell \in B_\ell \\ \bigoplus_{k=1}^{\ell} x_k = \emptyset}} \prod_{k=1}^{\ell} w_k(x_k) \leq (n|T_r|-1)!!)^c \prod_{k=1}^{n} \left(\ell \cdot \sum_{x \in A_k} \sum_{b \in Q} v(x, b)^2 \right)^{|B_r|/2} ,$$

27

where the last inequality follows from Lemma 14. Note that this upper bound does not depend on the choice of $(s_i)_{i \in \{1,\dots,t\} \setminus B_r} \in S^{t-|B_r|}$ in the outer sum in (8). Repeating this argument another $r-1$ times, and using that $\{1,\dots,t\}$ is the disjoint union of $B_1,\dots,B_r$, we obtain that

$$\left| \sum_{\substack{(s_i)_{i=1}^t \in S^t \\ \forall j \in \{1,\dots,r\}: \, \bigoplus_{i \in T_j} s_i = \emptyset}} \prod_{i=1}^t f(s_i) \right| \leq ((nt-1)!!)^{cr} \cdot \prod_{k=1}^n \left(\ell \sum_{x \in A_k} \sum_{b \in Q} v(x,b)^2 \right)^{t/2}.$$

Since there are at most 2^{2^t} ways of choosing r and the subsets $T_1,\dots,T_r$ and since $r \leq 2^t$, summing over these choices yields

$$\left| \mathbb{E}\left[T^t\right] \right| \leq 2^{2^t} ((nt-1)!!)^{cr} \cdot \prod_{k=1}^n \left(\ell \sum_{x \in A_k} \sum_{b \in Q} v(x,b)^2 \right)^{t/2}$$

$$\leq O_{t,n,c} \left(\ell^{nt/2} \prod_{k=1}^n \left(\sum_{x \in A_k} \sum_{b \in Q} v(x,b)^2 \right)^{t/2} \right).$$

$\square$

We are now ready to prove the main theorem of the subsection, a bound on the sum of squared deviations of the value function from its deviation when shifting by every $j \in [m]$, the second part of Theorem 4. As described in Section 2.1, this bound is an important ingredient in the proof of the first part of Theorem 4. Namely, in our inductive proof, it bounds the variance of the value obtained from the keys of one of the groups G_i when the keys from this group are shifted by a uniformly random XOR with $h(\alpha_i)$.

Theorem 16. *Let $h \colon \Sigma^c \to [m]$ be a simple tabulation hash function and $S \subseteq \Sigma^c$ a set of keys. Let $v \colon \Sigma^c \times [m] \to [-1,1]$ be a value function such that the set $Q = \{i \in [m] \mid \exists x \in \Sigma^c : v(x,i) \neq 0\}$ satisfies $|Q| \leq m^\varepsilon$, where $\varepsilon < \frac{1}{4}$ is a constant. For $j \in [m]$ define the random variable $V_j = \sum_{x \in S} v(x, h(x) \oplus j)$ and let $\mu = \mathbb{E}[V_j]$, noting that this is independent of j. For any $\gamma \geq 1$,*

$$\Pr\left[\sum_{j \in [m]} (V_j - \mu)^2 > C_\gamma^c \sum_{x \in S} \sum_{k \in [m]} v(x,k)^2 \right] = O_{\gamma,\varepsilon,c}(n/m^\gamma) \tag{9}$$

where $C_\gamma = 3 \cdot 2^6 \cdot \gamma^2$ and this bound is query invariant up to constant factors.

Proof. First, note that we may write

$$V_j - \mu = \sum_{x \in S} \sum_{k \in Q} v(x,k)[h(x) = j \oplus k] - \frac{1}{m} \sum_{x \in S} \sum_{k \in Q} v(x,k) = \sum_{x \in S} \sum_{k \in Q} v(x,k)\left([h(x) = j \oplus k] - \frac{1}{m}\right) \tag{10}$$

Now, define $v'(x) = \sum_{k \in Q} v(x,k)^2$ and for $X \subseteq \Sigma^c$ we let $v'(X) = \sum_{x \in X} v'(x)$ and define $v'_\infty(X) = \max_{x \in X} v'(x)$. Now applying Lemma 13 with respect to v' we get position characters $\alpha_1,\dots,\alpha_r$ with corresponding groups $G_1,\dots,G_r$, such that, $\uplus_{i=1}^r G_i = S$ and for every $i \in \{1,\dots,r\}$, $v'(G_i) \leq v'(S)^{1-1/c} v'_\infty(S)^{1/c}$. For $i \in \{1,\dots,r\}, j \in [m]$ we define the random variables

$$X_i^{(j)} = \sum_{x \in G_i} \sum_{k \in Q} v(x,k)\left([h(x \setminus \alpha_i) = j \oplus k] - \frac{1}{m}\right), \qquad Y_i^{(j)} = X_i^{(j \oplus h(\alpha_i))},$$

28

where we recall that $x \setminus \alpha_i$ denotes the set containing the position characters of x except α_i. Notice that by (10), $V_j - \mu = \sum_{i \in [r]} Y_i^{(j)}$. Writing $V = \sum_{j \in [m]} (V_j - \mu)^2 = \sum_{j \in [m]} \left(\sum_{i \in [r]} Y_i^{(j)} \right)^2$, the statement we wish to prove is

$$\Pr\left[V > C_\gamma^c v'(S) \right] \le O_{\gamma,c}\left(|S| \, m^{-\gamma} \right) .$$

We proceed by induction on c. The induction start, $c = 1$, and the induction step are almost identical, so we carry them out in parallel. Note that when $c = 1$ each group has size at most 1, i.e. $|G_i| \le 1$ for every $i \in \{1, \ldots, r\}$.

Let $\gamma \ge 1$ be fixed. We write

$$V = \underbrace{\sum_{j \in [m]} \sum_{i=1}^r \left(Y_i^{(j)} \right)^2}_{V_1} + \underbrace{\sum_{j \in [m]} \sum_{i=1}^r Y_i^{(j)} Y_{<i}^{(j)}}_{V_2} \tag{11}$$

and bound V_1 and V_2 separately starting with V_1.

Interchanging summations, $V_1 = \sum_{i=1}^r \sum_{j \in [m]} \left(Y_i^{(j)} \right)^2$. In the case $c = 1$, let $i \in \{1, \ldots, r\}$ be given. If $|G_i| = 0$, $\sum_{j \in [m]} \left(Y_i^{(j)} \right)^2 = 0$. If on the other hand $G_i = \{x_i\}$ for some $x_i \in \Sigma^c$,

$$\sum_{j \in [m]} \left(Y_i^{(j)} \right)^2 = \sum_{j \in [m]} \left(\sum_{k \in Q} v(x_i, k) \left([h(x_i) = j \oplus k] - \frac{1}{m} \right) \right)^2$$

$$= \sum_{j \in [m]} \left(\sum_{k \in [m]} v(x_i, k) \left([h(x_i) \oplus j = k] - \frac{1}{m} \right) \right)^2$$

$$= \sum_{j \in [m]} \left(\sum_{k \in [m]} v(x_i, k) \left([j = k] - \frac{1}{m} \right) \right)^2$$

$$= \sum_{j \in [m]} \left(v(x_i, j) - \frac{1}{m} \sum_{k \in [m]} v(x_i, k) \right)^2$$

$$\le \sum_{j \in [m]} v(x_i, j)^2$$

where the last inequality follows from the inequality $\mathbb{E}\left[(X - \mathbb{E}[X])^2 \right] \le \mathbb{E}\left[X^2 \right]$. Thus, we always have $V_1 \le v'(S) \le \frac{C_\gamma^c}{2} v'(S)$. In the case $c > 1$ we observe that the keys of G_i have a common position character. Hence, we can apply the induction hypothesis on the keys of G_i with the remaining $c - 1$ position characters to conclude that

$$\Pr\left[\sum_{j \in [m]} \left(Y_i^{(j)} \right)^2 > C_\gamma^{c-1} v'(G_i) \right] \le O_{\gamma,c}(|G_i| m^{-\gamma}) .$$

By a union bound,

$$\Pr\left[V_1 > \frac{C_\gamma^c}{2} v'(S) \right] \le \Pr\left[V_1 > C_\gamma^{c-1} v'(S) \right] \le \sum_{i \in [r]} O_{\gamma,c}(|G_i| m^{-\gamma}) = O_{\gamma,c}(|S| m^{-\gamma}) . \tag{12}$$

Next we proceed to bound V_2. For $0 \le i \le r$ define $Z_i = \sum_{j \in [m]} Y_i^{(j)} Y_{<i}^{(j)}$ with $Z_0 = 0$ and $\mathcal{F}_i = \sigma((h(\alpha_j))_{j=1}^i)$ with $\mathcal{F}_0 = \{\emptyset, \Omega\}$. As $Y_{<i}^{(j)}$ is $\mathcal{F}_{i-1}$ measurable for $j \in [m]$ it holds that

$$\mathbb{E}\left[Z_i \mid \mathcal{F}_{i-1} \right] = \sum_{j \in [m]} \mathbb{E}\left[Y_i^{(j)} \mid \mathcal{F}_{i-1} \right] Y_{<i}^{(j)} = 0 ,$$

and so $(Z_i, \mathcal{F}_i)_{i=0}^r$ is a martingale difference. We will define a modified martingale difference $(Z_i', \mathcal{F}_i)_{i=0}^r$ recursively as follows: We define the events A_i, B_i and C_i for $i \in \{1, \ldots, r\}$ as

$$A_i = \bigcap_{k=1}^{i} \left(\sum_{j \in [m]} \left(Y_k^{(j)} \right)^2 \le C_\gamma^{c-1} v'(G_k) \right),$$

$$B_i = \bigcap_{k=1}^{i} \left(\mathrm{Var}\left[Z_k \mid \mathcal{F}_{k-1} \right] \le m^{-1/2} v'(G_k) v'(G_{<k}) \right),$$

$$C_i = \left(\max_{1 \le k \le i} \{ Z_{<k}' \} \le \frac{C_\gamma^c}{2} v'(S) \right).$$

Finally, we let $Z_i' = [A_i \cap B_i \cap C_i] \cdot Z_i$. Clearly $B_i, C_i \in \mathcal{F}_{i-1}$. To see that this is also the case for A_i we note that for $k \le i$,

$$\sum_{j \in [m]} (Y_k^{(j)})^2 = \sum_{j \in [m]} (X_k^{(j \oplus h(\alpha_k))})^2 = \sum_{j \in [m]} (X_k^{(j)})^2 ,$$

and as each $X_k^{(j)}$ is $\mathcal{F}_{k-1}$-measurable it follows that $A_i \in \mathcal{F}_{i-1}$. Now, as $[A_i \cap B_i \cap C_i]$ is $\mathcal{F}_{i-1}$-measurable,

$$\mathbb{E}\left[Z_i' \mid \mathcal{F}_{i-1} \right] = [A_i \cap B_i \cap C_i] \mathbb{E}\left[Z_i \mid \mathcal{F}_{i-1} \right] = 0,$$

which implies that $(Z_i', \mathcal{F}_i)_{i=0}^r$ is a martingale difference.

If A_r, B_r, and C_r all occur then $\sum_{i=1}^r Z_i = \sum_{i=1}^r Z_i'$. In particular

$$\Pr\left[V_2 > \frac{C_\gamma^c}{2} v'(S) \right] = \Pr\left[\sum_{i \in [r]} Z_i > \frac{C_\gamma^c}{2} v'(S) \right] \le \Pr\left[\sum_{i=1}^r Z_i' > \frac{C_\gamma^c}{2} v'(S) \right] + \Pr\left[A_r^c \cup B_r^c \cup C_r^c \right] .$$

If C_r does not occur then $\sum_{i \in [r]} Z_i' > \frac{C_\gamma^c}{2} v'(S)$ so a union bound yields

$$\Pr\left[V_2 \ge \frac{C_\gamma^c}{2} v'(S) \right] \le 2 \Pr\left[\sum_{i=1}^r Z_i' > \frac{C_\gamma^c}{2} v'(S) \right] + \Pr\left[A_r^c \right] + \Pr\left[B_r^c \right] . \tag{13}$$

We now wish to apply Corollary 8 to the martingale difference $(Z_i', \mathcal{F}_i)_{i=0}^r$. Thus, we have to bound $|Z_i'|$ as well as the conditional variances $\mathrm{Var}\left[Z_i' \mid \mathcal{F}_{i-1} \right]$. For the bound on Z_i', observe that by the Cauchy-Schwarz inequality,

$$|Z_i'| = [A_i \cap B_i \cap C_i] \left| \sum_{j \in [m]} Y_i^{(j)} Y_{<i}^{(j)} \right| \le [A_i \cap B_i \cap C_i] \sqrt{\sum_{j \in [m]} \left(Y_i^{(j)} \right)^2} \sqrt{\sum_{j \in [m]} \left(Y_{<i}^{(j)} \right)^2} .$$

If A_i occurs we obtain

$$\sum_{j \in [m]} \left(Y_i^{(j)} \right)^2 \le C_\gamma^{c-1} v'(G_i) \le C_\gamma^{c-1} v'(S)^{1-1/c} v'_\infty(S)^{1/c} ,$$

by Lemma 13 and if A_i, B_i, and C_i all occur then

$$\sum_{j \in [m]} \left(Y_{<i}^{(j)} \right)^2 = \sum_{j \in [m]} \sum_{k < i} \left(Y_k^{(j)} \right)^2 + 2 Z_{<i}' \le C_\gamma^{c-1} v'(G_{<i}) + 2 C_\gamma^c v'(G_{<i}) \le 3 C_\gamma^c v'(S) .$$

In conclusion

$$|Z_i'| \le C_\gamma^{c-1/2} \sqrt{3} v'(S)^{1-1/(2c)} v'_\infty(S)^{1/(2c)} .$$

For the bound on the conditional variance note that if B_i occurs then $\mathrm{Var}\left[Z_i \mid \mathcal{F}_{i-1}\right] \leq m^{-1/2} v'(G_k) v'(G_{<k})$ and thus,

$$\mathrm{Var}\left[Z_i' \mid \mathcal{F}_{i-1}\right] = [A_i][B_i][C_i]\mathrm{Var}\left[Z_i \mid \mathcal{F}_{i-1}\right] \leq m^{-1/2} v'(G_k) v'(G_{<k}) \,.$$

It follows that $\sum_{i=1}^{r} \mathrm{Var}\left[Z_i' \mid \mathcal{F}_{i-1}\right] \leq m^{-1/2} v'(S)^2$. Letting

$$\sigma^2 = m^{-1/2} v'(S)^2$$

and

$$M = C_\gamma^{c-1/2} \sqrt{3} v'(S)^{1-1/(2c)} v_\infty'(S)^{1/(2c)}$$

in Corollary 8 we thus obtain

$$\Pr\left[\sum_{i=1}^{r} Z_i' > \frac{C_\gamma^c}{2} v'(S)\right] \leq \exp\left(-\frac{v'(S)^{1/c}}{3 C_\gamma^{2c-1} \cdot \sqrt{m} \cdot v_\infty'(S)^{1/c}} \mathcal{C}\left(\frac{(C\gamma)^{2c-1/2} \cdot \sqrt{3} \cdot \sqrt{m} \cdot v_\infty'(S)^{1/2c}}{2 v'(S)^{1/2c}}\right)\right) \,.$$

Applying Lemma 10 first with $b = \left(\frac{v_\infty'(S)}{v'(S)}\right)^{1/(2c)} \leq 1$ and then with $b = \sqrt{m} > 1$ yields

$$\frac{v'(S)^{1/c}}{3 C_\gamma^{2c-1} \sqrt{m} v_\infty'(S)^{1/c}} \mathcal{C}\left(\frac{C_\gamma^{2c-1/2} \sqrt{3} \sqrt{m} v_\infty'(S)^{1/2c}}{2 v'(S)^{1/2c}}\right) \geq \frac{1}{3 C_\gamma^{2c-1}} \mathcal{C}\left(\frac{C_\gamma^{2c-1/2} \sqrt{3} \sqrt{m}}{2}\right) \,.$$

We then use Lemma 9 to get

$$\frac{1}{3 C_\gamma^{2c-1}} \mathcal{C}\left(\frac{C_\gamma^{2c-1/2} \sqrt{3} \sqrt{m}}{2}\right) \geq \frac{\sqrt{C_\gamma}}{\sqrt{3} \cdot 4} \log\left(1 + \frac{(C\gamma)^{2c-1/2} \sqrt{3} \sqrt{m}}{2}\right) \geq \frac{\sqrt{C_\gamma}}{\sqrt{3} \cdot 8} \log(m) = \gamma \log(m) \,,$$

where we have used that $C_\gamma = 3 \cdot 8 \cdot \gamma^2$ and $\gamma \geq 1$. Combining this we get that

$$\Pr\left[\sum_{i=1}^{r} Z_i' > \frac{C_\gamma^c}{2} v'(S)\right] \leq m^{-\gamma} \,. \tag{14}$$

It thus suffices to bound the probabilities $\Pr[A_{r-1}^c]$ and $\Pr[B_{r-1}^c]$. For A_{r-1}^c, if $c = 1$ the discussion from the bound on V_1 proves that A_{r-1}^c never occurs. If $c > 1$, the inductive hypothesis on the groups G_i and a union bound yields

$$\Pr\left[A_{r-1}^c\right] = O_{\gamma,\epsilon,c}\left(\sum_{i=1}^{r} |G_i| m^{-\gamma}\right) = O(|S| m^{-\gamma}) \,. \tag{15}$$

For B_{r-1}^c, we can for each $i \in \{1, \ldots, r\}$ write

$$\mathrm{Var}\left[Z_i \mid \mathcal{F}_{i-1}\right] = \mathbb{E}\left[\left.\left(\sum_{j \in [m]} X_i^{(j \oplus h(\alpha_i))} Y_{<i}^{(j)}\right)^2 \right| \mathcal{F}_{i-1}\right]$$

$$= \frac{1}{m} \sum_{k \in [m]} \left(\sum_{j \in [m]} X_i^{(j \oplus k)} Y_{<i}^{(j)}\right)^2$$

$$= \frac{1}{m} \sum_{k \in [m]} \sum_{(j_1, j_2) \in [m]^2} Y_i^{(j_1 \oplus k)} Y_i^{(j_2 \oplus k)} Y_{<i}^{(j_1)} Y_{<i}^{(j_2)}$$

$$= \frac{1}{m} \sum_{\substack{(j_1, j_2, j_3, j_4) \in [m]^4 \\ j_1 \oplus j_2 \oplus j_3 \oplus j_4 = 0}} Y_i^{(j_1)} Y_i^{(j_2)} Y_{<i}^{(j_3)} Y_{<i}^{(j_4)}$$

31

Call this quantity T_i. It follows from Lemma 15 and Markov's inequality that

$$\Pr\left[T_i \geq m^{-1/2} v'(G_k) v'(G_{<k})\right] \leq \frac{\mathbb{E}\left[T_i^{2\gamma/(1-4\varepsilon)}\right]}{m^{\gamma/(1-4\varepsilon)} \left(v'(G_k) v'(G_{<k})\right)^{2\gamma/(1-4\varepsilon)}} \leq O_{\gamma,\varepsilon,c}(m^{-\gamma}).$$

Thus, $\Pr[B_{r-1}^c] = O(|S| m^{-\gamma})$ by a union bound.

Combining equations (11)-(15) we conclude that indeed $\Pr\left[V \geq C_\gamma^c v'(S)\right] = O_{\gamma,\varepsilon,c}(|S| m^{-\gamma})$. □

4.3 Establishing the Concentration Bound

With the results of the previous subsection at hand, we proceed to prove the first part of Theorem 4. We show that for a value function of support bounded in size by m^ε for some $\varepsilon < 1/4$, simple tabulation supports Chernoff-style bounds with added error probability inversely polynomial in m. For convenience, we restate the first part of Theorem 4 as Theorem 17. The statement is equivalent to Theorem 4 but for precision, we have chosen to write out the statement more explicitly.

Theorem 17. *Let $h\colon \Sigma^c \to [m]$ be a simple tabulation hash function and $S \subseteq \Sigma^c$ be a set of keys. Let $v\colon \Sigma^c \times [m] \to [-1,1]$ be a value such that the set $Q = \{i \in [m] \mid \exists x \in \Sigma^c : v(x,i) \neq 0\}$ satisfies $|Q| \leq m^\varepsilon$, where $\varepsilon < \frac{1}{4}$ is a constant. Define the random variable $W = \sum_{x \in S} v(x, h(x))$ and write $\mu = \mathbb{E}[W]$ and $\sigma^2 = \mathrm{Var}[W]$. Then for any constant $\gamma \geq 1$,*

$$\Pr\left[|W - \mu| \geq C_{\gamma,c} t\right] \leq 2\exp\left(-\sigma^2 \mathcal{C}\left(\frac{t}{\sigma^2}\right)\right) + O_{\gamma,\varepsilon,c}\left(|S| m^{-\gamma}\right) ,$$

where $C_{\gamma,c} = \left(1 + \frac{1}{\gamma}\right)^{3^{\frac{c(c-1)}{2}}} (Cc\gamma)^{3c}$ for some large enough universal constant C.

Proof. First, akin to the proof of Theorem 16, we may write

$$V = W - \mu = \sum_{x \in S} \sum_{k \in Q} v(x,k)\left([h(x) = j \oplus k] - \frac{1}{m}\right),$$

and note that

$$\mathrm{Var}[V] = \mathrm{Var}[W] = \sum_{x \in S}\left(\sum_{k \in Q} \frac{1}{m} v(x,k)^2 - \left(\sum_{k \in Q} \frac{1}{m} v(x,k)\right)^2\right).$$

We proceed by induction on c. For $c = 1$ we have full randomness and it follows immediately from Corollary 8 that

$$\Pr[|V| \geq t] \leq 2\exp\left(-\sigma^2 \mathcal{C}\left(\frac{t}{\sigma^2}\right)\right) .$$

Now assume that $c > 1$ and inductively that the result holds for smaller values of c. We define $v'(x) = \sum_{k \in Q} v(x,k)^2$ and for $X \subseteq \Sigma^c$ we let $v'(X) = \sum_{x \in X} v'(x)$ and define $v'_\infty(X) = \max_{x \in X} v'(x)$. Now, applying Lemma 13 with respect to $w = v'$ we get position characters $\alpha_1, \ldots, \alpha_r$ with corresponding groups $G_1, \ldots, G_r$, such that $\cup_{i=1}^r G_i = S$ and for every $i \in \{1, \ldots, r\}$, $v'(G_i) \leq v'(S)^{1-1/c} v'_\infty(S)^{1/c}$. For a bin $j \in [m]$ and an $i \in \{1, \ldots, r\}$ we again define

$$X_i^{(j)} = \sum_{x \in G_i} \sum_{k \in Q} v(x,k) \cdot \left([h(x \setminus \alpha_i) = j \oplus k] - \frac{1}{m}\right) , \qquad\qquad Y_i = X_i^{(h(\alpha_i))} .$$

Note that $\sum_{i=1}^r Y_i = V$. For $i \in \{1, \ldots, r\}$ we define the σ-algebra $\mathcal{F}_i = \sigma((h(\alpha_j))_{j=1}^i)$. We furthermore define $Y_0 = 0$ and $\mathcal{F}_0 = \{\emptyset, \Omega\}$. As Y_i is $\mathcal{F}_i$-measurable for $i \in [r+1]$ and $\mathbb{E}[Y_i \mid \mathcal{F}_{i-1}] = 0$ for $i \in \{1, \ldots, r\}$, $(Y_i, \mathcal{F}_i)_{i=0}^r$ is a martingale difference. Furthermore, for $i \in \{1, \ldots, r\}$,

$$\mathrm{Var}[Y_i \mid F_{i-1}] = \frac{1}{m} \sum_{j \in [m]} \left(X_i^{(j)}\right)^2 .$$

According to Theorem 16 there exists a constant $K = 3 \cdot 2^6 \cdot \gamma^2$ such that

$$\Pr \left[\sum_{j \in [m]} \left(X_i^{(j)} \right)^2 > K^{c-1} v'(G_i) \right] \leq O_{\gamma, \varepsilon, c} \left(|G_i| \, m^\gamma \right) \ . \tag{16}$$

For $i \in \{1, \ldots, r\}$ we define the events

$$A_i = \bigcap_{k \leq i} \left(\sum_{j \in [m]} (X_k^{(j)})^2 \leq K^{c-1} v'(G_k) \right) \ ,$$

$$B_i = \left(\max_{\substack{k \leq i \\ j \in [m]}} |X_k^{(j)}| \leq C_{\gamma+1, c-1} M \right) \ ,$$

for some M to be specified later. We define $Z_i = [A_i \cap B_i] Y_i$ for $i \in \{1, \ldots, r\}$ and $Z_0 = 0$. As both $A_i, B_i \in \mathcal{F}_{i-1}$ we have that $\mathbb{E} \left[Z_i \mid \mathcal{F}_{i-1} \right] = [A_i \cap B_i] \mathbb{E} \left[Y_i \mid \mathcal{F}_{i-1} \right] = 0$ for $\{1, \ldots, r\}$ so $(Z_i, \mathcal{F}_i)_{i=0}^r$ is a martingale difference. By definition of A_i and B_i it moreover holds for $i \in \{1, \ldots, r\}$ that

$$|Z_i| \leq C_{\gamma+1, c-1} M \quad \text{and} \quad \text{Var} \left[Z_i \mid \mathcal{F}_{i-1} \right] \leq \frac{K^{c-1} v'(G_i)}{m} \ .$$

Setting $\sigma_0^2 = \frac{K^{c-1} v'(S)}{m}$ and applying Corollary 8 we obtain

$$\Pr \left[\left| \sum_{i=1}^r Z_i \right| \geq t \right] \leq 2 \exp \left(-\frac{\sigma_0^2}{C_{\gamma+1, c-1}^2 M^2} \mathcal{C} \left(\frac{t C_{\gamma+1, c-1} M}{\sigma_0^2} \right) \right) \ . \tag{17}$$

If A_{r-1} and B_{r-1} both occur then $\sum_{i=1}^r Z_i = \sum_{i=1}^r Y_i$ so it must hold that

$$\Pr \left[|V| \geq t \right] \leq \Pr \left[\left| \sum_{i=1}^r Z_i \right| \geq t \right] + \Pr \left[A_{r-1}^c \right] + \Pr \left[B_{r-1}^c \right] \ .$$

We may assume that $m > 1$, i.e., the number of bins exceeds one, and then by the Cauchy-Schwarz inequality,

$$\sigma^2 = \sum_{x \in S} \left(\sum_{k \in Q} \frac{1}{m} v(x, k)^2 - \left(\sum_{k \in Q} \frac{1}{m} v(x, k) \right)^2 \right) \geq \sum_{x \in S} \left(\sum_{k \in Q} \frac{1}{m} v(x, k)^2 - \frac{1}{m^{2(1-\varepsilon)}} \sum_{k \in Q} \frac{1}{m^\varepsilon} v(x, k)^2 \right)$$

$$= \frac{v'(S)}{m} \left(1 - \frac{1}{m^{1-\varepsilon}} \right)$$

$$\geq \frac{v'(S)}{3m}$$

$$\geq \frac{\sigma_0^2}{3 K^{c-1}}$$

so using (17) we obtain

$$\Pr \left[|V| \geq C_{\gamma, c} t \right] \leq 2 \exp \left(-\frac{3 K^{c-1} \sigma^2}{C_{\gamma+1, c-1}^2 M^2} \mathcal{C} \left(\frac{C_{\gamma, c} \cdot t \cdot C_{\gamma+1, c-1} M}{3 K^{c-1} \sigma^2} \right) \right) + \Pr \left[A_{r-1}^c \right] + \Pr \left[B_{r-1}^c \right] \ . \tag{18}$$

By (16) and a union bound $\Pr \left[A_{r-1}^c \right] \leq O(|S| m^{-\gamma})$. For bounding $\Pr \left[B_{r-1}^c \right]$ we use the induction hypothesis on the groups, concluding that for $i \in \{1, \ldots, r\}$ and $j \in [m]$,

$$\Pr \left[\left| X_i^{(j)} \right| > C_{\gamma+1, c-1} M \right] \leq 2 \exp \left(-\sigma_i^2 \mathcal{C} \left(\frac{M}{\sigma_i^2} \right) \right) + O(|G_i| m^{-\gamma-1}) \ ,$$

33

where $\sigma_i^2 = \text{Var}\left[Y_i^{(j)}\right] \leq v'(G_i)/m$. By the initial assumption on the groups, this implies $\sigma_i^2 \leq v'(S)^{1-1/c}v'_\infty(S)/m$ and we denote the latter quantity τ^2. Combining with Lemma 11 we obtain by a union bound that

$$\Pr\left[B_{r-1}^c\right] \leq 2\,|S|\,m\exp\left(-\tau^2\mathcal{C}\left(\frac{M}{\tau^2}\right)\right) + O(|S|m^{-\gamma}) .$$

We fix M to be the unique real number with $\tau^2\mathcal{C}\left(\frac{M}{\tau^2}\right) = (\gamma+1)\log(m)$. With this choice of M, $\Pr\left[B_{r-1}^c\right] \leq O(|S|\,m^{-\gamma})$, so by (18) it suffices to show that

$$\frac{3K^{c-1}\sigma^2}{C_{\gamma+1,c-1}^2 M^2}\mathcal{C}\left(\frac{C_{\gamma,c}\cdot t\cdot C_{\gamma+1,c-1}M}{3K^{c-1}\sigma^2}\right) \geq \min\left\{\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right),\gamma\log(m)\right\} . \tag{19}$$

First since $\frac{C_{\gamma,c}C_{\gamma+1,c-1}}{3K^{c-1}} \geq 1$ Lemma 10 give us that

$$\frac{3\left(K\gamma\right)^{c-1}\sigma^2}{C_{\gamma+1,c-1}^2 M^2}\mathcal{C}\left(\frac{C_{\gamma,c}\cdot t\cdot C_{\gamma+1,c-1}M}{3\left(K\gamma\right)^{c-1}\sigma^2}\right) \geq \frac{C_{\gamma,c}}{C_{\gamma+1,c-1}}\frac{\sigma^2}{M^2}\mathcal{C}\left(\frac{tM}{\sigma^2}\right) .$$

Now by definition of $C_{\gamma,c}$ and $C_{\gamma+1,c-1}$ we get that

$$\frac{C_{\gamma,c}}{C_{\gamma+1,c-1}} = \frac{\left(1+\frac{1}{\gamma}\right)^{3\frac{c(c-1)}{2}}(Cc\gamma)^{3c}}{\left(1+\frac{1}{\gamma+1}\right)^{3\frac{(c-1)(c-2)}{2}}(Cc(\gamma+1))^{3(c-1)}} \geq (Cc\gamma)^3 .$$

So we have reduced the problem to showing that

$$(Cc\gamma)^3\frac{\sigma^2}{M^2}\mathcal{C}\left(\frac{tM}{\sigma^2}\right) \geq \min\left\{\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right),\gamma\log(m)\right\} .$$

For that we have to check a couple of cases.

Case 1. $\frac{tM}{\sigma^2} \leq 1$: Using Lemma 9 twice and the fact that $(Cc\gamma)^3 \geq \frac{3}{2}$, we get that

$$(Cc\gamma)^3\sigma^2\mathcal{C}\left(\frac{tM}{\sigma^2}\right) \geq \frac{(Cc\gamma)^3}{3}\frac{t^2}{\sigma^2} \geq \sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right) .$$

Case 2. $v'(S) \leq m^{(1-\varepsilon/c)\left(1+\frac{1}{2c-1}\right)}$: We then get that

$$\tau^2 \leq \frac{v'(S)^{1-1/c}v'_\infty(S)}{m} \leq \frac{m^{(1-\varepsilon/c)\left(1-\frac{1}{2c-1}\right)}}{m^{1-\varepsilon/c}} = m^{-\frac{1-\varepsilon/c}{2c-1}} .$$

Now we note that $M \leq 12\gamma c$ since

$$\tau^2\mathcal{C}\left(\frac{12\gamma c}{\tau^2}\right) \geq 12\gamma c\log\left(1+\frac{12\gamma c}{\tau^2}\right)/2 \geq 6\gamma c\log(1/\tau^2) \geq 6\gamma c\frac{1-\varepsilon/c}{2c-1}\log(m) \geq (\gamma+1)\log(m) ,$$

where we have used that $\varepsilon \leq \frac{1}{4}$ and $\gamma \geq 1$.
We then get that

$$(Cc\gamma)^3\frac{\sigma^2}{M^2}\mathcal{C}\left(\frac{tM}{\sigma^2}\right) \geq \frac{(Cc\gamma)^3}{M}\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right) \geq \frac{(Cc\gamma)^3}{12c\gamma}\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right) \geq \sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right) .$$

Case 3. $\frac{tM}{\sigma^2} > 1$ **and** $v'(S) > m^{(1-\varepsilon/c)\left(1+\frac{1}{2c-1}\right)}$**:** We see that $M \leq \max\left\{6\gamma\log(m), \sqrt{6\gamma\log(m)}\tau\right\}$ since

$$\tau^2\mathcal{C}\left(\frac{\max\left\{6\gamma\log(m), \sqrt{6\gamma\log(m)}\tau\right\}}{\tau^2}\right) \geq \max\left\{\frac{6\gamma\log(m)}{3}, \frac{6\gamma\log(m)}{3}\right\} \geq (\gamma+1)\log(m)\,,$$

where we have used Lemma 9 and that $\gamma \geq 1$. Now we have that $\sigma^2 \geq \frac{v'(S)}{3m} > m^{\frac{1-2\varepsilon}{2c-1}}/3$ and $\tau^2 \leq \frac{v'(S)^{1-1/c}v'_\infty(S)^{1/c}}{m}$. Combining this we get that

$$\frac{\sigma^2}{M^2} \geq \min\left\{\frac{\sigma^2}{36\gamma^2\log(m)^2}, \frac{\sigma^2}{6\gamma\log(m)\tau^2}\right\}$$

$$\geq \min\left\{\frac{m^{\frac{1-2\varepsilon}{2c-1}}}{108\gamma^2\log(m)^2}, \left(\frac{v'(S)}{v'_\infty(S)}\right)^{1/c} \cdot \frac{1}{3\cdot 6\gamma\log(m)}\right\}$$

$$\geq \min\left\{\frac{m^{\frac{1-2\varepsilon}{2c-1}}}{108\gamma^2\log(m)^2}, m^{\frac{1-2\varepsilon}{2c}} \cdot \frac{1}{18\gamma\log(m)}\right\}$$

$$\geq \frac{m^{\frac{1}{4c}}}{108\gamma^2\log(m)^2}$$

$$\geq \frac{\log(m)}{108\cdot 4^3 c^3\gamma^2}\,,$$

where have used that $\varepsilon < \frac{1}{4}$ and that $\frac{m^{\frac{1}{4c}}}{\log(m)^2} \geq \frac{\log(m)}{4^3 c^3}$. Now we get that

$$(Cc\gamma)^3\frac{\sigma^2}{M^2}\mathcal{C}\left(\frac{tM}{\sigma^2}\right) \geq (Cc\gamma)^3\frac{\log(m)}{108\cdot 4^3 c^3\gamma^2}/3 \geq \gamma\log(m)\,.$$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

5 General Value Functions – Arbitrary Bins

The goal of this section is to prove Theorem 5, the second step towards Theorem 2. Again, we postpone the argument that our concentration bounds are query invariant to Section 7. Recall that Theorem 5 is concerned with a hash function of the form $h = \tau \circ g$, where $g : \Sigma^c \to [m]$ is a simple tabulation hash function and τ is a uniformly random permutation. Our goal is to prove that for any value function $v : \Sigma^c \times [m] \to [-1, 1]$, the sum $\sum_{x\in\Sigma^c} v(x, h(x))$ is strongly concentrated with high probability in m. This result follows by combining the distributional properties of g with the randomness of τ.

We start out by proving a lemma. The lemma describes properties we need g to possess for the final composition with τ to yield Chernoff-style concentration.

Lemma 18. *Let $m \geq 2$ be an integer and $C, T \in \mathbb{R}^+$ positive reals. Furthermore, let $\mathcal{V} : [m] \times [m] \to \mathbb{R}$ be a value function satisfying $\sum_{i\in[m]} \mathcal{V}(i, j) = 0$ for every $j \in [m]$ and such that*

$$\max_{i,j\in[m]} |\mathcal{V}(i,j)| \leq M := \max\left\{C, \frac{\sigma^2}{T}\right\}\,,$$

where $\sigma^2 = \frac{1}{m}\sum_{i\in[m]}\sum_{j\in[m]} \mathcal{V}(i, j)^2$. If $\tau : [m] \to [m]$ is a uniformly random permutation, then the random variable $Z = \sum_{i\in[m]} \mathcal{V}(\tau(i), i)$ satisfies

$$\Pr[|Z| \geq Dt] \leq 4\left(\exp\left(-\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right)\right) + \exp\left(-\frac{T^2}{2\sigma^2}\right)\right)\,,$$

where $D = \max\{8C, 12\}$ is a universal constant depending on C.

Proof. We define $Y_1 = \sum_{i=0}^{\lceil m/2 \rceil - 1} \mathcal{V}(\tau(i), i)$ and $Y_2 = \sum_{i=\lceil m/2 \rceil}^{m-1} \mathcal{V}(\tau(i), i)$. Since $Z = Y_1 + Y_2$ it follows that if $Z > Dt$ then there exists $i \in \{1, 2\}$ such that $Y_i \geq \frac{D}{2}t$. It suffices to show that

$$\Pr\left[Y_i \geq \frac{D}{2}t\right] \leq \exp\left(-\sigma^2 \mathcal{C}\left(\frac{t}{\sigma^2}\right)\right) + \exp\left(-\sigma^2 \mathcal{C}\left(\frac{T}{\sigma^2}\right)\right), \tag{20}$$

for $i \in \{1, 2\}$. A union bound over i then yields a bound on $\Pr[Z \geq Dt]$. Since we may instead consider the value function $-\mathcal{V}$, the same argument yields a bound on $\Pr[Z \leq -Dt]$, which concludes the proof.

Thus, we shall prove (20) for Y_1 – the proof is completely analogous for Y_2. Define the filtration $(\mathcal{F}_i)_{i=0}^{\lceil m/2 \rceil}$ by $\mathcal{F}_i = \sigma\left((\tau(j))_{j \in [i]}\right)$ and let $X_i = \mathbb{E}[Y_1 \mid \mathcal{F}_i]$ such that $(X_i, \mathcal{F}_i)_{i=0}^{\lceil m/2 \rceil}$ is a martingale, $X_0 = \mathbb{E}[Y_1]$, and $X_{\lceil m/2 \rceil} = Y_1$. Towards applying Corollary 8, we bound $|X_i - X_{i-1}|$ and $\sum_{i=1}^{\lceil m/2 \rceil} \mathrm{Var}[X_i - X_{i-1} \mid \mathcal{F}_{i-1}]$.

First, we bound $|X_i - X_{i-1}|$. We start by writing

$$X_i - X_{i-1} = \mathbb{E}[Y_1 \mid \mathcal{F}_i] - \mathbb{E}[Y_1 \mid \mathcal{F}_{i-1}]$$

$$= \mathcal{V}(\tau(i-1), i-1) - \mathbb{E}[\mathcal{V}(\tau(i-1), i-1) \mid \mathcal{F}_{i-1}] + \sum_{k=i}^{\lceil m/2 \rceil - 1} \left(\mathbb{E}[\mathcal{V}(\tau(k), k) \mid \mathcal{F}_i] - \mathbb{E}[\mathcal{V}(\tau(k), k) \mid \mathcal{F}_{i-1}]\right).$$

Now, note that for $k \geq i$,

$$\mathbb{E}[\mathcal{V}(\tau(k), k) \mid \mathcal{F}_i] = -\frac{1}{m-i} \sum_{j=0}^{i-1} \mathcal{V}(\tau(j), k),$$

since $\sum_{\ell \in [m]} \mathcal{V}(\ell, k) = 0$, and furthermore,

$$\mathbb{E}[\mathcal{V}(\tau(k), k) \mid \mathcal{F}_{i-1}] = -\frac{1}{m-i}\left(\mathbb{E}[\mathcal{V}(\tau(i-1), k) \mid \mathcal{F}_{i-1}] + \sum_{j=0}^{i-2} \mathcal{V}(\tau(j), k)\right).$$

Hence, it follows that

$$X_i - X_{i-1} = \mathcal{V}(\tau(i-1), i-1) - \mathbb{E}[\mathcal{V}(\tau(i-1), i-1) \mid \mathcal{F}_{i-1}]$$

$$- \frac{1}{m-i} \sum_{k=i}^{\lceil m/2 \rceil - 1} \left(\mathcal{V}(\tau(i-1), k) - \mathbb{E}[\mathcal{V}(\tau(i-1), k) \mid \mathcal{F}_{i-1}]\right).$$

Since $|\mathcal{V}(i, j)| \leq M$ for all $i, j \in [m]$, it follows that $|X_i - X_{i-1}| \leq 4M$.

Second, we bound $\mathrm{Var}[X_i - X_{i-1} \mid \mathcal{F}_{i-1}]$. To this end, observe that

$$\mathrm{Var}[X_i - X_{i-1} \mid \mathcal{F}_{i-1}] = \mathrm{Var}\left[\mathcal{V}(\tau(i-1), i-1) - \frac{1}{m-i} \sum_{k=i}^{\lceil m/2 \rceil - 1} \mathcal{V}(\tau(i-1), k) \,\middle|\, \mathcal{F}_{i-1}\right]$$

$$\leq 2\left(\mathrm{Var}[\mathcal{V}(\tau(i-1), i-1) \mid \mathcal{F}_{i-1}] + \frac{1}{m-i} \sum_{k=i}^{\lceil m/2 \rceil - 1} \mathrm{Var}[\mathcal{V}(\tau(i-1), k) \mid \mathcal{F}_{i-1}]\right),$$

where the inequality follows from the fact that $2\mathrm{Cov}(A, B \mid \mathcal{H}) \leq \mathrm{Var}[A \mid \mathcal{H}] + \mathrm{Var}[B \mid \mathcal{H}]$, for any random

variables A and B and any sigma algebra $\mathcal{H}$. For any $k \in [m]$,

$$\text{Var}\left[\mathcal{V}(\tau(i-1),k)\mid\mathcal{F}_{i-1}\right] \leq \mathbb{E}\left[\mathcal{V}(\tau(i-1),k)^2\mid\mathcal{F}_{i-1}\right]$$

$$= \frac{1}{m-i+1}\sum_{j\in[m]\setminus\tau([i-1])}\mathcal{V}(j,k)^2$$

$$\leq \frac{1}{m-i+1}\sum_{j\in[m]}\mathcal{V}(j,k)^2$$

$$\leq \frac{2}{m}\sum_{j\in[m]}\mathcal{V}(j,k)^2\,,$$

where the last inequality follows from the fact that $i \leq \lceil m/2 \rceil$. Hence,

$$\text{Var}\left[X_i - X_{i-1}\mid\mathcal{F}_{i-1}\right] \leq 2\left(\text{Var}\left[\mathcal{V}(\tau(i-1),i-1)\mid\mathcal{F}_{i-1}\right] + \frac{1}{m-i}\sum_{k=i}^{\lceil m/2\rceil-1}\text{Var}\left[\mathcal{V}(\tau(i-1),k)\mid\mathcal{F}_{i-1}\right]\right)$$

$$\leq \frac{4}{m}\sum_{j\in[m]}\mathcal{V}(j,i)^2 + \frac{2}{m-i}\cdot\frac{2}{m}\sum_{k=i}^{\lceil m/2\rceil-1}\sum_{j\in[m]}\mathcal{V}(j,k)^2$$

$$\leq \frac{4}{m}\sum_{j\in[m]}\mathcal{V}(j,i)^2 + \frac{16}{m^2}\sum_{k\in[m]}\sum_{j\in[m]}\mathcal{V}(j,k)^2\,,$$

again using that $i \leq \lceil m/2 \rceil$. We now see that

$$\sum_{i=1}^{\lceil m/2\rceil}\text{Var}\left[X_i - X_{i-1}\mid\mathcal{F}_{i-1}\right] \leq \sum_{i=1}^{\lceil m/2\rceil}\left(\frac{4}{m}\sum_{j\in[m]}\mathcal{V}(j,i)^2 + \frac{16}{m^2}\sum_{k\in[m]}\sum_{j\in[m]}\mathcal{V}(j,k)^2\right)$$

$$\leq \sum_{i\in[m]}\left(\frac{4}{m}\sum_{j\in[m]}\mathcal{V}(j,i)^2 + \frac{16}{m^2}\sum_{k\in[m]}\sum_{j\in[m]}\mathcal{V}(j,k)^2\right)$$

$$\leq 20\sigma^2\,.$$

The assumption on $\mathcal{V}$ implies that $\mathbb{E}\left[\mathcal{V}(\tau(i),i)\right] = 0$ for each $i \in [m]$, so also $\mathbb{E}\left[Y_1\right] = 0$. Applying Corollary 8 then yields,

$$\Pr\left[Y_1 \geq \frac{D}{2}t\right] \leq \exp\left(-\frac{20\sigma^2}{(4M)^2}\mathcal{C}\left(\frac{(D/2)t\cdot 4M}{20\sigma^2}\right)\right) = \exp\left(-\frac{5\sigma^2}{4M^2}\mathcal{C}\left(\frac{DMt}{10\sigma^2}\right)\right).$$

The goal is now to show that

$$\frac{5\sigma^2}{4M^2}\mathcal{C}\left(\frac{DMt}{10\sigma^2}\right) \geq \min\left\{\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right), \frac{T^2}{2\sigma^2}\right\}. \tag{21}$$

Because if this is the case, then as desired

$$\Pr\left[Y_1 \geq \frac{D}{2}t\right] \leq \exp\left(-\min\left\{\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right), \frac{T^2}{2\sigma^2}\right\}\right) \leq \exp\left(-\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right)\right) + \exp\left(-\frac{T^2}{2\sigma^2}\right).$$

We check (21) by cases. This completes the proof.

Case 1. $M \leq \frac{10}{D}$: In this case, $\frac{DM}{10} \leq 1$. Thus, by Lemma 10,

$$\frac{5\sigma^2}{4M^2}\mathcal{C}\left(\frac{DMt}{10\sigma^2}\right) \geq \frac{D^2}{80}\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right) \geq \sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right)\,,$$

using that $D \geq 12 \geq \sqrt{80}$.

Case 2. $\frac{10}{D} \leq M \leq C$**:** In this case, $\frac{DM}{10} \geq 1$. Thus, by Lemma 10,

$$\frac{5\sigma^2}{4M^2}\mathcal{C}\left(\frac{DMt}{10\sigma^2}\right) \geq \frac{D}{8M}\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right) \geq \frac{D}{8C}\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right) \geq \sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right) ,$$

using that $D \geq 8C$.

Case 3. $M \leq \frac{\sigma^2}{T}$**:** In this case, recall that $D \geq 12$ such that $\frac{D}{10} \geq 1$ and we may apply Lemma 10, yielding

$$\frac{5\sigma^2}{4M^2}\mathcal{C}\left(\frac{DMt}{10\sigma^2}\right) \geq \frac{5}{4}\frac{T^2}{\sigma^2}\mathcal{C}\left(\frac{D}{10}\frac{t}{T}\right) \geq \frac{D}{8}\frac{T^2}{\sigma^2}\mathcal{C}\left(\frac{t}{T}\right) .$$

By Lemma 9,

$$\mathcal{C}\left(\frac{t}{T}\right) \geq \mathcal{C}\left(\min\left\{\frac{t}{T}, 1\right\}\right) \geq \min\left\{\frac{t^2}{3T^2}, \frac{1}{3}\right\} .$$

So finally,

$$\frac{5\sigma^2}{4M^2}\mathcal{C}\left(\frac{DMt}{10\sigma^2}\right) \geq \min\left\{\frac{D}{24}\frac{t^2}{\sigma^2}, \frac{D}{24}\frac{T^2}{\sigma^2}\right\} \geq \min\left\{\frac{D}{12}\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right), \frac{D}{24}\frac{T^2}{\sigma^2}\right\} \geq \min\left\{\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right), \frac{T^2}{2\sigma^2}\right\} ,$$

where we have used Lemma 9 and the fact that $D \geq 12$.

$\square$

With this result in hand we are ready to prove Theorem 5. We restate it here in a more technically explicit version. For a more intuitive understanding, please refer back to the original statement. Note that we only require the hash function h of the theorem to be 2-independent, whereas Theorem 5 requires the hash function to be 3-independent. The difference lies in that the statement of Theorem 5 is slightly stronger, guaranteeing query invariance. Having deferred the treatment of query invariance until later, we only need 2-independence for now.

Theorem 19. *Let $\varepsilon \in (0,1]$ and $m \geq 2$ be given. Let $h\colon A \to [m]$ be a 2-independent hash function satisfying the following. For every $\gamma > 0$ and every value function $\tilde{v}\colon A \times [m] \to [-1,1]$ such that $Q = \{i \in [m] \mid \exists x \in A\colon \tilde{v}(x,i) \neq 0\}$ has size $|Q| \leq m^\varepsilon$, the random variables $W = \sum_{x \in A} \tilde{v}(x, h(x))$ and $W_j = \sum_{x \in A} \tilde{v}(x, h(x) \oplus j), j \in [m]$ with mean $\mu_W = \mathbb{E}[W] = \mathbb{E}[W_j]$ and variance $\sigma_W^2 = \mathrm{Var}[W]$ satisfy the inequalities*

$$\Pr\left[|W - \mu_W| \geq C \cdot t\right] \leq 2\exp\left(-\sigma_W^2\mathcal{C}\left(\frac{t}{\sigma_W^2}\right)\right) + O(|A|\, m^{-\gamma}), \tag{22}$$

$$\Pr\left[\sum_{j \in [m]}(W_j - \mu_W)^2 \geq D \cdot \sum_{x \in A}\sum_{k \in Q}\tilde{v}(x,k)^2\right] = O(|A|\, m^{-\gamma}), \tag{23}$$

for every $t > 0$, where C and D are universal constants depending on γ and ε.

Let $v\colon A \times [m] \to [-1,1]$ be any value function, $\tau\colon [m] \to [m]$ a uniformly random permutation independent of h, and $\gamma > 0$. The random variable $U = \sum_{x \in A} v(x, \tau(h(x)))$ with expectation $\mu = \mathbb{E}[U]$ and variance $\sigma^2 = \mathrm{Var}[U]$ satisfies

$$\Pr\left[|U - \mu| \geq E \cdot t\right] \leq 6\exp\left(-\sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right)\right) + O(|A|\, m^{-\gamma}) \tag{24}$$

for every $t > 0$, where E is a universal constant depending on γ and ε.

Proof. Define $v' : A \times [m] \to [-1, 1]$ by letting $v'(x, i) = \frac{1}{2}\left(v(x, i) - \frac{1}{m}\sum_{j\in[m]} v(x, j)\right)$ and write $V = U - \mu$. Since $\sum_{i\in[m]}\left([\tau(h(x)) = i] - \frac{1}{m}\right) = 0$, we may write

$$V = \sum_{x\in A}\sum_{i\in[m]} v(x, i)[\tau(h(x)) = i] - \frac{1}{m}\sum_{x\in A}\sum_{i\in[m]} v(x, i) + \sum_{x\in A}\left(\left(\sum_{i\in[m]}\left([\tau(h(x)) = i] - \frac{1}{m}\right)\right)\cdot\left(\frac{1}{m}\sum_{j\in[m]} v(x, j)\right)\right)$$

$$= \sum_{x\in A}\sum_{i\in[m]}\left(v(x, i) - \frac{1}{m}\sum_{j\in[m]} v(x, j)\right)\left([\tau(h(x)) = i] - \frac{1}{m}\right)$$

$$= 2\sum_{x\in A}\sum_{i\in[m]} v'(x, i)\left([\tau(h(x)) = i] - \frac{1}{m}\right).$$

We write $V' = \sum_{x\in A}\sum_{i\in[m]} v'(x, i)\left([\tau(h(x)) = i] - \frac{1}{m}\right)$ such that $V = 2V'$. We note that by 2-independence

$$\sigma^2 = \sum_{x\in A}\mathrm{Var}\left[v(x, \tau(h(x)))\right] = \sum_{x\in A}\mathbb{E}\left[\left(v(x, \tau(h(x))) - \frac{1}{m}\sum_{j\in[m]} v(x, j)\right)^2\right] = \frac{4}{m}\sum_{x\in A}\sum_{i\in[m]} v'(x, i)^2.$$

Thus, we may write $\sigma'^2 = \mathrm{Var}\left[V'\right] = \frac{1}{m}\sum_{x\in A}\sum_{i\in[m]} v'(x, i)^2$. We proceed to show that for some constant E' depending on γ and ε,

$$\Pr\left[|V'| \geq E' \cdot t\right] \leq 6\exp\left(-\sigma'^2 h\left(\frac{t}{\sigma'^2}\right)\right) + O(|A|m^{-\gamma}),$$

As $\sigma' \leq \sigma$ and $V = 2V'$ the theorem then follows with $E = 2E'$ by applying Lemma 11.

For $i \in [m]$ we define $\sigma_i^2 = \frac{1}{m}\sum_{x\in A} v'(x, i)^2$, so that $\sum_{i\in[m]}\sigma_i^2 = \sigma'^2$. Assume without loss of generality that $\sigma_0^2 \geq \cdots \geq \sigma_{m-1}^2$. Now define $\mathcal{V} : [m] \times [m] \to \mathbb{R}$ by

$$\mathcal{V}(i, j) = \sum_{x\in A} v'(x, j)\left([h(x) = i] - \frac{1}{m}\right).$$

Note that for any $j \in [m]$, $\sum_{i\in[m]}\mathcal{V}(i, j) = 0$, regardless of the (random) choice of h. With this definition, $V' = \sum_{i\in[m]}\mathcal{V}(i, \tau(i)) = \sum_{j\in[m]}\mathcal{V}(\tau^{-1}(j), j)$. Now let

$$V_1 = \sum_{j\in[m^\varepsilon]}\mathcal{V}(\tau^{-1}(j), j) \quad\text{and}\quad V_2 = \sum_{j\in[m]\setminus[m^\varepsilon]}\mathcal{V}(\tau^{-1}(j), j),$$

and note that $V_1 + V_2 = V'$. Defining value functions $v_1', v_2' : A \times [m] \to [-1, 1]$ by

$$v_1'(x, i) = \begin{cases} v'(x, i), & \text{if } i \in [m^\varepsilon] \\ 0, & \text{otherwise} \end{cases} \quad\text{and}\quad v_2'(x, i) = \begin{cases} v'(x, i), & \text{if } i \in [m]\setminus[m^\varepsilon] \\ 0, & \text{otherwise} \end{cases},$$

we observe that

$$V_1 = \sum_{x\in A} v_1'(x, \tau(h(x))) - \mathbb{E}\left[\sum_{x\in A} v_1'(x, \tau(h(x)))\right] \quad\text{and}\quad V_2 = \sum_{x\in A} v_2'(x, \tau(h(x))) - \mathbb{E}\left[\sum_{x\in A} v_2'(x, \tau(h(x)))\right]$$

Let $D \geq 1$ be such that Eq. (23) holds with added error probability $O(|A|m^{-\gamma-1})$ and let $M = \max\left\{C, \frac{\sigma'}{\sqrt{2D\gamma\log m}}\right\}$ for some large constant C to be fixed later. Define the two events

$$\mathcal{A} = \bigcap_{j\in[m]\setminus[m^\varepsilon]}\left(\max_{i\in m}|\mathcal{V}(i, j)| \leq M\right) \quad\text{and}\quad \mathcal{B} = \bigcap_{j\in[m]}\left(\sum_{i\in[m]}\mathcal{V}(i, j)^2 < D\sigma_j^2 m\right).$$

39

By a union bound,

$$\Pr[|V'| \geq E't] \leq \Pr[|V_1| \geq E't/2] + \Pr[|[\mathcal{A}] \cdot [\mathcal{B}] \cdot V_2| \geq E't/2] + \Pr[\mathcal{A}^c] + \Pr[\mathcal{B}^c],$$

and we proceed to bound each of these terms individually.

First, we bound $\Pr[|V_1| \geq E't/2]$. To do so, suppose we fix the permutation $\tau = \tau_0$. With this conditioning and by 2-independence,

$$\mathrm{Var}\left[V_1 \mid \tau = \tau_0\right] = \mathrm{Var}\left[\sum_{x \in A} [\tau_0(h(x)) \in [m^\varepsilon]] \cdot v'(x, \tau_0(h(x)))\right] \leq \sum_{x \in A} \mathbb{E}\left[[\tau_0(h(x)) \in [m^\varepsilon]] \cdot v'(x, \tau_0(h(x)))^2\right]$$

$$= \frac{1}{m} \sum_{x \in A} \sum_{j \in [m^\varepsilon]} v'(x, j)^2 \leq \sigma'^2.$$

Defining $\overline{v} : A \times [m] \to [-1, 1]$ by $\overline{v}(x, i) = v_1'(x, \tau_0(i))$ it holds that

$$V_1 = \sum_{x \in A} \sum_{i \in \tau_0^{-1}([m^\varepsilon])} \overline{v}(x, i) \left([h(x) = i] - \frac{1}{m}\right).$$

As $\overline{v}$ has support of size at most m^ε we can apply Eq. (22) to conclude that

$$\Pr[|V_1| \geq E't/2 \mid \tau = \tau_0] \leq 2 \exp\left(-\sigma'^2 \mathcal{C}\left(\frac{t}{\sigma'^2}\right)\right) + O(|A| m^{-\gamma}),$$

if the constant E' is large enough. Since this holds for any fixed τ_0, it also holds for the unconditioned probability.

We now bound $\Pr[|[\mathcal{A}] \cdot [\mathcal{B}] \cdot V_2| \geq E't/2]$. It suffices to condition on $h = h_0$ for some h_0 satisfying that $[\mathcal{A}] = [\mathcal{B}] = 1$ and make the bound over the randomness of τ. For this we may use Lemma 18. Indeed if $h = h_0$ for some h_0 such that $[\mathcal{A}] = [\mathcal{B}] = 1$, then $\sum_{i \in [m]} \sum_{j \in [m]} \frac{1}{m} \mathcal{V}(i, j)^2 \leq D\sigma'^2$. Here we used the conditioning on $\mathcal{A}$. Define the function $\mathcal{V}_0 : [m] \times [m] \to \mathbb{R}$ by $\mathcal{V}_0(i, j) = \mathcal{V}(i, j)$ when $j \in [m] \setminus [m^\varepsilon]$ and $\mathcal{V}_0(i, j) = 0$ otherwise. Then also $\sum_{i \in [m]} \sum_{j \in [m]} \frac{1}{m} \mathcal{V}_0(i, j)^2 \leq D\sigma'^2$ and further, for each $j \in [m]$, $\sum_{i \in [m]} \mathcal{V}_0(i, j) = 0$. Finally, the conditioning on $\mathcal{B}$ gives that $\max_{i, j \in [m]} \mathcal{V}_0(i, j) \leq M$. Note that $V_2 = \sum_{j \in [m]} \mathcal{V}_0(\tau^{-1}(j), j)$. Applying Lemma 18 to $\mathcal{V}_0$, noting that the bound obtained in that lemma is increasing in σ, we obtain that

$$\Pr\left[|V_2| \geq E't/2\right] \leq 4 \left(\exp\left(-D\sigma'^2 \mathcal{C}\left(\frac{t}{D\sigma'^2}\right)\right) + \exp\left(-\gamma \log m\right)\right) = 4 \exp\left(-\Omega\left(\sigma'^2 \mathcal{C}\left(\frac{t}{\sigma'^2}\right)\right)\right) + O(m^{-\gamma}),$$

if E' is sufficiently large. From this it follows that,

$$\Pr[|[\mathcal{A}] \cdot [\mathcal{B}] \cdot |V_2| \geq E't/2] \leq 4 \exp\left(-\sigma'^2 \mathcal{C}\left(\frac{t}{\sigma'^2}\right)\right) + O(m^{-\gamma}).$$

We finally need to bound $\Pr[\mathcal{A}^c]$ and $\Pr[\mathcal{B}^c]$. By the choice of D and a union bound we obtain that $\Pr[\mathcal{B}^c] = O(|A| m^{-\gamma})$, so for completing the proof it suffices to bound $\Pr[\mathcal{A}^c]$ which we proceed to do now. More specifically we bound $\Pr[|\mathcal{V}(i, j)| \geq M]$ for each $(i, j) \in [m] \times ([m] \setminus [m^\varepsilon])$, finishing with a union bound over the m^2 choices. So let $(i, j) \in [m] \times ([m] \setminus [m^\varepsilon])$ be fixed and define $\tilde{v} : A \times [m] \to [-1, 1]$ by $\tilde{v}(x, i) = v_2'(x, j)$ and $\tilde{v}(x, k) = 0$ for $k \neq i$. Then $\tilde{v}$ has support $A \times \{i\}$,

$$\mathcal{V}(i, j) = \sum_{x \in A} \sum_{k \in [m]} \tilde{v}(x, k) \left([\tau(h(x)) = i] - \frac{1}{m}\right),$$

and $\mathrm{Var}\left[\mathcal{V}(i, j)\right] \leq \frac{1}{m} \sum_{x \in A} v_2'(x, j)^2 = \sigma_j^2 \leq \sigma'^2 / m^\varepsilon$. The last inequality follows from our assumption that $\sigma_0^2 \geq \cdots \geq \sigma_{m-1}^2$ and $j \geq m^\varepsilon$.

By the assumption of Eq. (22) with γ replaced by $\gamma + 2$ it follows that

$$\Pr[|\mathcal{V}(i,j)| \geq M] \leq 2\exp\left(-\Omega\left(\sigma_j^2 \mathcal{C}\left(\frac{M}{\sigma_j^2}\right)\right)\right) + O(|A|m^{-\gamma-2}) \leq 2\exp\left(-D'\frac{\sigma'^2}{m^\varepsilon}\mathcal{C}\left(\frac{Mm^\varepsilon}{\sigma'^2}\right)\right) + O(|A|m^{-\gamma-2}),$$

for some constant D'. We finish the proof by showing that if the constant C from the definition of M is large enough, then

$$2\exp\left(-D'\frac{\sigma'^2}{m^\varepsilon}\mathcal{C}\left(\frac{Mm^\varepsilon}{\sigma'^2}\right)\right) = O(m^{-\gamma-2}).$$

For this it suffices to show that if C is large enough and m is greater than some constant, then

$$\frac{\sigma'^2}{m^\varepsilon}\mathcal{C}\left(\frac{Mm^\varepsilon}{\sigma'^2}\right) \geq \frac{(\gamma+2)\log m}{D'}.$$

Suppose first that $\sigma'^2 \leq m^{\varepsilon/2}$. In that case we use Lemma 9 to conclude that

$$\frac{\sigma'^2}{m^\varepsilon}\mathcal{C}\left(\frac{Mm^\varepsilon}{\sigma'^2}\right) \geq \frac{M}{2}\log\left(\frac{Mm^\varepsilon}{\sigma'^2}+1\right) \geq \frac{C}{2}\log\left(Cm^{\varepsilon/2}+1\right) \geq \frac{C\varepsilon}{4}\log m,$$

so if $C \geq 4\frac{\gamma+2}{D'\varepsilon}$ this is at least $\frac{(\gamma+2)\log m}{D'}$. Now suppose $m^{\varepsilon/2} < \sigma'^2 \leq m^{2\varepsilon}/(2D\gamma\log m)$. In that case we recall that $M = \max\left\{C, \frac{\sigma'}{\sqrt{2D\gamma\log m}}\right\}$ and use the bound

$$\frac{\sigma'^2}{m^\varepsilon}\mathcal{C}\left(\frac{Mm^\varepsilon}{\sigma'^2}\right) \geq \frac{M}{2}\log\left(\frac{Mm^\varepsilon}{\sigma'^2}+1\right) \geq \frac{\sigma'}{\sqrt{8D\gamma\log m}}\log\left(\frac{m^\varepsilon}{\sigma'\sqrt{2D\gamma\log m}}+1\right) = \Omega\left(\frac{m^{\varepsilon/4}}{\sqrt{\log m}}\right).$$

If m is larger than some constant, this is certainly at least $\frac{(\gamma+2)\log m}{D'}$. Finally suppose that $\sigma'^2 > m^{2\varepsilon}/(2D\gamma\log m)$. Using the inequality $\log(1+x) \geq \frac{x}{2}$ holding for $0 \leq x \leq 1$ we find that

$$\frac{\sigma'^2}{m^\varepsilon}\mathcal{C}\left(\frac{Mm^\varepsilon}{\sigma'^2}\right) \geq \frac{\sigma'}{\sqrt{8D\gamma\log m}}\log\left(\frac{m^\varepsilon}{\sigma'\sqrt{2D\gamma\log m}}+1\right) \geq \frac{m^\varepsilon}{8D\gamma\log m}.$$

Again it holds that if m is greater than some constant, this is at least $\frac{(\gamma+2)\log m}{D'}$. It follows that if C is large enough, then $\Pr[|\mathcal{V}(i,j)| \geq M] = O(|A|m^{-\gamma-2})$. Union bounding over $(i,j) \in [m] \times ([m] \setminus [m^\varepsilon])$ we find that $\Pr[\mathcal{A}^c] = O(|A|m^{-\gamma})$. Combining the bounds we find that

$$\Pr[|V'| \geq E't] \leq 6\exp\left(-\sigma'^2 h\left(\frac{t}{\sigma'^2}\right)\right) + O(|A|m^{-\gamma}),$$

which completes the proof. $\qquad\square$

6 Extending the Hash Range

This section is dedicated to proving Theorem 6, which we will restate shortly. Again, we will postpone the argument that our concentration bounds are query invariant to Section 7. First, we prove the following technical lemma.

Lemma 20. *Let $\sigma^2 > 0$ and $t > 0$. Writing $s = \max\left\{\sigma^2, \sqrt{t\sigma^2}\right\}$,*

$$s \cdot \mathcal{C}\left(\frac{t}{s}\right) \geq \sigma^2 \mathcal{C}\left(\frac{t}{\sigma^2}\right)/4 .$$

Proof. For $t \leq \sigma^2$ the inequality is trivial, so suppose $t > \sigma^2$. We note that for $x \geq 0$, $1 + \sqrt{x} \geq \sqrt{1+x}$, such that $\lg(1 + \sqrt{x}) \geq \lg(1+x)/2$ for every $x \geq 0$. Using this fact in between two applications of Lemma 9, we find that

$$\sqrt{t\sigma^2}\mathcal{C}\left(\frac{t}{\sqrt{t\sigma^2}}\right) \geq t\lg\left(1 + \sqrt{\frac{t}{\sigma^2}}\right)/2 \geq t\lg\left(1 + \frac{t}{\sigma^2}\right)/4 \geq \sigma^2\mathcal{C}\left(\frac{t}{\sigma^2}\right)/4 \,.$$

$\square$

Next, we recall the law of total variance.

Lemma 21 (Law of Total Variance). *For every pair of random variables X, Y,*

$$\mathrm{Var}\,[Y] = \mathbb{E}\,[\mathrm{Var}\,[Y \mid X]] + \mathrm{Var}\,[\mathbb{E}\,[Y \mid X]] \,.$$

In particular, $\mathrm{Var}\,[Y] \geq \mathrm{Var}\,[\mathbb{E}\,[Y \mid X]]$.

We are now ready to prove the main theorem of the section, which informally states that concatenating the output values of hash functions preserves the property of having Chernoff-style bounds. Note that the following is a much more explicit and elaborate statement of Theorem 6. The purpose of this restatement is to make a formal proof more readable. The reader is encouraged to refer back to Theorem 6 for intuition regarding the theorem statement. Again, we highlight that we have left out the part of Theorem 6 concerning query independence. How query independence is obtained will be discussed in Section 7

Theorem 22. *Let A be a finite set. Let $(X_a)_{a\in A}$ and $(Y_a)_{a\in A}$ be pairwise independent families of random variables taking values in B_X and B_Y, respectively, and satisfying that the distributions of $(X_a)_{a\in A}$ and $(Y_a)_{a\in A}$ are independent. Suppose that there exist universal constants $D_X, D_Y \geq 1$, $K_X, K_Y > 0$, and $R_X, R_Y \geq 0$ such that for every choice of value functions $v_X \colon A \times B_X \to [0,1]$ and $v_Y \colon A \times B_Y \to [0,1]$ and for every $t > 0$,*

$$\Pr\left[\left|\sum_{a\in A} v_X(a, X_a) - \mu_X\right| > t\right] < K_X \exp\left(-\sigma_X^2 \mathcal{C}\left(\frac{t}{\sigma_X^2}\right)/D_X\right) + R_X \,, \tag{25}$$

$$\Pr\left[\left|\sum_{a\in A} v_Y(a, Y_a) - \mu_Y\right| > t\right] < K_Y \exp\left(-\sigma_Y^2 \mathcal{C}\left(\frac{t}{\sigma_Y^2}\right)/D_Y\right) + R_Y \,. \tag{26}$$

where $\mu_X = \mathbb{E}\left[\sum_{a\in A} v_X(a, X_a)\right]$, $\mu_Y = \mathbb{E}\left[\sum_{a\in A} v_Y(a, Y_a)\right]$, $\sigma_X^2 = \mathrm{Var}\left[\sum_{a\in A} v_X(a, X_a)\right]$, and $\sigma_Y^2 = \mathrm{Var}\left[\sum_{a\in A} v_Y(a, Y_a)\right]$. Then for every value function $\overline{v} \colon A \times B_X \times B_Y \to [0,1]$ and every $t > 0$,

$$\Pr\left[\left|\sum_{a\in A} \overline{v}(a, X_a, Y_a) - \mu_{XY}\right| > t\right] < K_{XY} \exp\left(-\sigma_{XY}^2 \mathcal{C}\left(\frac{t}{\sigma_{XY}^2}\right)/D_{XY}\right) + R_{XY} \,,$$

where $\mu_{XY} = \mathbb{E}\left[\sum_{a\in A} \overline{v}(a, X_a, Y_a)\right]$, $\sigma_{XY}^2 = \mathrm{Var}\left[\sum_{a\in A} \overline{v}(a, X_a, Y_a)\right]$, $D_{XY} = \max\{144D_X, 72D_Y\}$, $K_{XY} = 3K_X + K_Y$, and $R_{XY} = 3R_X + R_Y$.

Proof. Let a value function, $\overline{v} \colon A \times B_X \times B_Y \to [0,1]$, and a positive real, $t > 0$, be given. Define $V_a = \overline{v}(a, X_a, Y_a)$, $\mu_a = \mathbb{E}\,[V_a]$, and $\sigma_a^2 = \mathrm{Var}\,[V_a]$. We shall be concerned with the variance of V_a when conditioned on X_a. Hence, we define

$$L_a = \left[\mathrm{Var}\,[V_a \mid X_a] > \sqrt{\frac{6\sigma_{XY}^2}{t}}\right] \quad \text{and} \quad S_a = \left[\mathrm{Var}\,[V_a \mid X_a] \leq \sqrt{\frac{6\sigma_{XY}^2}{t}}\right]$$

42

to be the indicators on the conditional variance of V_a given X_a being larger or smaller, respectively, than the threshold $\sqrt{\frac{6\sigma_{XY}^2}{t}}$. Noting that $L_a + S_a = 1$, we split the sum $\sum_{a \in A}(V_a - \mu_a)$ into three parts.

$$\sum_{a \in A}(V_a - \mu_a) = \underbrace{\sum_{a \in A}(\mathbb{E}\left[V_a \mid X_a\right] - \mu_a)}_{T_1}$$

$$+ \underbrace{\sum_{a \in A} L_a(V_a - \mathbb{E}\left[V_a \mid X_a\right])}_{T_2}$$

$$+ \underbrace{\sum_{a \in A} S_a(V_a - \mathbb{E}\left[V_a \mid X_a\right])}_{T_3}$$

Now, the triangle inequality and a union bound yields

$$\Pr\left[\left|\sum_{a \in A}\bar{v}(a, X_a, Y_a) - \mu_{XY}\right| > t\right] = \Pr\left[\left|\sum_{a \in A}(V_a - \mu_a)\right| > t\right] \leq \sum_{i=1}^{3}\Pr\left[|T_i| > t/3\right].$$

We shall bound each of the three terms T_1, T_2, and T_3 individually.

For bounding $\Pr\left[|T_1| > t/3\right]$, define the value function $v_X^{(1)} : A \times B_X \rightarrow [0, 1]$ by $v_X^{(1)}(a, x) = \mathbb{E}\left[V_a \mid X_a = x\right]$. Note that $\mathbb{E}\left[\mathbb{E}\left[V_a \mid X_a\right]\right] = \mu_a$ and $\mathrm{Var}\left[\mathbb{E}\left[V_a \mid X_a\right]\right] \leq \sigma_a^2$, by the law of total variance, such that $\mathrm{Var}\left[\sum_{a \in A} v_X^{(1)}(a, X_a)\right] \leq \sigma_{XY}^2$. Thus, by Equation (25) and Lemma 10,

$$\Pr\left[|T_1| > t/3\right] = \Pr\left[\sum_{a \in A}\left(v_X^{(1)}(a, X_a) - \mu_a\right) > t/3\right]$$

$$< K_X \exp\left(-\sigma_{XY}^2 \mathcal{C}\left(\frac{t/3}{\sigma_{XY}^2}\right)/D_X\right) + R_X$$

$$\leq K_X \exp\left(-\sigma_{XY}^2 \mathcal{C}\left(\frac{t}{\sigma_{XY}^2}\right)/(9D_X)\right) + R_X .$$

For bounding $\Pr\left[|T_2| > t/3\right]$, we may assume that $t > 6\sigma_{XY}^2$ since otherwise $T_2 = 0$ almost surely. Now, recall that $L_a = \left[\mathrm{Var}\left[V_a \mid X_a\right] > \sqrt{6\sigma_{XY}^2/t}\right]$ and write $Z = \sum_{a \in A} L_a$. We observe that since $V_a \in [0, 1]$ almost surely, $Z \geq |T_2|$ almost surely. By the law of total variance, $\mathbb{E}\left[\mathrm{Var}\left[V_a \mid X_a\right]\right] \leq \sigma_a^2$, so by Markov's inequality,

$$\mathbb{E}\left[L_a\right] = \Pr\left[\mathrm{Var}\left[V_a \mid X_a\right] > \sqrt{\frac{6\sigma_{XY}^2}{t}}\right] \leq \sigma_a^2 \sqrt{\frac{t}{6\sigma_{XY}^2}}.$$

Now, $\mathrm{Var}\left[L_a\right] \leq \mathbb{E}\left[L_a\right] \leq \sigma_a^2 \sqrt{t/(6\sigma_{XY}^2)}$ as $L_a \in [0, 1]$ almost surely. Thus, $\mathbb{E}\left[Z\right] \leq \sqrt{t\sigma_{XY}^2/6}$ and $\mathrm{Var}\left[Z\right] \leq \sqrt{t\sigma_{XY}^2/6}$. Combining this with $t > 6\sigma_{XY}^2$, we may write

$$\Pr\left[|T_2| > t/3\right] \leq \Pr\left[Z - \mathbb{E}\left[Z\right] > t/3 - \sqrt{t\sigma_{XY}^2/6}\right] \leq \Pr\left[|Z - \mathbb{E}\left[Z\right]| > t/6\right].$$

Applying Equation (25) with the value function $v_X^{(2)} : A \times B_X \rightarrow [0, 1]$ given by $v_X^{(2)}(a, X_a) = L_a$ to

$\Pr\left[\left|Z - \mathbb{E}\left[Z\right]\right| > t/6\right]$ yields

$$\Pr\left[\left|T_2\right| > t/3\right] < K_X \exp\left(-\sqrt{t\sigma_{XY}^2/6} \cdot \mathcal{C}\left(\frac{t/6}{\sqrt{t\sigma_{XY}^2/6}}\right)/D_X\right) + R_X$$

$$\leq K_X \exp\left(-\sigma_{XY}^2 \mathcal{C}\left(\frac{t/6}{\sigma_{XY}^2}\right)/(4D_X)\right) + R_X$$

$$\leq K_X \exp\left(-\sigma_{XY}^2 \mathcal{C}\left(\frac{t}{\sigma_{XY}^2}\right)/(144 \cdot D_X)\right) + R_X,$$

where the second follows from Lemma 20 and the third inequality follows from Lemma 10.

Lastly, we shall bound $\Pr\left[\left|T_3\right| > t/3\right]$. By a union bound,

$$\Pr\left[\left|T_3\right| > t/3\right] \leq \underbrace{\Pr\left[\left(\left|T_3\right| > t/3\right) \wedge \left(\mathrm{Var}\left[T_3 \mid (X_a)_{a \in A}\right] \leq 2\max\left\{\sigma_{XY}^2, \sqrt{t\sigma_{XY}^2}\right\}\right)\right]}_{R_1}$$

$$+ \underbrace{\Pr\left[\mathrm{Var}\left[T_3 \mid (X_a)_{a \in A}\right] > 2\max\left\{\sigma_{XY}^2, \sqrt{t\sigma_{XY}^2}\right\}\right]}_{R_2} \cdot$$

First, we bound R_1. For each $a \in A$, let $x_a \in B_X$ be given such that $P(\forall a \in A\colon X_a = x_a) > 0$. We bound the probability of R_1 conditioned on $(X_a = x_a)_{a \in A}$ and since our bound will be the same for every choice of $(x_a)_{a \in A}$, the bound will hold unconditionally. Now, if $\mathrm{Var}\left[T_3 \mid (X_a = x_a)_{a \in A}\right] > 2\max\left\{\sigma_{XY}^2, \sqrt{t\sigma_{XY}^2}\right\}$, then $R_1 = 0$. So assume otherwise and define the value function $v_Y^{(1)}\colon A \times B_Y \to [0,1]$ by $v_Y^{(1)}(a,y) = S_a \cdot \overline{v}(a, x_a, y)$, where $S_a = \left[\mathrm{Var}\left[V_a \mid X_a = x_a\right] \leq \sqrt{6\sigma_{XY}^2/t}\right]$. Then $T_3 = \sum_{a \in A}\left(v_Y^{(1)}(Y_a) - \mathbb{E}\left[v_Y^{(1)}(Y_a)\right]\right)$ and by assumption, $\mathrm{Var}\left[\sum_{a \in A} v_Y^{(1)}(a, Y_a)\right] \leq 2\max\left\{\sigma_{XY}^2, \sqrt{t\sigma_{XY}^2}\right\}$. Thus, we may apply Equation (26) with $v_Y^{(1)}$ to obtain

$$\Pr\left[\left(\left|T_3\right| > t/3\right) \wedge \left(\mathrm{Var}\left[T_3 \mid (X_a)_{a \in A}\right] \leq 2\max\left\{\sigma_{XY}^2, \sqrt{t\sigma_{XY}^2}\right\}\right) \,\middle|\, (X_a = x_a)_{a \in A}\right]$$

$$\leq K_Y \exp\left(-2\max\left\{\sigma_{XY}^2, \sqrt{t\sigma_{XY}^2}\right\} \mathcal{C}\left(\frac{t/3}{2\max\left\{\sigma_{XY}^2, \sqrt{t\sigma_{XY}^2}\right\}}\right)/D_Y\right) + R_Y$$

$$\leq K_Y \exp\left(-\max\left\{\sigma_{XY}^2, \sqrt{t\sigma_{XY}^2}\right\} \mathcal{C}\left(\frac{t}{\max\left\{\sigma_{XY}^2, \sqrt{t\sigma_{XY}^2}\right\}}\right)/(18D_Y)\right) + R_Y$$

$$\leq K_Y \exp\left(-\sigma_{XY}^2 \mathcal{C}\left(\frac{t}{\sigma_{XY}^2}\right)/(72D_Y)\right) + R_Y,$$

where the second follows from Lemma 10 and the third inequality follows from Lemma 20. In conclusion,

$$R_1 \leq K_Y \exp\left(-\sigma_{XY}^2 \mathcal{C}\left(\frac{t}{\sigma_{XY}^2}\right)/(72D_Y)\right) + R_Y.$$

Second, we bound R_2. Define the value function $v_X^{(3)}\colon A \times B_X \to [0,1]$ by

$$v_X^{(3)}(a, x_a) = \left[\mathrm{Var}\left[V_a \mid X_a = x_a\right] \leq \sqrt{\frac{6\sigma_{XY}^2}{t}}\right] \cdot \mathrm{Var}\left[V_a \mid X_a = x_a\right].$$

Then $\mathrm{Var}\,[T_3 \mid (X_a)_{a \in A}] = \sum_{a \in A} v_X^{(3)}(a, X_a)$. Now, by the law of total variance,

$$\mathbb{E}\,[\mathrm{Var}\,[T_3 \mid (X_a)_{a \in A}]] \leq \mathrm{Var}\,[T_3] \leq \sigma_{XY}^2,$$

and since $\sqrt{\frac{t}{6\sigma_{XY}^2}} v_X^{(3)}(X_a) \in [0, 1]$ almost surely for every $a \in A$, pairwise independence yields

$$\mathrm{Var}\left[\sqrt{\frac{t}{6\sigma_{XY}^2}} \mathrm{Var}\,[T_3 \mid (X_a)_{a \in A}]\right] \leq \mathbb{E}\left[\sqrt{\frac{t}{6\sigma_{XY}^2}} \mathrm{Var}\,[T_3 \mid (X_a)_{a \in A}]\right] \leq \sqrt{t\sigma_{XY}^2/6}\,.$$

Applying Equation (25) with $v_X^{(3)}$, Lemma 20, and Lemma 10, we obtain

$$R_2 \leq \Pr\left[\left|\mathrm{Var}\,[T_3 \mid (X_a)_{a \in A}] - \mathbb{E}\,[\mathrm{Var}\,[T_3 \mid (X_a)_{a \in A}]]\right| > \max\left\{\sigma_{XY}^2, \sqrt{t\sigma_{XY}^2}\right\}\right]$$

$$= \Pr\left[\sqrt{\frac{t}{6\sigma_{XY}^2}}\left|\mathrm{Var}\,[T_3 \mid (X_a)_{a \in A}] - \mathbb{E}\,[\mathrm{Var}\,[T_3 \mid (X_a)_{a \in A}]]\right| > \max\left\{\sqrt{t\sigma_{XY}^2/6}, t/6\right\}\right]$$

$$\leq \Pr\left[\sqrt{\frac{t}{6\sigma_{XY}^2}}\left|\mathrm{Var}\,[T_3 \mid (X_a)_{a \in A}] - \mathbb{E}\,[\mathrm{Var}\,[T_3 \mid (X_a)_{a \in A}]]\right| > t/6\right]$$

$$< K_X \exp\left(-\sqrt{t\sigma_{XY}^2/6}\,\mathcal{C}\left(\frac{t/6}{\sqrt{t\sigma_{XY}^2/6}}\right)/D_X\right) + R_X$$

$$\leq K_X \exp\left(-\sigma_{XY}^2 \mathcal{C}\left(\frac{t/6}{\sigma_{XY}^2}\right)/(4D_X)\right) + R_X$$

$$\leq K_X \exp\left(-\sigma_{XY}^2 \mathcal{C}\left(\frac{t}{\sigma_{XY}^2}\right)/(144D_X)\right) + R_X.$$

Combining the bounds on $\Pr\,[|T_i| > t/3]$ for $i \in \{1, 2, 3\}$ completes the proof. $\qquad\square$

7 Query invariance

In the following, we will briefly explain for each of the main sections of the paper, why all theorems still hold when adding the condition of query invariance of Definition 2. Recall that query invariance comes into play when we have a hash function and a concentration bound in the following manner. The concentration bound is query invariant if for any hash key q, a *query key*, the concentration bound still holds whenever we condition the hash function on the hash value of q.

Simple Tabulation Hashing. In [38] it is observed that ordering the position characters $\alpha_1 \prec \cdots \prec \alpha_r$ such that $\alpha_1, \ldots, \alpha_c$ are the position characters of the query key q only worsens the bound on the groups, G_i, by a factor of 2. We consider a slightly more general case, but exactly the same argument still applies. Always imposing this ordering in our proofs lets us condition on the hash value of q and only causes some of the constants to increase by a small factor.

Tabulation-Permutation In the proof of Theorem 5 we consider some specific value function w. We proceed by considering separately the m^ε bins $S \subset [m]$ of largest contribution to the variance, σ^2, and then the remaining bins, $[m] \setminus S$. The contribution of each subset of bins is then individually bounded. In the first case, we simply use the assumption on the hash function h that we received in a black box manner and use no properties of the permutation. Now, say towards query invariance that we require that $\tau \circ h(q) = i$. To support this, we instead chose S to have have size $|S| = m^\varepsilon/2$. This does not change the proof by more than constant factors and simply adding i to S yields a set $S' = S \cup \{i\}$ of size $S' < m^\varepsilon$, such that the assumption on h directly yields the result. In conclusion, the proof goes through exactly as before.

Extending the Codomain In this section nothing in the proof requires us to take into special consideration the conditioning on a query key. We simply consider families of hash functions in a black box manner and thus, we may as well consider families that have already been condition on the hash value of the query key q.

8 Tightness of Concentration: Simple Tabulation into Few Bins

Recall the result of Theorem 1. If $h\colon [u] \to [m]$ is a simple tabulation hash function with $[u] = \Sigma^c$ and $c = O(1)$, and $S \subseteq [u]$ is a set of hash keys of size $n = |S|$ where each key $x \in S$ is given a weight $w_x \in [0,1]$. Then for arbitrary $y \in [m]$ and a constant $\gamma > 0$ the total weight of the balls landing in bin y, given by the random variable $X = \sum_{x \in S} w_x[h(x) = y]$, satisfies the concentration bound

$$\Pr\left[|X - \mu| \ge t\right] \le 2\exp(-\Omega(\sigma^2 \mathcal{C}(t/\sigma^2))) + n/m^\gamma, \tag{27}$$

where $\mu = \mathbb{E}[X]$ and $\sigma^2 = \mathrm{Var}[X]$ are the expectation and variance, respectively, of X, and the constant in the Ω-notation depends on γ. As mentioned in the introduction, the added error probability n/m^γ renders the theorem nearly useless for small m, the prime example being the tossing of an unbiased coin corresponding to $m = 2$. The purpose of this section is to show that the bound of (27) is optimal in the sense that an added error probability of at least $m^{-\gamma}$ for some constant γ is inevitable so long as we insist on strong concentration according to Definition 1. In other words, we must accept an added error probability of $m^{-\gamma}$ to have Chernoff-style bounds on the sum X. In fact, it will turn out that unless allowing an error term of the form $m^{-\gamma}$, the deviation from the case of a fully random hash function can be quite significant.

The example where simple tabulation does not concentrate well, which we shall use in the formal proof below, is the following. For some $k < |\Sigma|$, we consider the key set $S = [k]^{c-1} \times \Sigma \subset \Sigma^c$ with weights $w_x = 1$ for every $x \in S$. We shall think of k as slightly superconstant and mutually dependent on γ. Recall that h is defined by c fully random functions $h_0, \ldots, h_{c-1}\colon \Sigma \to [m]$ and that $h(x) = \bigoplus_{i=0}^c h_i(x_i)$. With probability $m^{-(k-1)(c-1)}$, h_i is constant on $[k]$ for each $0 \le i \le c-2$. Under such a *collapse* it holds for every $\alpha \in \Sigma$ that every key from the set $[k]^{c-1} \times \{\alpha\}$ hashes to the same value in $[m]$ under h. Hence, each entry of h_{c-1} decides where k^{c-1} keys hash to. Thus, during such a collapse, we may view the hashing of S into $[m]$ as throwing $|\Sigma|$ balls each of weight k^{c-1} into m bins. This increases the variance by a factor of k^{c-1} affecting the Chernoff bounds accordingly.

Without further ado, let us present the formal statement of the above. Essentially, it states that there is a *delay* of the exponential decrease which depends on γ. If γ is superconstant, so is the delay, and hence, we do not have strong concentration according to Definition 1.

Theorem 23. *Let $m \le |\Sigma|^{1-\varepsilon}$ for some constant $\varepsilon > 0$ and $h\colon [u] \to [m]$ be a simple tabulation hash function. Let $0 < \varepsilon' < \varepsilon$ be a constant and suppose that $C\colon \mathbb{R}^+ \to \mathbb{R}^+$ is a function such that for all $0 \le \gamma \le |\Sigma|^{\varepsilon'/c}$, all sets $S \subseteq [u]$, all choices of weights $w_x \in [0,1], x \in S$, and every $y \in [m]$, the random variable $X = \sum_{x \in S} w_x[h(x) = y]$ satisfies*

$$\Pr[|X - \mu| \ge t] \le 2\exp\left(\frac{-\sigma^2 \mathcal{C}(t/\sigma^2)}{C(\gamma)}\right) + m^{-\gamma} \tag{28}$$

for all $t > 0$. Then $C(\gamma) = \Omega(\gamma^{c-2})$.

Proof. Assume the existence of the function C. As suggested above, consider the set of keys $S = [k]^{c-1} \times \Sigma$ for some k to be determined. Denote by $\mathcal{E}$ the event that the first $c-1$ position characters of S collapse, i.e., that each $h_i, 0 \le i \le c-2$ is constant on $[k]$. We easily calculate $\Pr[\mathcal{E}] = m^{-(k-1)(c-1)}$. Now, conditioning on $\mathcal{E}$, we may consider the situation as follows. Let y' be the random variable satisfying $\bigoplus_{i=0}^{c-2} h_i(x_i) = y'$ for all $x_0, \ldots, x_{c-2} \in [k]$. The last positional hash function h_{c-1} is a fully random hash function $\Sigma \to m$ such that the conditioned variable $(X|\mathcal{E})$ satisfies

$$(X|\mathcal{E}) = \sum_{\alpha \in \pi_{c-1}(S)} k^{c-1}[h_{c-1}(\alpha) = y \oplus y'] \stackrel{d}{=} \sum_{\alpha \in \Sigma} k^{c-1}[h_{c-1}(\alpha) = 0] =: X',$$

46

where $\overset{d}{=}$ denotes equality of distribution. We write $\sigma'^2 = \text{Var}\,[X'] = k^{2(c-1)}\,|\Sigma|\,\frac{m-1}{m^2}$ and note that $\mathbb{E}\,[X'] = \mu$. Now, since h_{c-1} is a uniformly random hash function, tightness of the Bennet inequality, Eq. (3), implies that for $t = O(\sigma'^2)$,

$$\Pr\left[|X' - \mu| \geq t\right] = \Omega\left(\exp\left(-\sigma'^2 \mathcal{C}(t/\sigma'^2)\right)\right) = \Omega\left(\exp\left(-\frac{t^2}{3\sigma'^2}\right)\right) \tag{29}$$

where we have applied Lemma 9.

Towards our main conclusion, observe that $\sigma^2 = k^{c-1}\,|\Sigma|\,\frac{m-1}{m^2} = \sigma'^2/k^{c-1}$. Letting $t = \sigma'\sqrt{\log m}$, $t \leq \sigma'^2$ since $\sigma' > \sqrt{\log m}$ by the assumption on the size of m, so we may apply (29) to get

$$\Pr\left[|X - \mu| \geq t\right] \geq \Pr\left[\mathcal{E}\right] \cdot \Pr\left[|X' - \mu| \geq t\right] \geq \Omega\left(m^{-ck}\right).$$

On the other hand, (28) demands that whenever $k \leq \gamma$,

$$\Pr\left[|X - \mu| \geq t\right] \leq 2\exp\left(-\frac{\sigma^2 \mathcal{C}\left(\sqrt{\log(m)k^{c-1}}/\sigma\right)}{C(\gamma)}\right) + m^{-\gamma} \leq 2m^{-\frac{k^{c-1}}{C(\gamma)}} + m^{-\gamma},$$

where the last inequality used Lemma 9 and $\sqrt{\log(m)k^{c-1}} < \sigma$. Let $k = \frac{\gamma}{2c}$ and combine the above inequalities to conclude that $2m^{-\frac{(\gamma/(2c))^{c-1}}{C(\gamma)}} + m^{-\gamma} = \Omega(m^{-\gamma/2})$. It follows that indeed, $C(\gamma) = \Omega(\gamma^{c-2})$. $\square$

Twisted Tabulation and "permutation-tabulation" Variations upon the example above can also be used to show that the analysis of twisted tabulation hashing is tight in the sense that the added error probability cannot be improved while maintaining strong concentration. In twisted tabulation we *twist* the last position character of the input before applying simple tabulation. The twist is a Feistel permutation that for the key set $S = [k]^b \times \Sigma^{c-b}$, will only permute the keys within the set. Since the set of twisted keys is the same as the original set S, this has no effect on the filling of bins. For almost the same reason, a reversal of the order of operations in our new tabulation-permutation hashing, i.e., if is first permuted each position character and the applied simple tabulation, would not improve the analysis, since the set S while not invariant under the operation, would retain the same structure.

Acknowledgement

References

[1] AAMAND, A., DAS, D., KIPOURIDIS, E., KNUDSEN, J. B. T., RASMUSSEN, P. M. R., AND THORUP, M. No repetition: Fast streaming with highly concentrated hashing. *CoRR* (2020). arxiv.org/abs/2004.01156.

[2] ANDERSSON, A., MILTERSEN, P. B., RIIS, S., AND THORUP, M. Static dictionaries on AC^0 RAMs: Query time $\Theta(\sqrt{\log n/\log\log n})$ is necessary and sufficient. In *37th Annual Symposium on Foundations of Computer Science (FOCS)* (1996), pp. 441–450.

[3] APPLEBY, A. MurmurHash3. https://github.com/aappleby/smhasher/wiki/MurmurHash3, 2016.

[4] Aumasson, J.-P., Neves, S., Wilcox-O'Hearn, Z., and Winnerlein, C. Blake2: Simpler, smaller, fast as MD5. In *Applied Cryptography and Network Security* (Berlin, Heidelberg, 2013), M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, Eds., Springer Berlin Heidelberg, pp. 119–135.

[5] Bar-Yossef, Z., Jayram, T. S., Kumar, R., Sivakumar, D., and Trevisan, L. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)* (2002), pp. 1–10.

[6] Bennett, G. Probability inequalities for the sum of independent random variables. *Journal of the American Statistical Association 57*, 297 (1962), 33–45.

[7] Bernstein, S. N. On a modification of Chebyshev's inequality and of the error formula of Laplace. *Ann. Sci. Inst. Sav. Ukraine, Sect. Math.*, 1 (1924), 38–49.

[8] Broder, A. Z. On the resemblance and containment of documents. In *Compression and Complexity of Sequences (SEQUENCES)* (1997), pp. 21–29.

[9] Carter, L., and Wegman, M. N. Universal classes of hash functions. *Journal of Computer and System Sciences 18*, 2 (1979), 143–154. Announced at STOC'77.

[10] Celis, L. E., Reingold, O., Segev, G., and Wieder, U. Balls and bins: Smaller hash families and faster evaluation. In *52nd Annual Symposium on Foundations of Computer Science (FOCS)* (2011), pp. 599–608.

[11] Chandra, A. K., Stockmeyer, L. J., and Vishkin, U. Constant depth reducibility. *SIAM J. Comput. 13*, 2 (1984), 423–439.

[12] Chernoff, H. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics 23*, 4 (1952), 493–507.

[13] Christiani, T., and Pagh, R. Generating k-independent variables in constant time. In *55th Annual Symposium on Foundations of Computer Science (FOCS)* (2014), pp. 196–205.

[14] Christiani, T., Pagh, R., and Thorup, M. From independence to expansion and back again. In *Proceedings of the 47rd ACM Symposium on Theory of Computing (STOC)* (2015).

[15] Chung, K.-M., Mitzenmacher, M., and Vadhan, S. Why simple hash functions work: Exploiting the entropy in a data stream. *Theory of Computing 9*, 1 (2013), 897–945.

[16] Dahlgaard, S., Knudsen, M. B. T., Rotenberg, E., and Thorup, M. Hashing for statistics over k-partitions. In *56th Annual Symposium on Foundations of Computer Science (FOCS)* (2015), pp. 1292–1310.

[17] Dahlgaard, S., Knudsen, M. B. T., and Thorup, M. Practical hash functions for similarity estimation and dimensionality reduction. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)* (2017), Curran Associates Inc., pp. 6618–6628.

[18] Dietzfelbinger, M. Universal hashing and k-wise independent random variables via integer arithmetic without primes. In *Proceedings of the 13th Symposium on Theoretical Aspects of Computer Science (STACS)* (1996), pp. 569–580.

[19] Dietzfelbinger, M., and Meyer auf der Heide, F. Dynamic hashing in real time. In *Informatik, Festschrift zum 60. Geburtstag von Günter Hotz.* 1992, pp. 95–119.

[20] Dietzfelbinger, M., and Rink, M. Applications of a splitting trick. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)* (2009), pp. 354–365.

[21] DIETZFELBINGER, M., AND WEIDLING, C. Balanced allocation and dictionaries with tightly packed constant size bins. *Theor. Comput. Sci. 380* (06 2007), 47–68.

[22] DIETZFELBINGER, M., AND WOELFEL, P. Almost random graphs with simple hash functions. In *Proceedings of the 25th ACM Symposium on Theory of Computing (STOC)* (2003), pp. 629–638.

[23] DUMEY, A. I. Indexing for rapid random access memory systems. *Computers and Automation 5*, 12 (1956), 6–9.

[24] FAN, X., GRAMA, I., AND LIU, Q. Hoeffding's inequality for supermartingales. *Stochastic Processes and their Applications 122*, 10 (2012), 3545 – 3559.

[25] FOTAKIS, D., PAGH, R., SANDERS, P., AND SPIRAKIS, P. Space efficient hash tables with worst case constant access time. *Theory of Computing Systems 38*, 2 (Feb 2005), 229–248.

[26] GOPALAN, P., KANE, D. M., AND MEKA, R. Pseudorandomness via the discrete fourier transform. *SIAM J. Comput. 47*, 6 (2018), 2451–2487.

[27] HAGERUP, T., AND THOLEY, T. Efficient minimal perfect hashing in nearly minimal space. In *Proceedings of the 18th Symposium on Theoretical Aspects of Computer Science (STACS)* (2001), pp. 317–326.

[28] HENNESSY, J. L., AND PATTERSON, D. A. *Computer Architecture - A Quantitative Approach, 5th Edition*. Morgan Kaufmann, 2012.

[29] KNUTH, D. E. Notes on open addressing. Unpublished memorandum. See http://citeseer.ist.psu.edu/knuth63notes.html, 1963.

[30] KRISHNAMURTHY, B., SEN, S., ZHANG, Y., AND CHEN, Y. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of the 3rd Internet Measurement Conference (IMC)* (2003), pp. 234–247.

[31] LEMIRE, D., AND KASER, O. Faster 64-bit universal hashing using carry-less multiplications. *Journal of Cryptographic Engineering 6*, 3 (Sep 2016), 171–185.

[32] MANSOUR, Y., NISAN, N., AND TIWARI, P. The computational complexity of universal hashing. *Theor. Comput. Sci. 107*, 1 (1993), 121–133.

[33] MEKA, R., REINGOLD, O., ROTHBLUM, G. N., AND ROTHBLUM, R. D. Fast pseudorandomness for independence and load balancing - (extended abstract). In *Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP)* (2014), pp. 859–870.

[34] MILTERSEN, P. B. Lower bounds for static dictionaries on rams with bit operations but no multiplication. In *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming (ICALP)* (1996), pp. 442–453.

[35] MITZENMACHER, M., AND VADHAN, S. P. Why simple hash functions work: exploiting the entropy in a data stream. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2008), pp. 746–755.

[36] MOTWANI, R., AND RAGHAVAN, P. *Randomized Algorithms*. Cambridge University Press, 1995.

[37] PAGH, A., AND PAGH, R. Uniform hashing in constant time and optimal space. *SIAM Journal on Computing 38*, 1 (2008), 85–96.

[38] PĂTRAŞCU, M., AND THORUP, M. The power of simple tabulation-based hashing. *Journal of the ACM 59*, 3 (2012), Article 14. Announced at STOC'11.

[39] PĂTRAŞCU, M., AND THORUP, M. On the k-independence required by linear probing and minwise independence. *ACM Trans. Algorithms 12*, 1 (2016), 8:1–8:27.

[40] PIKE, G., AND ALAKUIJALA, J. Introducing cityhash. `https://opensource.googleblog.com/2011/04/introducing-cityhash.html`, 2011.

[41] PĂTRAŞCU, M., AND THORUP, M. Twisted tabulation hashing. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2013), pp. 209–228.

[42] SCHILLING, R. L. *Measures, Integrals and Martingales*. Cambridge University Press, 2005.

[43] SCHMIDT, J. P., SIEGEL, A., AND SRINIVASAN, A. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics 8*, 2 (1995), 223–250. Announced at SODA'93.

[44] SIEGEL, A. On universal classes of extremely random constant-time hash functions. *SIAM Journal on Computing 33*, 3 (2004), 505–543. Announced at FOCS'89.

[45] THORUP, M. Simple tabulation, fast expanders, double tabulation, and high independence. In *54th Annual Symposium on Foundations of Computer Science (FOCS)* (2013), pp. 90–99.

[46] THORUP, M. High speed hashing for integers and strings. *CoRR* (2015). `arxiv.org/abs/1504.06804`.

[47] THORUP, M., AND ZHANG, Y. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM Journal on Computing 41*, 2 (2012), 293–331. Announced at SODA'04 and ALENEX'10.

[48] WEGMAN, M. N., AND CARTER, L. New classes and applications of hash functions. *Journal of Computer and System Sciences 22*, 3 (1981), 265–279. Announced at FOCS'79.

[49] ZOBRIST, A. L. A new hashing method with application for game playing. Tech. Rep. 88, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1970.

A   Experiments

This section is dedicated to provide further details regarding the timing experiments presented in the introduction in Section 1.7.4. Furthermore, we present experiments which demonstrate concrete bad input sets for several hash functions that do not guarantee strong concentration bounds.

As explained in Section 1.7.4, we ran experiments on various basic hash functions. More precisely, we compared our new hashing schemes tabulation-permutation and tabulation-1permutation with the following hashing schemes: k-independent PolyHash [9], Multiply-Shift [18], simple tabulation [49], twisted tabulation [41], mixed tabulation [16], and double tabulation [45]. We were interested in both the speed of the hash functions involved, and the quality of the output. For our timing experiments we studied the hashing of 32-bit keys to 32-bit hash values, and 64-bit keys to 64-bit hash values. Aside from having strong theoretical guarantees, our experiments show that tabulation-permutation and tabulation-1permutation are very fast in practice.

All experiments are implemented in C++11 using a random seed from `https://www.random.org`. The seed for the tabulation based hashing methods uses a random 100-independent PolyHash function. PolyHash is implemented using the Mersenne primes $p = 2^{61} - 1$ for 32 bits and $p = 2^{89} - 1$ for 64 bits. Furthermore, it has been implemented using Horner's rule, and GCC's 128-bit integers to ensure an efficient implementation. Double tabulation is implemented as described in [45] with $\Sigma = [2^{16}], c = 2, d = 20$.

	Running time (ms)			
	Computer 1		Computer 2	
Hash function	32 bits	64 bits	32 bits	64 bits
Multiply-Shift	4.2	7.5	23.0	36.5
2-Independent PolyHash	14.8	20.0	72.2	107.3
Simple tabulation	13.7	17.8	53.1	55.9
Twisted tabulation	17.2	26.1	65.6	92.5
Mixed tabulation	28.6	68.1	120.1	236.6
Tabulation-1permutation	16.0	19.3	63.8	67.7
Tabulation-permutation	27.3	43.2	118.1	123.6
Double tabulation	1130.1	−	3704.1	−
"Random" (100-Independent PolyHash)	2436.9	3356.8	7416.8	11352.6

Table 3: The time for different hash functions to hash 10^7 keys of length 32 bits and 64 bits, respectively, to ranges of size 32 bits and 64 bits. The experiment was carried out on two computers. The hash functions written in italics are those without general Chernoff-style bounds. Hash functions written in bold are the contributions of this paper. The hash functions in regular font are known to provide Chernoff-style bounds. Note that we were unable to implement double tabulation from 64 bits to 64 bits since the hash tables were too large to fit in memory.

Timing We timed the speed of the hash functions on two different computers. The first computer (Computer 1 in Table 3) has a 2.4 GHz Quad-Core Intel Core i5 processor and 8 GB RAM, and it is running macOS Catalina. The second computer (Computer 2 in Table 3) has 1.5 GHz Intel Core i3 processor and 4 GB RAM, and it is running Windows 10. We restate the results of our experiments in Table 3 and refer the reader to Section 1.7.4 for a discussion of these results and of the choice of parameters used in the various hashing schemes.

Quality We will now present experiments with concrete bad instances for the schemes without general concentration bounds, that is, Multiply-Shift, 2-independent PolyHash, simple tabulation, and twisted tabulation. In each case, we compare with our new tabulation-permutation scheme as well as 100-independent PolyHash, which is our approximation to an ideal fully random hash function. We note that all schemes considered are 2-independent, so they all have exactly the same variance as fully-random hashing. From 2-independence, it also follows that the schemes work perfectly on sufficiently random input [35]. Our concern is therefore concrete inputs making them fail in the tail.

First, we consider simple bad instances for Multiply-Shift and 2-independent PolyHash. These are analyzed in detail in [39, Appendix B]. The specific instance we consider is that of hashing the arithmetic progression $A = \{a \cdot i \mid i \in [50000]\}$ into 16 bins, where we are interested in the number of keys from A that hashes to a specific bin. We performed this experiment 5000 times, with independently chosen hash functions. The cumulative distribution functions on the number of keys from A hashing to a specific bin is presented in Figure 2. We see that most of the time 2-independent PolyHash and Multiply-Shift distribute the keys perfectly with exactly 1/16 of the keys in our bin. Since the variance is the same as with fully random hashing, this should suggest a much heavier tail, which is indeed what our experiments show. For contrast, we see that the cumulative distribution function with our tabulation-permutation hash function is almost indistinguishable from that of 100-independent Poly-Hash. We note that our experiments with tabulation-permutation is only a sanity check: No experiment can prove good performance on all possible inputs.

Our second set of experiments shows bad instances for simple tabulation and twisted tabulation. We already know theoretically from Section 8 that these bad instances exist, but we shall now see that, in a sense, things can be even worse than described in Section 8 for certain sets of keys. The specific instance we consider is hashing the discrete cube $Q = [2]^7 \times [2^6]$ to $m = 2$ bins using simple tabulation, twisted
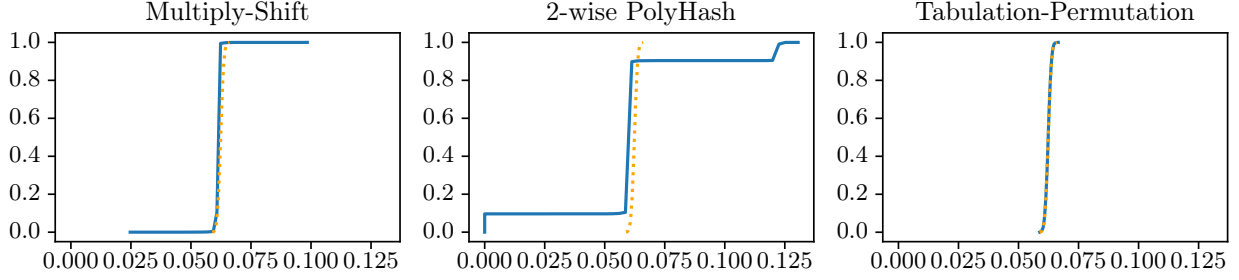
Figure 2: Hashing the arithmetic progression $\{a \cdot i \mid i \in [50000]\}$ to 16 bins for a random integer a. The dotted line is a 100-independent PolyHash.
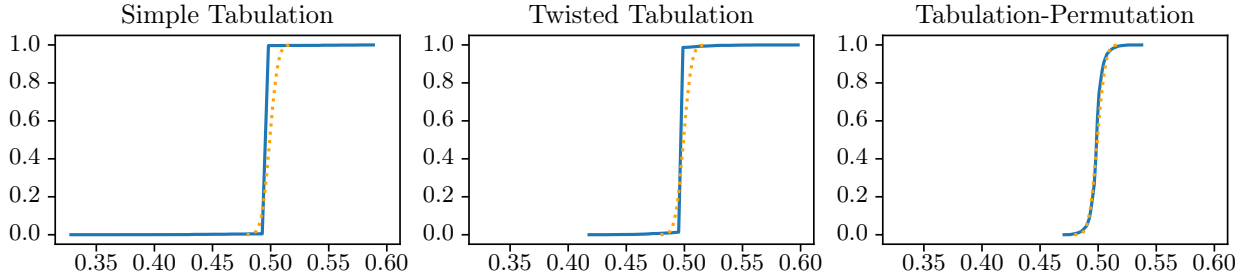

Figure 3: Hashing the discrete cube $[2]^7 \times [2^6]$ to 2 bins. The dotted line is a 100-independent PolyHash.

tabulation, and tabulation-permutation. We performed this experiment 5000 times, with independently chosen hash functions, and again we were interested in the number of keys from Q hashing to one of the bins. The cumulative distribution functions of the number of such keys is presented in Figure 3. Let us explain the appearance of the curves for simple and twisted tabulation. In general, if we hash the set of keys $[2] \times R$ to $[2]$ with simple tabulation, then if $h_1(0) \neq h_1(1)$, each bin will get exactly the same amount of keys. When we hash the set of keys $[2]^7 \times [2^6]$ this happens with probability $1 - 2^{-7}$. If on the other hand $h_i(0) = h_i(1)$ for each $i = 1, \ldots, 7$, which happens with probability 2^{-7}, the distribution of the balls in the bins is the same as that when 2^6 balls, each of weight 2^7, are distributed independently and uniformly at random into the two bins. If this happens, the variance of the number of balls in a bin is a factor of 2^7 higher, so we expect a much heavier tail than in the completely independent case. These observations agree with the results in Figure 3. Most of the time, the distribution is perfect, but the tail is very heavy. We believe that this instance is also one of the worst instances for tabulation-permutation hashing. We would therefore expect to see that on this instance it performs slightly worse than 100-independent PolyHash, which is indeed what our experiments show. We note that that no amount of experimentation can prove that tabulation-permutation always works well for all inputs. We do, however, have mathematical concentration guarantees, and the experiments performed here give us some idea of the impact of the constant delay hidden in the exponential decrease in the bounds of Theorem 2. For completeness, we note that the situation with mixed tabulation is unresolved. Neither do we have strong concentration bounds, nor any bad instances showing that such bounds do not hold. Running experiments is not expected to resolve this issue since mixed tabulation, as any other 2-independent hashing scheme, performs well on almost all inputs [35].