On Risks of Using a High Performance Hashing Scheme With Common Universal Classes

Dissertation zur Erlangung des akademischen Grades
Doctor rerum naturalium (Dr. rer. nat.)

vorgelegt der Fakultät für Informatik und Automatisierung
der Technischen Universität Ilmenau

von

Dipl.-Inform. Ulf Schellbach

Vorgelegt am: 27. März 2009
Verteidigt am: 3. Juli 2009

Gutachter:

1. Univ.-Prof. Dr. rer. nat. (USA) M. Dietzfelbinger, Technische Universität Ilmenau
2. Associate Prof. Rasmus Pagh, IT University of Copenhagen
3. Assistant Prof. Philipp Woelfel, University of Calgary

urn:nbn:de:gbv:ilm1-2009000150

Dedicated to Life

Summary

The contribution of this thesis is a mathematical analysis a high performance hashing scheme called *cuckoo hashing* when combined with two very simple and efficient classes of functions that we refer to as the *multiplicative class* and the *linear class*, respectively. We prove that cuckoo hashing tends to work badly with these classes. In order to show this, we investigate how the inner structure of such functions influences the behavior of the cuckoo scheme when a set S of keys is inserted into initially empty tables.

Cuckoo Hashing uses two tables of size m each. It is known that the insertion of an arbitrary set S of size $n = (1 - \delta)m$ for an arbitrary constant $\delta \in (0, 1)$ (which yields a *load factor* $n/(2m)$ of up to 1/2) fails with probability $O(1/n)$ if the hash functions are chosen from an $\Omega(\log n)$-wise independent class. This leads to the result of expected amortized constant time for a single insertion. In contrast to this we prove lower bounds of the following kind: If S is a uniformly random chosen set of size $n = m/2$ (leading to a load factor of only 1/4 (!)) then the insertion of S fails with probability $\Omega(1)$, or even with probability $1 - o(1)$, if the hash functions are either chosen from the multiplicative or the linear class.

This answers an open question that was already raised by the inventors of cuckoo hashing, Pagh and Rodler, who observed in experiments that cuckoo hashing exhibits a bad behavior when combined with the multiplicative class. Our results implicitly show that the quality of pairwise independence is not sufficient for a hash class to work well with cuckoo hashing. Moreover, our work exemplifies that a recent result of Mitzenmacher and Vadhan, who prove that under certain constraints simple universal functions yield values that are highly independent and very close to uniform random, has to be applied with care: It may not hold if the constraints are not satisfied.

Zusammenfassung

Der Beitrag dieser Dissertation ist die mathematische Analyse eines Hashing-Verfahrens namens *Cuckoo Hashing* in Kombination mit einfachen, effizient auswertbaren Funktionen zweier Hashklassen, die wir die *multiplikative Klasse* bzw. die *lineare Klasse* nennen. Cuckoo Hashing hat die deutliche Tendenz, mit Funktionen dieser beiden Klassen schlecht zu funktionieren. Um dies zu beweisen, untersuchen wir den Einfluss der inneren Struktur solcher Funktionen auf das Verhalten des Cuckoo-Verfahrens, wenn der Versuch unternommen wird, eine Schlüsselmenge S in anfangs leere Tabellen einzufügen.

Cuckoo Hashing verwendet zwei Tabellen der jeweiligen Größe m. Man weiß, dass die Einfügung einer beliebigen Menge S der Größe $n = (1-\delta)m$ für eine beliebige Konstante $\delta \in (0,1)$ (was einen *Lastfaktor* $n/(2m)$ von bis zu $1/2$ liefert) mit Wahrscheinlichkeit $O(1/n)$ fehlschlägt, falls die Hashfunktionen aus einer $\Omega(\log n)$-fach unabhängigen Klasse gewählt werden. Damit läßt sich beweisen, dass die erwarteten amortisierten Kosten einer einzelnen Einfügung konstant sind. Demgegenüber beweisen wir untere Schranken der folgenden Art: Wenn S eine uniform zufällig gewählte Schlüsselmenge der Größe $m/2$ ist (Lastfaktor $1/4$ (!)), dann schlägt die Einfügung von S mit großer Wahrscheinlichkeit $\Omega(1)$, oder sogar $1-o(1)$, fehl, falls Funktionen der multiplikativen bzw. der linearen Klasse gewählt werden.

Damit beantworten wir eine Frage, die bereits von Pagh und Rodler aufgeworfen wurde, als sie mit Hilfe von Experimenten feststellten, dass es riskant ist, Cuckoo Hashing mit Funktionen der multiplikativen Klasse zu kombinieren. Unsere Resultate zeigen, dass paarweise Unabhängigkeit und uniforme Verteilung der Hashwerte keine Garantie dafür ist, dass Cuckoo Hashing gut funktioniert. Zudem machen unsere Ergebnisse exemplarisch deutlich, dass bei der Anwendung eines kürzlich veröffentlichten Resultates von Mitzenmacher und

Vadhan, welches besagt, dass selbst Funktionen aus einfachen, schwach universellen Klassen unter gewissen Bedingungen zu hochgradig unabhängigen und fast uniform zufälligen Werten führen können, Vorsicht geboten ist: Falls dessen Bedingungen nicht erfüllt sind, gilt es unter Umständen nicht.

Acknowledgements

According to my belief, this thesis is an achievement of life: Life has created this thesis through my brain and hands. Life has influenced this process of coming into existence through many people in my near and far surrounding. Therefore, I would like to express my deepest gratitude to life at first.

As soon as I start to mention somebody by name I will surely miss the name of a person who played an important role without that I was aware of it. To all those people it may be a comfort to hear that I happend to forget many things during the past few weeks as I was so focused on the completion of the thesis. However, I am thankful to Martin Dietzfelbinger, my thesis advisor. He guided me with patience, and he was a great source of inspiration for me with respect to scientific work. I also thank all my present and past colleages who amicably accompanied me during the past four years, and who shared their knowledge with me: Manfred Kunde, Michael Brinkmeier, Elke Hübel, Karl-Heinz Niggl, Henning Wunderlich, Folke Eisterlehner, Michael Rink (who bore to share the office with me during the last few weeks of finishing the thesis. . .), Sascha Grau, Lucia Penso, Petra Schüller, and our secretary Jana Kopp (who ceaselessly reminds us of important dates, and does the work that we like the least).

I surely will not forget to thank my parents Edeltraut and Klaus, and my brother Philipp. They supported me in countless ways, with conditionless love.

Finally, I feel deepest gratefulness towards my beloved partner Ellen Stadie, and her children Theresa and Anton. They help me to stand with both feet on the ground.

Now, it just remains to be acknowledged that large parts of this thesis are based on revised versions of [15] and [16].

Contents

CHAPTER 1

Introduction

Sets are without doubt fundamental to mathematics. As for computer science, data structures that represent sets and algorithms that manipulate set data structures are of comparable importance.

Sets manipulated by algorithms can grow, shrink, or otherwise change over time. We call such sets *dynamic*. A dynamic set whose elements are pairs (x, r) of a unique identifier x associated with some satellite data r (such that its *states* can be viewed as *mappings*) is called a *dictionary*.

A multitude of different ways to implement a dictionary is known. Among the most efficient dictionary implementations are *hashing* schemes. Hashing schemes can be characterized by using a *hash table* to store the elements of a set, and making use of a *hash function* to find the location of elements in the table. The performance of such a scheme strongly depends on the choice of the hash function. One can, for example, choose the hash function randomly from a given *class* of functions. This scenario is called *universal hashing*. It is attractive from a practical point of view: There are classes of simple functions that can be represented and evaluated very efficiently such that the combination of a given hashing scheme with such a "universal" class performs well.

The contribution of this thesis is a mathematical analysis of a high performance hashing scheme called *cuckoo hashing* when combined with two very simple and efficient classes of functions that we refer to as the *multiplicative class* and the *linear class*, respectively. We prove that cuckoo hashing tends to work badly with these classes. In order to show this, we investigate how the inner structure of such functions influences the behavior of the cuckoo scheme when a set S of keys is inserted into initially empty tables.

Cuckoo Hashing uses two tables of size m each. It is known that the insertion of an arbitrary set S of size $n = (1 - \delta)m$ for an arbitrary constant $\delta \in (0, 1)$ (which yields a *load factor* $n/(2m)$ of up to 1/2) fails with probability $O(1/m)$ if the hash functions are chosen from an $\Omega(\log n)$-wise independent class. This leads to the result of expected amortized constant time for a single insertion. In contrast to this we prove lower bounds of the following kind: If S is a uniformly random chosen set of size $n = m/2$ (leading to a load factor of only 1/4 (!)) then the insertion of S fails with probability $\Omega(1)$, or even with probability $1 - o(1)$, if the hash functions are either chosen from the multiplicative or the linear class.

Our results answer an open question that was already raised by the inventors of cuckoo hashing, Pagh and Rodler, who observed in experiments that cuckoo hashing exhibits a bad behavior when combined with the multiplicative class. Our results implicitly show that the quality of pairwise independence is not sufficient for a hash class to work well with cuckoo hashing. Moreover, our work exemplifies that a recent result of Mitzenmacher and Vadhan, who prove that under certain constraints simple universal functions yield values that are highly independent and very close to uniform random, has to be applied with care: It may not hold if the constraints are not satisfied.

We hope that our work will in the long term enhance the practical use of cuckoo hashing.

1.1 Background

Roughly following the line drawn above, we introduce the background that is necessary to understand the significance of our results.

1.1.1 The Dictionary Data Type

From a mathematical point of view, the state of a *dictionary* can be modeled as a mapping $f\colon D \to R$ that assigns *satellite data* in a set R to *keys* in a finite subset D of some *universe* U. Recall that according to the mathematical definition of f being a mapping from D to R we have $f \subseteq D \times R$ and for all $x \in D$ there exists a unique $r \in R$ with $(x, r) \in f$. Furthermore, a dictionary supports the following operations that change its state:

1. `empty`. Construction of an initially empty dictionary:

$$\texttt{empty}(()) = \emptyset$$

2. `lookup`. Given a dictionary f and a key x, return the data $f(x)$ that is assigned to x if x is in the domain D:

$$\texttt{lookup}(f, x) := \begin{cases} f(x) & \text{if } x \in D \\ \text{undefined} & \text{otherwise} . \end{cases}$$

3. `delete`. Given a dictionary f and a key x, delete x and the corresponding data $f(x)$ if $x \in D$:

$$\texttt{delete}(f, x) := \begin{cases} f & \text{if } x \notin D \\ f - \{(x, f(x))\} & \text{otherwise} . \end{cases}$$

4. `insert`. Given a dictionary f, a key x, and a data element r, insert (x, r) into f if $x \notin D$, otherwise replace $f(x)$ with r:

$$\texttt{insert}(f, x, r) := \begin{cases} f \cup \{(x, r)\} & \text{if } x \notin D \\ (f - \{(x, f(x))\}) \cup \{(x, r)\} & \text{otherwise} . \end{cases}$$

In this thesis we do not care about how the problem of assigning data r to a key x is practically solved, as this will depend on the range R. We identify pairs (x, r) with the corresponding key x, and assume that the assignment problem is solved efficiently.

1.1.2 Hashing

This thesis analyzes a *hashing* scheme. Hashing is arguably one of the most clever ways to implement a dictionary: Given a *universe* U of keys, $|U| = N$, and a *hash table* T of size m, $m \in \mathbb{N}$, keys are mapped to table *positions* in a *range* M with $|M| = m$ via a *hash function* $h : U \to M$, and one would like to store a key x in cell $T[h(x)]$, i. e. in cell $h(x)$ of T. Of course there may be *collisions*, i. e. distinct keys $x, y \in U$ with $h(x) = h(y)$. Different ways to handle collisions have led to many different hashing schemes over the past 50 years. (For a basic introduction to hashing and its history see [28].) Arguably the most natural scheme is *chained hashing*, where collisions are resolved by using a linear list for each hash value such that keys with the same value are stored in the same list. Almost as old as chained hashing and of great popularity is the *linear probing* scheme, where keys are stored in the table itself, and not in linear lists. Whenever a key x is inserted and another key y already resides in cell $h(x)$, a linear probe of table cells starting at $h(x)$ is performed, i. e. sequentially the cells $h(x), h(x)+1, h(x)+2, \ldots$ (modulo the table size) are inspected, and x is inserted in the first cell that is not occupied by another key.

1.1.3 Hash Functions

The importance of choosing a suitable hash function h for a given hashing scheme is obvious. It is, for example, always a bad idea to choose a constant function—mapping all keys to the same table position. For chained hashing, where the time for an unsuccessful search of a key x is merely bounded by the length of the linear list belonging to $h(x)$, one clearly sees the bad effect of a constant function.

For a given set $S \subseteq U$ of keys to be inserted in the hash table T it would of course be perfect if h was 1-1 on S, i. e. for each set of two distinct keys $x, y \in S$ we have $h(x) \neq h(y)$. This leads us to the realm of *perfect hashing*. We won't go deep into it here. Nevertheless we mention the following: the goal of a perfect hash function can only be reached (in the most natural case $|U| = N \gg m = |M|$) if S is known in advance, and it is the more difficult to reach the more we are interested in a solution without much space overhead. To our knowledge, the best known practical way to construct a perfect hash function (PHF) for a set S of size n is due to Chazelle et al. [7], who implicitly explain it as a modification of the "Bloomier Filter" data structure. An improved analysis of this construction as well as a description of how to use such PHFs to construct minimal PHFs ($n = m$) is due to Bothelo et al. [3]. The evaluation time for these PHFs is constant and the amount of space needed to represent them is around a factor 2 from the information theoretical minimum[1]. A scheme for the

[1] In the case $n = m$ the information theoretical minimum is known to be approximately $1.44n$

construction of a minimal PHF proposed by Hagerup and Tholey [25] has even better properties from a theoretical point of view: constant evaluation time and asymptotic optimal space usage. However, their scheme is not well-defined for $n < 2^{150}$ as in this case it yields buckets (table cells) of size less than 1. This has the effect that their approach is not space efficient for reasonable values of n. A good survey of this whole topic can be found in [10] and [12].

Many theoretical analyses of hashing schemes assume that the hash function h be *fully random*, which means that the set $\{h(x) \mid x \in U\}$ is a set of independent and uniformly distributed random variables. (Basic terms of probability theory and how they can be applied in design and analysis of algorithms can be found in, e. g., [31].) Here too, we do not go deep into it, as this is not the approach followed in the thesis. We just mention that in practice full randomness in a mathematical sense can only be simulated. We get back to this in the context of cuckoo hashing in Section 1.1.5.3.

1.1.4 Universal Hashing

In this thesis we are concerned with what seems to be a practical way out of the dilemma of constructing a perfect hash function or simulating full randomness: *universal hashing*. In this scheme the hash function h is sampled from a given "universal" hash function class $\mathcal{H}$, where sampling and evaluating a function from $\mathcal{H}$ should be possible in an efficient way.

Universal hashing was first proposed by Carter and Wegman in the late 1970s. In their seminal paper [6] they, among other things, analyze chained hashing in combination with classes $\mathcal{H}$ that satisfy the following universality property (later on referred to as 1-universality, see Definition 1): For arbitrary distinct keys $x, y \in U$ and h chosen uniformly at random from $\mathcal{H}$ we have

$$\Pr(h(x) = h(y)) \leq \frac{1}{m}\,. \tag{1.1.1}$$

They show that if (1.1.1) holds then the expected length of a list is the same as for a fully random function, namely $\alpha = n/m$ (known as the *load factor*, where n refers to the number of keys currently stored in all lists). The proof of this result is fairly easy. But very often analyzing a hashing scheme with universal classes turns out to be difficult. The following may be evidence of this assertion: Carter and Wegman in [6] already raised the question of extending their analysis to linear probing. However, the behavior of linear probing with simple universal classes eluded researchers until Anna Pagh, Rasmus Pagh and Milan Ružić [38] recently presented an analysis that proves 5-wise independence[2] of the hash functions to be sufficient for this scheme to work well in the sense that the expected cost of an operation is constant, whereas pairwise independence does not suffice in that it may lead to expected logarithmic cost per operation.

In this thesis we analyze cuckoo hashing in combination with hash classes that we refer to as the *multiplicative class* and the *linear class*: First, let $1 \leq l \leq k$. The multiplicative class consists of functions h_a that map keys $x \in U = [2^k] = \{0, 1, \ldots, 2^k - 1\}$ via

$$h_a(x) := (a \cdot x \bmod 2^k) \operatorname{div} 2^{k-l}$$

to table positions in the range $[2^l]$, where the parameter a is an odd number modulo 2^k. Second, let p be a prime number and $m < p$. Then the linear class can be described as consisting of functions $h_{a,b}$ that map keys $x \in [p]$ via

$$h_{a,b}(x) := ((ax + b) \bmod p) \bmod m$$

to the range $[m]$, using parameters $a \in [p] - \{0\}$ and $b \in [p]$. Note how simple the functions of these classes are, and that their evaluation can be done very efficiently (compare [19]). Moreover, they have powerful universality properties (see Section 2.3).

1.1.5 Cuckoo Hashing and Its Relatives

Cuckoo hashing was invented by Pagh and Rodler in 2001 [40]. It is a *multiple choice* hashing scheme, where in a multiple choice hashing scheme each key can reside in one out of d possible cells of the hash table(s), for a fixed number $d \geq 2$. Furthermore it belongs to the special class of hashing schemes where collisions between keys are resolved by moving keys in the table as needed (under the constraints defined by the choices for each key). Precisely: The cuckoo hashing

[2]By k-wise independence, $k \geq 2$, of a hash class $\mathcal{H}$ we mean that for $h \in \mathcal{H}$ chosen uniformly at random each set of k random variables from $\{h(x) \mid x \in U\}$ is independent in the probability theoretic sense. Moreover, we implicitly mean that the hash value distribution is uniform, or at least "close" to uniform. See the brief discussion in Section **??**.

data structure consists of two tables T_1 and T_2, each of size m, and two hash functions h_1 and h_2 associated with T_1 and T_2, respectively. Each key $x \in U$ has two choices. It can either be stored in cell $T_1[h_1(x)]$ or in cell $T_2[h_2(x)]$. This obviously allows for lookups and deletions that run in constant time. As for insertions, a key x is made behave like a cuckoo: It is stored in cell $h_1(x)$ of T_1. If this cell was empty before, the insertion of x is complete. Otherwise if the cell was occupied by some key y then y is kicked out of the cell by x and becomes "nestless". Then y is stored in cell $h_2(y)$ of T_2. If this cell was empty before, we are done. Otherwise a key z previously stored in cell $h_2(y)$ of T_2 becomes nestless, and so forth. The procedure stops successfully, if for the first time a nestless key is inserted into an empty cell. In order to avoid infinite loops, Pagh and Rodler suggest to define a limit L for the maximum tolerated number of nestless keys during an insertion, and to perform a *rehash* once this limit is reached, i. e. the set S of all keys currently residing in the table is inserted from scratch with new hash functions h_1 and h_2. They set $L := \alpha \log m$ for a large enough positive constant α, and use this setting for their performance analysis of the insertion procedure.

If we step back for a moment and view the insertion procedure from a distance, overlooking its standard analysis in [41], then we see that the probability of the following event is an important parameter of its performance:

"For a given set $S \subseteq U$ the hash functions h_1 and h_2 are *not suitable*,"

i. e. the probability that it is not possible to place all keys of S according to the choices defined by h_1 and h_2. We call the probability for this bad event which depends on possible randomness of S, h_1, and h_2 the *failure probability* and abbreviate it with $p_F = p_F(S, h_1, h_2)$ in the following. We focus on p_F as a measure of performance throughout the thesis and prove lower bounds for it.

1.1.5.1 Cuckoo Hashing Under High Degree of Independence

In the following we let $m = (1+\varepsilon)n$ for an arbitrary constant $\varepsilon \in (0,1)$, or equivalently $n = (1-\delta)m$ for $\delta := \varepsilon/(1+\varepsilon)$. (In the literature both versions for bounding the number of keys stored in the tables occur.) If the hash functions are chosen uniformly at random from an $\Omega(\log n)$-wise independent class $\mathcal{H}$

then we know that for an arbitrary set $S \subseteq U$ of size n the failure probability p_F is $O(1/m)$. This is shown by Pagh in [39]. Based on this result Pagh and Rodler prove in [40] $O(n)$ expected time for the insertion of n keys under $\Omega(\log n)$-wise independence, and hence amortized expected constant time for a single insertion.

Another line of related work focuses on the case of *full randomness*, i. e. all hash values are independent and uniformly distributed. Kutzelnigg and Drmota show in [20, 29] that in this case we not only have $p_F = O(1/m)$ but $p_F = \Theta(1/m)$. (There is a quite simple and elegant proof of this fact, see Section 2.2.2.) In particular, they compute the exact constant of the Θ-notation as a function of δ for the above constant $\delta \in (0,1)$ with $n = (1-\delta)m$. They also consider the case $n = m$ and show $p_F = 1 - \sqrt{2/3} - o(1)$.

So, given a high degree of independence and randomness for all hash values, the expected behavior of cuckoo hashing is extremely good. However, the load factor $n/(2m)$ of no more than $1/2$ that is necessary to guarantee this good performance is a bit unsatisfactory.

1.1.5.2 How to Get a Better Space Usage?

There are generalizations of cuckoo hashing that afford a better space usage than the standard variant: *d-ary* and *blocked* cuckoo hashing. The former (which is characterized by using one table, $d > 2$ hash functions and buckets of size 1) was invented by Fotakis, Pagh, Sanders and Spirakis [23]. The latter (also being characterized by one table, but using only two hash functions and buckets of constant size ≥ 2) was independently proposed by Panigrahy [42], and by Dietzfelbinger and Weidling [17].

In the offline case, where a set S of keys is given and we just ask whether these keys can be placed (regardless of a specific insertion procedure), both variants achieve a space overhead of no more than an ε fraction together with an access time of $O(\log(1/\varepsilon))$ for an arbitrary constant $\varepsilon \in (0,1)$ under the assumption of complete independence and uniform distribution of all hash values. In other words, the threshold load factor β_d such that, with high probability, a set S of size n can be placed if $n/m \leq \beta_d$ and cannot be placed if $n/m > \beta_d$ is $1 - 2^{-\Theta(d)}$.

For blocked cuckoo hashing the above result is independently achieved by

Cain, Sanders, and Wormald in [4] and by Fernholz and Ramachandran in [21]. They make use of a natural graph theoretic interpretation of blocked cuckoo hashing: Buckets correspond to vertices and the two hash values (vertices) of a key correspond to an edge. Then a placement of keys into buckets of size d translates into orienting the edges in such a way that the in-degree of each vertex is at most d.

As for d-ary cuckoo hashing, lower bounds on the threshold load factor for $d = 2, 3, 4$ are established by Batu, Berenbrink, and Cooper in [1]. They reduce the problem of finding a key placement (for n keys with d choices each and m buckets of size 1) to finding a bipartite matching in a graph with bipartition (X, Y), where $x \in X$ corresponds to a key and $y \in Y$ corresponds to a bucket, and each edge (x, y) corresponds to one of the d choices of a key (represented by the vertex x). Lower bounds on the threshold load factor for each $d > 2$ are found by Dietzfelbinger and Pagh in [13]. They use a remarkably different approach by Calkin [5] who computes the probability that a matrix with n rows (corresponding to n keys) and m columns (corresponding to m buckets of size 1) has full row-rank if each of its rows is a random bit vector with exactly d ones (corresponding to the d choices of a key).

1.1.5.3 Is a High Degree of Independence Feasible?

In [32], Mitzenmacher and Vadhan prove a result of the following kind: If the data $S \subseteq U$ exhibits a sufficient amount of entropy then even simple universal hash functions—under certain constraints—lead to "almost fully random" hash values. This is a good explanation for the fact that simple hash functions often tend to work well in practice. However, as we will explain in Section 2.4 one has to apply this result with care.

If no randomness can be assumed for S, $|S| = n$, then full randomness of the hash values can still be simulated as shown in [14, 18, 35, 37, 44, 45]. However, the price one has to pay for full randomness is quite high: For storing a fully random function we need space of at least $n \log m$, which is the information theoretic minimum. Moreover, there is a time-space tradeoff. The best known construction exhibits an asymptotically optimal space usage of $(1 + \varepsilon)n \log m$ and an evaluation time of $O(\log(1/\varepsilon))$ ([14]). So, at least if space efficiency is an issue then all these approaches to simulate full randomness may not be an option.

Fortunately, for cuckoo hashing to work well we do not necessarily need full randomness. It is sufficient that the hash functions be $\log n$-wise independent ([41]). This can, e. g., be achieved by Siegel's construction [45] at a price of space $O(n^{\zeta})$ for a not too small positive constant $\zeta < 1$ and a large constant evaluation time. Arguably the simplest and most efficient approach to achieve this goal is the one described in [14] (*split-and-share*, first mentioned in [23], sketched in [17], and more thoroughly explained in [12]). Split-and-share can be used for cuckoo hashing in a way that consumes only space $O(n^{\delta})$ for an arbitrary small positive constant $\delta < 1$ while having (relatively) small constant evaluation time. (For details see [14].)

So, whether we consider a high degree of independence to be feasible or not depends upon the price in terms of time and space that we are willing to pay for it. All told, the approach of using simple universal functions remains very attractive from a practical point of view as such functions tend to be highly efficient with respect to evaluation time and space consumption—even compared to the above-mentioned split-and-share approach.

1.1.5.4 Behavior of Cuckoo Hashing with Weaker Hash Classes

In experiments cuckoo hashing reveals a good behavior with weaker universal classes like the class of square polynomials over a prime field (compare Section 3.5). However, in [41] Pagh and Rodler report on experiments where for the multiplicative class, that almost has the universality property (1.1.1) a bad behavior was observed. Independently from [41], Andrea Ott in [36] also reports on erratic behavior of this class. It is one of the main purposes of this thesis to mathematically explain this phenomenon.

In Section 2.3.2 we will discuss a work of Cohen and Kane[3] that has not yet been published. There it is claimed that 2-, 3-, and 5-wise independence (and even so-called $\Theta(\log n)$-wise-joint-independence) of the hash functions may not be sufficient to guarantee a good behavior of cuckoo hashing. Furthermore, a proof for good behavior of cuckoo hashing when h_1 is chosen from a pairwise

[3] `http://web.mit.edu/dankane/www/Independence%20Bounds.pdf`

independent class and h_2 is chosen from a $O(\log n)$-wise independent class is suggested. However, for the time being just note that all the hash classes under consideration are nonstandard and quite unattractive for practical use.

All told, the best proven upper bound on the degree of independence sufficient for cuckoo hashing is that of [40]: $O(\log n)$. In this thesis we give a further example for the fact that pairwise independence is not sufficient to guarantee a good behavior of cuckoo hashing. The class we consider is a standard hash class with a space efficient representation.

1.2 Results of this Thesis

If we agree upon "expected amortized constant time for an insertion" being "good behavior" then (with view to the results described in Section 1.1.5.1) it is justified to say that cuckoo hashing behaves well if the failure probability p_F is $O(m^{-c})$ for a constant $c \geq 1$. However, according to Kirsch, Mitzenmacher, and Wieder [27] already a failure probability of $O(1/m^3)$ leads to a failure frequency that is arguably too high for production systems. If that is true then all the more it holds for a *constant* failure probability, or even a failure probability of $1 - o(1)$.

The following chapters are devoted to proofs of lower bounds on p_F when cuckoo hashing is combined with the *multiplicative class* and the *linear class* (see Section 1.1.4). We distinguish between $m/N \geq N^{\gamma-1}$ and $m/N < N^{\gamma'-1}$ for suitable constants $\gamma, \gamma' \in (0,1)$, corresponding to *dense key sets* (m/N "large") and *sparse key sets* (m/N "small"), respectively.

1.2.1 Dense Key Sets

(a) In Section 3.2 we show that if cuckoo hashing with $2^l = m = 2n$ is employed (compare Section 1.1.5.1: this table size is almost twice as large as the threshold sufficient for the standard analysis), then all pairs of multiplicative functions will work badly with high probability for fully random key sets of size n (if $m/N \geq N^{\gamma-1}$ for a suitable constant $\gamma \in (0,1)$). In words according to [32], although the entropy of the input data is as large as possible, for every pair of multiplicative hash functions the failure probability for a key set S of relatively small size in comparison to m is extremely high. This explains in part the experimental results obtained in both [41] and [36], and justifies a warning against using this simple class of hash functions in combination with cuckoo hashing. Moreover, our result shows that the condition in [32] which translates into the requirement that S must not be too dense in the universe (see Section 2.4) is necessary, and that it is relevant even in very natural circumstances (standard hash classes, fully random key sets).

(b) In Section 3.3 we show that cuckoo hashing with the standard pairwise independent linear class exhibits a similar behavior as the multiplicative class in (a). This is true even when the two hash functions use different prime moduli, as proven in Section 3.4.

The proofs of all results mentioned so far follow the same scheme: We study the "complete cuckoo graph" created by all keys combined with arbitrary fixed hash functions from the considered classes, where an edge represents the hash values of a key, and identify certain "bad edge sets" with the property that the insertion of the corresponding sets of keys must fail. We show that random edge sets of relatively small size contain a bad edge set with high probability.

1.2.2 Sparse Key Sets

We show in Section 4.1 that the above result for the multiplicative class can be "lifted" to larger universes U, i. e. sparse key sets, but then the key set S must be chosen randomly from a special subset $U' \subseteq U$, where again $m/|U'| \geq |U'|^{\gamma-1}$.

The rest of Chapter 4 is dedicated to a similar result. However, construction and proof are totally different. We show that there are structured key sets S, constructed as a mixture of regular patterns and randomly chosen keys, that will make cuckoo hashing fail with probability $\Omega(1)$ (in place of the $O(1/m)$, resulting from [39]). Here again, both of our results for the case of sparse key sets are no contradiction to [32], as we allow only very restricted randomness in the data.

Finally, in the respective situations of the above-mentioned results for both dense and for sparse key sets the existence of key sets without any random elements for which cuckoo hashing behaves badly is a natural consequence.

CHAPTER 2

Preliminaries

This chapter summarizes notions and concepts necessary for understanding our work. For the sake of completeness and the reader's convenience, we start in Section 2.1 with an additional description of cuckoo hashing, reduced to the main facts.

In Section 2.2 we introduce the important concept of the cuckoo graph. It serves as a mathematical abstraction of the cuckoo hashing scenario. As important as the concept of the cuckoo graph is the notion of a bad edge set that we define and explain in Section 2.2.1. A bad edge set corresponds to a set of keys that cannot be inserted according to the cuckoo scheme. A first application of this knowledge can be found in Section 2.2.2 where we present a simple and elegant proof of the failure probability (compare Sections 1.1.5 and 1.1.5.1) in the case of full randomness.

In Section 2.3 we define the hash function families that are in the focus of this thesis. In order to describe the randomness properties of these families when used in the universal hashing scenario we introduce the notion of universality. A very common alternative way to describe such randomness properties is to apply the term "independence". The relation between the notions universality and independence is discussed in Section 2.3.1. In the following Section 2.3.2 we briefly comment on a related work of Cohen and Kane with results quite near to ours but focussing on a special kind of hash families of which the members cannot be represented efficiently.

The rest of Chapter 2, Section 2.4, is dedicated to a detailed description of an important result of Mitzenmacher and Vadhan [32]: Under certain constraints, a sufficient amount of randomness in the keys combined with a weak universal function can lead to values with randomness properties that are close to full randomness. We carefully discuss the relation to our results in Sections 2.4.5 and 2.4.6.

2.1 Cuckoo Hashing

Given two hash tables T_1, T_2, each of size m, and hash functions h_1, h_2 mapping a *universe* U of N keys to $[m] = \{0, 1, \ldots, m-1\}$, the two possible positions for a key x are cell $h_1(x)$ of T_1 and cell $h_2(x)$ of T_2 (Fig. 2.1).

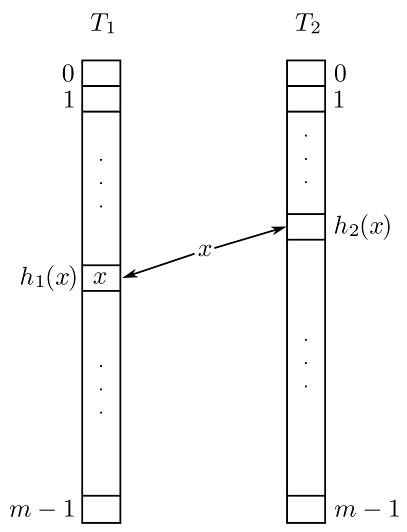


Figure 2.1: A key x is either stored in cell $h_1(x)$ of T_1 or in cell $h_2(x)$ of T_2.

For a given set $S \subseteq U$ and hash functions h_1, h_2 we say that h_1 *and* h_2 *are suitable for* S if it is possible to store the keys from S in T_1, T_2 according to h_1, h_2 in such a way that distinct keys are stored in distinct table cells. For a detailed description and the basic analysis of cuckoo hashing, see [41].

2.2 The Cuckoo Graph

The *cuckoo graph* (see [11, 18, 35]) represents the hash values on a set S of keys in U for hash functions $h_1, h_2 \colon U \to [m]$. Its vertices correspond to the cells in tables T_1 and T_2, and an edge connects the two possible locations $T_1[h_1(x)]$ and $T_2[h_2(x)]$ for a key $x \in S$. Formally, the cuckoo graph $G(S, h_1, h_2)$ is defined as an undirected bipartite multigraph (V_1, V_2, E) with vertex sets $V_1 = [m]$ and $V_2 = [m]$, and edge (multi)set $E = \{(h_1(x), h_2(x)) \mid x \in S\}$ (Fig. 2.2). We refer to $G(U, h_1, h_2)$ as the *complete cuckoo graph*.

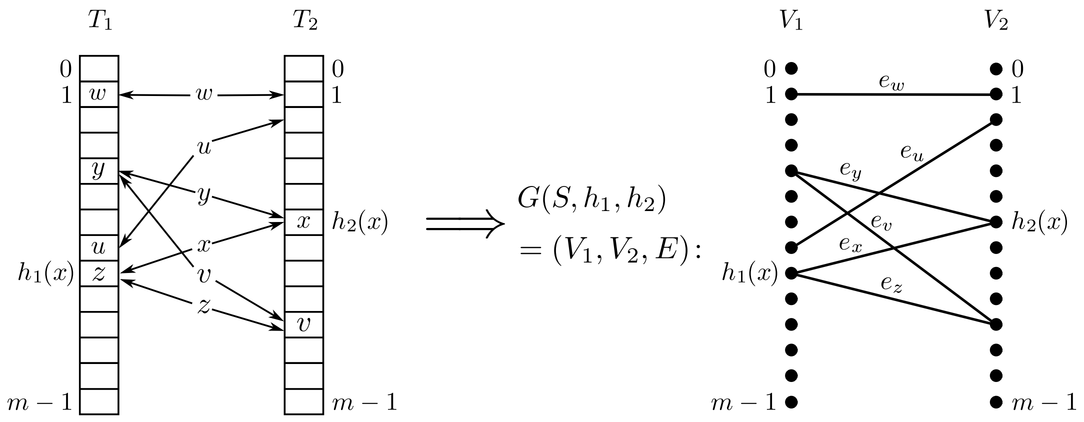


Figure 2.2: The cuckoo graph represents the hash values on a set $S \subseteq U$ for functions h_1, h_2. Here we have $S = \{u, v, w, x, y, z\}$.

2.2.1 Bad Edge Sets

If $G(S, h_1, h_2) = (V_1, V_2, E)$, we call $E' \subseteq E$ a *bad edge set* if $|E'|$ is larger than the number of distinct vertices that are incident with edges in E'. The following two lemmata characterize suitability in different ways. Both characterizations will turn out to be useful for the proofs of our main theorems.

Lemma 1. *The hash functions h_1 and h_2 are suitable for S if and only if $G(S, h_1, h_2)$ does not contain a bad edge set.*

Proof. If $G = G(S, h_1, h_2) = (V_1, V_2, E)$ contains a bad edge set $E' \subseteq E$ then the corresponding set $S' \subseteq S$ of keys cannot be stored in T_1, T_2 according to h_1, h_2 in such a way that distinct keys are stored in distinct table cells because the number of possible cells for the keys in S' is at least by one smaller than $|S'|$. So, h_1 and h_2 are not suitable for S', and thus cannot be suitable for the whole set S either.

Vice versa, let there be no bad edge set in the cuckoo graph G. Assume that, during the sequential insertion of the keys in S according to the cuckoo scheme without bound for the maximum tolerated number of nestless keys during an insertion (or—compare Section 1.1.5—equivalently with $L = \infty$), the insertion procedure for a first key $x \in S$ nevertheless gets into an infinite loop. Then consider the corresponding walk W in G that starts in $h_1(x) \in V_1$ and traverses the edges that belong to the keys which become nestless during the insertion. As G is finite W visits some of the vertices in G infinitly often. Let u be the first vertex that W visits for the second time. Furthermore, be aware of the following crucial fact: According to the cuckoo scheme the direction of an edge's traversal by W alternates, and between two subsequent traversals of the same edge lies at least one traversal of another edge. This, in combination with u being the first vertex to be visited repeatedly, has two implications that turn out to be useful for our purposes:

(1) The edges traversed by W from u to u form a simple cycle C.

(2) The first vertex v traversed by W after the traversal of C is not a vertex on C.

Assume w. l. o. g. that $u = h_1(x)$ holds. (Otherwise we consider the walk W' that starts when W visits u for the first time.) The next vertex w that W visits repeatedly after finding v is either on C or not. However, as v is not on C, the second visit of w closes a cycle that is distinct from C. The edges traversed by W so far form a subgraph with two distinct cycles. A subgraph with two distinct cycles forms a bad edge set. Contradiction! Consequently, S can be inserted successfully by means of the cuckoo scheme, which implies that h_1 and h_2 are suitable for S. $\square$

The next lemma was proven by Devroye and Morin in [11] (implicit also in [39] and [41]).

Lemma 2. *The hash functions h_1 and h_2 are suitable for S if and only if each connected component of $G(S, h_1, h_2)$ is either a tree or unicyclic.*

Proof. It suffices to show that each connected component of $G(S, h_1, h_2)$ is either a tree or unicyclic if and only if there exists no bad edge set in $G(S, h_1, h_2)$. Then we apply Lemma 1.

First assume that $G(S, h_1, h_2)$ contains a connected component C with more than one cycle. Obviously C contains more edges than vertices, and hence its edges form a bad edge set. Second assume that E' is a bad edge set in

$G(S, h_1, h_2)$. We can w.l.o.g. assume that the subgraph G' corresponding to E' is connected. As E' contains more edges than vertices, G' contains more than one cycle. □

Observe that Lemmata 1 and 2 imply the following: $G(S, h_1, h_2)$ contains a bad edge set if and only if one of its connected components is neither a tree nor unicyclic, and hence contains at least two cycles.

In all proofs of the theorems in Chapter 3, we will focus on *minimal* bad edge sets (of constant size), where we call a bad edge set $E' \subseteq E$ minimal if each true subset of E' is no longer a bad edge set. If E' is a minimal bad edge set then the corresponding subgraph G' of $G(S, h_1, h_2)$ is connected, contains one edge more than vertices, and has no leaves, i.e. vertices with degree 1. The abstract shape of such a subgraph is depicted below (Fig. 2.3; lines represent simple paths of length at least 1, and dots are selected vertices): Either one simple cycle together with a simple path that connects exactly two distinct vertices of this cycle, or two simple cycles with pairwise distinct edges and a simple path of length at least 0 that connects these cycles.

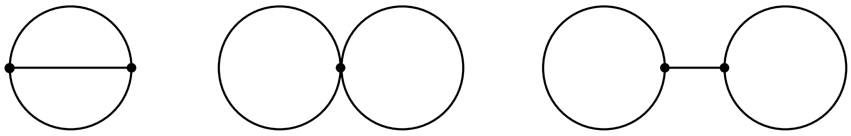


Figure 2.3: abstract shape of minimal bad edge sets

2.2.2 Failure probability p_F under Full Randomness

Let us assume full randomness, i.e. all hash values are independent and their distribution is uniform. Lemmata 1 and 2 are a key that opens the door to a simple and elegant computation of an asymptotic bound of p_F for sets S of size $(1-\delta)m$.

Theorem 1 ([20, 29, 40]). *Let $S \subseteq U$ be of size $n = (1-\delta)m$ for an arbitrary constant $\delta \in (0,1)$. Then we have $p_F \in \Theta(1/m)$ under the assumption of full randomness.*

Proof. As for the lower bound, observe that a smallest possible bad edge set, or equivalently a smallest possible bipartite multigraph (V_1, V_2, E') with $E' \neq \emptyset$ of which the induced subgraphs of the connected components of size > 1 are neither trees nor unicyclic, consists of no more than three edges with the same endpoints $v \in V_1$ and $w \in V_2$ (Fig. 2.4).

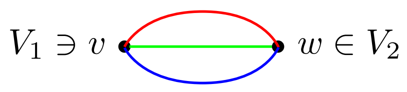


Figure 2.4: a smallest possible bad edge set

Under the full randomness assumption we can view $G(S, h_1, h_2)$ as the result of a random experiment where n edges (in the following represented by $1, \ldots, n$) are "thrown" independently and uniformly at random into m^2 initially empty bins $(i, j) \in [m] \times [m] = V_1 \times V_2$. All told, the probability that there exist three of the n "edge balls" which fall into the same bin is a lower bound for p_F, because three edge balls in the same bin represent a smallest possible bad edge set. If we proceed and try to bound the probability of the named event from below then the application of a union bound does not help, of course. We have to make use of a tool like the following *conditional expectation inequality* instead (for a proof see [31]).

Lemma 3. *Let Y be the sum of arbitrary Bernoulli random variables $Y_1, \ldots, Y_t$, $t \in \mathbb{N}$. Then we have*

$$\Pr(Y > 0) \geq \sum_{1 \leq i \leq t} \frac{\Pr(Y_i = 1)}{\mathrm{E}(Y \mid Y_i = 1)} .$$

This lemma also plays a vital role in our proofs of the theorems in Chapter 3. However, let X_T, $T \subseteq \{1, \ldots, n\}$, be a Bernoulli random variable that takes the value 1 if and only if all balls in T fall into the same bin. Define

$$X := \sum_{T \subseteq \{1,\ldots,n\}, |T|=3} X_T$$

such that X measures the number of distinct subsets of three edge balls of which the respective elements fall into the same bin. Then we have

$$\begin{aligned} p_F &\geq \Pr(\exists \text{ three balls that fall into the same bin}) \\ &= \Pr(X > 0) . \end{aligned}$$

We apply Lemma 3 for $(Y_1, \ldots, Y_t) = (X_{T_1}, \ldots, X_{T_{\binom{n}{3}}})$ and $Y = X$, where an arbitrary enumeration $T_1, \ldots, T_{\binom{n}{3}}$ of the sets of three balls from $\{1, \ldots, n\}$ is assumed to be given. Fix $T_i = \{i_1, i_2, i_3\}$ arbitrarily, $1 \leq i \leq \binom{n}{3}$. The probability that balls i_1, i_2, and i_3 fall into the same bin is equal to the probability that i_2 and i_3 fall into an arbitrarily fixed bin, i. e.

$$\Pr(X_{T_i} = 1) = \frac{1}{m^4}\,. \tag{2.2.1}$$

Furthermore, we have

$$\mathrm{E}(X \mid X_{T_i} = 1) = \sum_{1 \le j \le \binom{n}{3}} \Pr(X_{T_j} = 1 \mid X_{T_i} = 1)\,. \tag{2.2.2}$$

Obviously, $\Pr(X_{T_j} = 1 \mid X_{T_i} = 1)$ depends on $|T_j \cap T_i|$. It is not hard to see that for arbitrarily fixed T_j we have:

$$\Pr(X_{T_j} = 1 \mid X_{T_i} = 1) = \begin{cases} 1 & \text{if } j = i\,, \\ \frac{1}{m^2} & \text{if } |T_j \cap T_i| = 2\,, \text{and} \\ \frac{1}{m^4} & \text{if } |T_j \cap T_i| \le 1\,. \end{cases} \tag{2.2.3}$$

By counting the number of different sets T_j, $1 \le j \le \binom{n}{3}$, which have d elements in common with T_i, $d \in \{0,1,2,3\}$, respectively, we derive from (2.2.2) and (2.2.3) that $\mathrm{E}(X \mid X_{T_i} = 1)$ is equal to

$$1 + \binom{3}{2} \cdot \binom{n-3}{1} \cdot \frac{1}{m^2} + \binom{3}{1} \cdot \binom{n-3}{2} \cdot \frac{1}{m^4} + \binom{n-3}{3} \cdot \frac{1}{m^4}\,.$$

So, taking into account that $n = (1-\delta)m$, we have

$$\mathrm{E}(X \mid X_{T_i} = 1) \in O(1)\,. \tag{2.2.4}$$

This finally leads to the desired result:

$$\begin{aligned} p_{\mathrm{F}} &\ge \Pr(X > 0) && (\text{def. } X) \\ &\ge \sum_{1 \le i \le \binom{n}{3}} \frac{\Pr(X_{T_i} = 1)}{\mathrm{E}(X \mid X_{T_i} = 1)} && (\text{Lemma } 3) \\ &\in \Omega\left(\frac{1}{m}\right) && ((2.2.1), (2.2.4), \text{ and } n = (1-\delta)m) \end{aligned}$$

As for the upper bound, observe that a bad edge set E' contains a minimal bad edge set $E'' \subseteq E'$. This can be proven by induction (over the structure of E'): If E' is minimal then we are done. Otherwise there exists a true subset $B' \subsetneq E'$ which is bad. By induction hypothesis $E' := B'$ contains a minimal bad edge set E''—as claimed.

Recall the characteristic shape (Fig. 2.3) of the subgraph corresponding to a minimal bad edge set. It is not hard to see that every such subgraph consists of one simple path along vertices $v_1, \ldots, v_k$, with edges e_t connecting subsequent vertices v_t and v_{t+1}, $1 \leq t \leq k-1$, plus two further edges e_0, connecting v_1 to some vertex v_i in $\{v_2, \ldots, v_k\}$, and e_k, connecting v_k to some vertex v_j in $\{v_1, \ldots, v_{k-1}\}$ (Fig. 2.5).

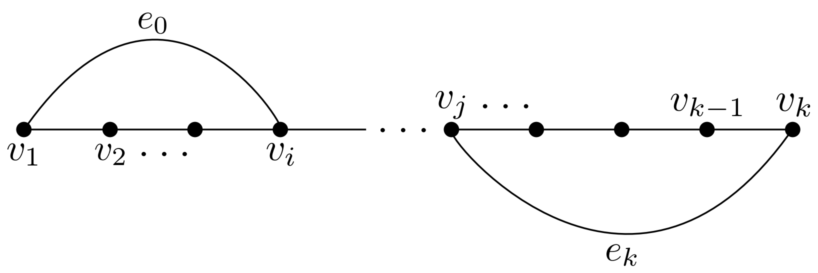


Figure 2.5: the subgraph corresponding to a minimal bad edge set; the cases $i > j$, $i = j$, and $i < j$ represent the three cases depicted in Fig. 2.3

The probability that such a "bad" subgraph occurs can be bounded as follows. We sum over $2 \leq k \leq n-1$, because the length of the simple path is ≥ 1 and $\leq n-2$. Now consider a fixed k. Then there exist no more than $2 \cdot m^k \cdot k^2$ bad subgraphs of the above shape: They consist of a simple path $v_1, \ldots, v_k$ starting either in V_1 or V_2 (factor 2) with its k vertices alternatingly from V_1 and V_2 (factor m^k, as $|V_1| = |V_2| = m$) plus two further edges v_i and v_j (factor $(k-1)^2 < k^2$). Now fix a bad subgraph consisting of a path $v'_1, \ldots, v'_k$ and extra vertices v'_i and v'_j. For bounding the probability that $G(S, h_1, h_2)$ contains this bad subgraph we again consider the random experiment of throwing n edge balls into m^2 bins corresponding to $V_1 \times V_2$. There are fewer than n^{k+1} ways to fix the edges (among the n edge balls) that are supposed to be part of the bad subgraph (factor n^{k+1}). Now fix these edge balls. The probability that each of these $k+1$ balls falls into the correct bin is $m^{-2(k+1)}$. This finally yields

$$\begin{aligned}
p_{\mathrm{F}} &\leq \Pr(\exists \text{ (minimal) bad edge set}) \\
&\leq \sum_{2 \leq k \leq n-1} (2 \cdot m^k \cdot k^2) \cdot (n^{k+1} \cdot m^{-2(k+1)}) \\
&= \frac{2}{m} \cdot \sum_{2 \leq k \leq n-1} \left(\frac{n}{m}\right)^{k+1} \cdot k^2 \\
&< \frac{2}{m} \cdot \sum_{2 \leq k \leq \infty} k^2 (1-\delta)^{k+1} \in O\left(\frac{1}{m}\right) \qquad (\text{for } n = (1-\delta)m),
\end{aligned}$$

since the k-series is bounded by a constant. □

2.3 Hash Function Families

We consider two different kinds of hash function families. First, let $1 \leq l \leq k$. Then let

$$\mathcal{H}_{k,l}^{\text{mult}} := \{h_a \colon [2^k] \to [2^l] \mid a \in O_k\}\ ,$$

where $O_k := \{1, 3, \ldots, 2^k - 1\}$, and for $x \in [2^k]$ we let

$$h_a(x) := (a \cdot x \bmod 2^k) \operatorname{div} 2^{k-l}\ .$$

We refer to this family as the *multiplicative class* [19]. Second, let p be a prime number, and $m < p$. Then let

$$\mathcal{H}_{p,m}^{\text{lin}*} := \{h_{a,b} \colon [p] \to [m] \mid a \in [p] - \{0\}, b \in [p]\}\ ,$$

where, for all $x \in [p]$, we define

$$h_{a,b}(x) := ((ax + b) \bmod p) \bmod m\ .$$

We refer to this family as the *linear class* [6]. More generally, we consider the hash family of polynomials of degree up to $d - 1$, i. e.,

$$\mathcal{H}_{p,m}^{d} := \{h_{a_0,\ldots,a_{d-1}} \colon [p] \to [m] \mid a_0, \ldots, a_{d-1} \in [p]\}\ ,$$

where

$$h_{a_0,\ldots,a_{d-1}}(x) = ((a_0 + a_1 x + \cdots + a_{d-1} x^{d-1}) \bmod p) \bmod m$$

for $x \in [p]$, see [6]. Obviously, $\mathcal{H}_{p,m}^{\text{lin}*}$ is practically the same as $\mathcal{H}_{p,m}^{2}$. We used $\mathcal{H}_{p,m}^{3}$ as a reference class in our experiments.

In order to classify families of hash functions, we use the following well known generalizations of the original notion of universality, which is due to Carter and Wegman [6].

Definition 1. *A family $\mathcal{H}$ of hash functions $h \colon U \to [m]$ is called (c,k)-*universal *if for arbitrary distinct keys $x_1, \ldots, x_k \in U$, arbitrary values $y_1, \ldots, y_k \in [m]$, and a function $h \in \mathcal{H}$ chosen uniformly at random, we have*

$$\Pr(h(x_1) = y_1, \ldots, h(x_k) = y_k) \leq \frac{c}{m^k}\ .$$

*It is called c-*universal *if for arbitrary keys $x \neq y$ and h chosen uniformly at random, we have*

$$\Pr(h(x) = h(y)) \leq \frac{c}{m}\ .$$

So, if $\mathcal{H}$ is a (c, k)-universal class and h is chosen uniformly at random from $\mathcal{H}$ then k is a measure for the degree of independence of the hash values of h, whereas c bounds their deviation from the uniform distribution. Furthermore, our notion of c-universality is just an abbreviation of $(c, 1)$-universality.

In [19], the multiplicative class is proved to be 2-universal. For any fixed $d \geq 2$, the class of polynomials of degree up to $d - 1$ is known to be $(2, d)$-universal. The linear class is 1-universal and $(1, 2)$-universal.

2.3.1 Independence versus Universality

As already mentioned in Section 1.1.4, we refer to a class $\mathcal{H}$ of hash functions as being k-wise independent, $k \geq 2$, if for $h \in \mathcal{H}$ chosen uniformly at random each set of k random variables from $\{h(x) \mid x \in U\}$ is independent in the probability theoretic sense and the hash value distribution is uniform, or at least "close" to uniform. In order to describe the quality of the hash value distribution more precisely we prefer to make use of our above defined notion of universality (compare Definition 1). However, yet another way to classify the quality of the hash value distribution can, e. g., be found in [38]: There a class $\mathcal{H}$ is called ε-approximately uniform, $\varepsilon > 0$, if the hash values for a randomly chosen $h \in \mathcal{H}$ have L_∞ distance at most ε from the uniform distribution, i. e. for any $x \in U$ and any $y \in M$ it holds that $|\Pr(h(x) = y) - 1/m| \leq \varepsilon$. Direct comparison of the different notions yields the following. If h is chosen uniformly at random from a (c, k)-universal class $\mathcal{H}$, $c \geq 1$, then it is k-wise independent and $((c^{1/k} - 1)/m)$-approximately uniform. Accordingly, if h is chosen from a c-universal class then it is $((c - 1)/m)$-approximately uniform and we know nothing about the degree of independence of its hash values.

2.3.2 Brief Discussion of a Recent Work

In Section 1.1.5.4 we mentioned a work of Cohen and Kane that has not yet been published, but is closely related to the results of this thesis. There it is claimed, e. g., that 2-, 3-, and even 5-wise independence of the hash functions may not suffice to guarantee a good behavior of cuckoo hashing, where they define k-wise independence in a way that directly corresponds to our notion of

$(O(1), k)$-universality (Definition 1). So, we are not the first to suggest a proof for the fact that cuckoo hashing works badly with certain weaker universal hash classes. However, the main difference to the results of this thesis seems to be the following. In contrast to the linear and the multiplicative class that we consider, all the hash classes considered by Cohen and Kane are nonstandard and quite unattractive for practical use because of their large space consumption: These hash classes $\mathcal{H}$ without exception have the property to be *value oblivious*, i. e., if $h : U \to [m]$ belongs to $\mathcal{H}$ then for each $\pi \in S_m$, i. e. each permutation π of the range $[m]$, $\pi(h)$ (defined by $\pi(h)(i) := \pi(h(i))$) also belongs to $\mathcal{H}$. Now, observe that a nonempty value oblivious class $\mathcal{H}$ has a cardinality of at least $m!$, and hence we need $\Omega(m \log m)$ bits to represent a single function from $\mathcal{H}$. This is a relatively large amount of space.

Two further things we feel free to criticise in a (hopefully) constructive way: First of all, Cohen and Kane prove a little less than what they claim[1]. They claim that something holds for a, say, k-wise independent class. But then they prove it for a class that (for a randomly chosen h) does not have the necessary property

$$\Pr(h(x_1) = a_1, \ldots, h(x_k) = a_k) = \frac{O(1)}{m^k}$$

but only the far weaker property

$$\Pr(h(x_1) = \cdots = h(x_k)) = \frac{O(1)}{m^{k-1}}$$

for arbitrary distinct keys $x_1, \ldots, x_k$ and arbitrary values $a_1, \ldots, a_k$. Second, in the proof (suggested by Cohen and Kane) of a good behavior of cuckoo hashing when h_1 is chosen from a pairwise independent class and h_2 is chosen from a $O(\log n)$-wise independent class the named authors make the assumption that m is a sufficiently large constant times n (where n denotes the size of the set S of keys in the tables). Following their argumentation we get $m \geq 8 \cdot n$. Obviously, this strongly relativizes the good behavior that they prove.

2.4 When Do Simple Hash Functions Work?

As we said, theoretical analyses of hashing schemes often make the assumption of full randomness. But from a practical point of view full randomness can

[1]Thanks to Peter Bro Miltersen for pointing this out

only be simulated, and the efficiency of the best known simulations is quite unsatisfactory with respect to their space consumption, as explained in Section 1.1.5.3. However, in experiments one can observe the phenomenon that hashing schemes combined with simple universal functions tend to work fairly well, even though the performance guarantees that we can prove for this setting may be noticeably weaker than those proven under the full randomness assumption.

Recently, Mitzenmacher and Vadhan [32] (an improved version can be found in [9]) tried to explain this phenomenon, saying that if the data (the set S of keys to be inserted) is "sufficiently" random then even a function that is randomly chosen from a simple universal class may yield a hash value distribution that is very "close" to full randomness, and thus will lead to essentially the same performance as in the case of full randomness. The idea behind this is not entirely new. A hash function class with the above-mentioned property is actually also known as a *randomness extractor* [34]. Randomness extractors have many applications in theoretical computer science. They play a vital role in the realm of pseudorandomness. (For surveys about the theory of constructing a randomness extractor see [33, 43].) However, Mitzenmacher and Vadhan seem to be the first who apply this theory to universal hashing.

In the following we will present one specific result of [32] which we consider to be the most important one among all results in [32] with respect to the results of this thesis. Only a very close look at this result enables to see that there is no contradiction to our results.

2.4.1 Some Notation

For a discrete random variable X, the *support* of X is

$$\mathrm{Supp}(X) = \{x \mid \Pr(X = x) > 0\} .$$

For a finite set F, U_F denotes a random variable that is uniformly distributed on F.

2.4.2 Some Definitions

Let $\mathcal{H}$ be a class of hash functions that map a finite set U of size N to a range M, $|M| = m$. For a random variable X taking values in U, the *max probability* is

$$\mathrm{mp}(X) = \max_{x \in U} \Pr(X = x)\,,$$

i. e. an upper bound for the probability that X takes an arbitrarily fixed value $x \in U$. The *collision probability* is

$$\mathrm{cp}(X) = \sum_{x \in U} \Pr(X = x)^2\,,$$

and thus can be viewed as the expected probability that equally distributed and independent random variables X and X' collide, i. e. they take the same value. If Y is another random variable with values in U, then the *statistical difference* of X and Y is

$$\Delta(X, Y) = \max_{S \subseteq U} |\Pr(X \in S) - \Pr(Y \in S)|\,,$$

which is an upper bound for the absolute value of the difference between the probability that an arbitrary event $S \subseteq U$ occurs for X and the probability that it occurs for Y. X and Y are called *ε-close* if $\Delta(X, Y) \leq \varepsilon$. Furthermore we say that a sequence of random variables $(X_1, \ldots, X_T)$ taking values in U^T is a *block source with collision probability p_b per block* (respectively, *max probability p_b per block*) if for every i, $1 \leq i \leq T$, and every $(x_1, \ldots, x_T) \in \mathrm{Supp}(X_1, \ldots, X_T)$, we have

$$\mathrm{cp}(X_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) \leq p_b$$

(respectively, $\mathrm{mp}(X_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) \leq p_b$). So, if we assume for a block source with collision probability p_b that its random variables take values sequentially (first X_1, then X_2, and so forth) then we know something about the effect of dependencies between the random variables on the probabilities of values in this sequence.

2.4.3 The Model

In [32] the keys to be inserted in a table are viewed as being random variables X distributed over the finite set U of size N. As a measure of the amount of randomness in a key they use the above maximum probability $\mathrm{mp}(X)$ as well as the collision probability $\mathrm{cp}(X)$. Measuring these quantities is equivalent to measuring the so-called *min-entropy*

$$\mathrm{H}_\infty = \min_{x\in U} \log\left(\frac{1}{\Pr(X=x)}\right) = \log\left(\frac{1}{\mathrm{mp}(X)}\right)$$

and the *Rényi entropy*

$$\mathrm{H}_2 = \log\left(\frac{1}{\sum_{x\in U}\Pr(X=x)^2}\right) = \log\left(\frac{1}{\mathrm{cp}(X)}\right) ,$$

respectively. If $|\mathrm{Supp}(X)| \leq K$, then

$$\mathrm{mp}(X) \geq \mathrm{cp}(X) \geq \frac{1}{K}$$

(i. e. $\mathrm{H}_\infty(X) \leq \mathrm{H}_2(X) \leq \log K$), with equality if and only if X is uniform on its support. Moreover, min-entropy and Rényi entropy can be used interchangeably because it also holds that

$$\mathrm{mp}(X) \leq \mathrm{cp}(X)^{\frac{1}{2}} ,$$

and hence these two measures are always within a factor two of each other. Now Mitzenmacher and Vadhan model a sequence of T keys as a block source, i. e. a sequence of arbitrarily correlated random variables, yet where each item is guaranteed to have some entropy conditioned on the previous items.

2.4.4 Randomness Extraction

Mitzenmacher and Vadhan conclude from the so-called *Leftover Hash Lemma* [26] (a similar lemma was also used in [2]) that each 1-universal hash class is a "strong" randomness extractor. Now the classic approach to extracting randomness from a block source is to apply a (strong) randomness extractor to each block of the source. An application of this approach leads to the following theorem.

Theorem 2 ([8, 46]). *Let $h\colon U \to M$ be chosen uniformly at random from a 1-universal class $\mathcal{H}$. For every block source $(X_1,\ldots,X_T)$ with collision probability $1/K$ per block, the random variable $(h,h(X_1),\ldots,h(X_T))$ is $(T/2)\cdot\sqrt{m/K}$-close to (h,U_M).*

So if we have T keys coming from a block source $(X_1, \dots, X_T)$, and we want the hash values $(h(X_1), \dots, h(X_T))$ resulting from uniformly random choosing h from a 1-universal class to be ε-close to the uniform distribution on M^T then each key is required to have (Rényi) entropy at least $\log K \geq \log(mT^2/4\varepsilon^2) = \log m + 2\log(T/2\varepsilon)$. (In [9] it is shown that the factor 2 can be omitted. The amount of entropy required is only $\approx \log m + \log T$. However, the following observation remains true.)

2.4.5 Crucial Observation

Obviously we have $K \leq N$. In this thesis we consider a situation where $T \geq m/2$. If in this setting we would like to derive ε-closeness of $(h(X_1), \dots, h(X_T))$ to U_{M^T} then it directly follows from the above considerations that N must be at least $m^3/(16\varepsilon^2)$ or, equivalently, $m \leq (4\varepsilon)^{2/3} N^{1/3}$, where $(4\varepsilon)^{2/3} \leq 1$ for all reasonable values of ε. Another observation of great importance is that a Rényi entropy of at least $\log K$ per block essentially means that each new key must be uniformly distributed on its support (even conditioned on the hash values of the previous keys). Therefore, if either

(1) a nonnegligibly sized part of the sequence of keys is determined or

(2) the key set is arbitrarily, say fully, random but $m \gg N^{1/3}$

then we cannot expect the hash values $(h(X_1), \dots, h(X_T))$ resulting from uniformly random choosing h from a 1-universal class to be close to the uniform distribution on M^T, and hence we cannot apply any of the nice results for the case of full randomness. For each of our Theorems 3, 4, 5, and 6, as well as for our Corollaries 1, 2, 3, 4, 5, and 6, either (1) or (2) holds.

2.4.6 Theorem 2 and the Multiplicative Class

In [15] and [16] the authors suggested that a theorem of comparable strength as Theorem 2 holds for 2-universal classes like $\mathcal{H}^{\text{mult}}_{k,l}$, too, but in fact they do not know whether this is true or not. However, they do know that if we apply the proof of Theorem 2 to h chosen uniformly at random from a c-universal class $\mathcal{H}$, $c \geq 1$, instead of a 1-univeral class then this leads to $\varepsilon(c) := ((T/2) \cdot$

$\sqrt{m/K + (c-1)}$)-closeness of $(h(X_1), \ldots, h(X_T))$ to the uniform distribution on M^T. (In order to see this one has to apply the proof of the Leftover-Hash-Lemma (compare, e. g., [30]) to a c-universal class instead of a just 1-universal, or even pairwise independent, class at first.) The crux is that for every $\varepsilon > 0$ we can choose K large enough compared to m such that $\varepsilon(1) \leq \varepsilon$ whereas $\varepsilon(c) > T/2$ for all $m, K \in \mathbb{N}$ and $c \geq 2$.

CHAPTER 3

The Case of Dense Key Sets

In this chapter we consider the case where n/N resp. m/N is at least $N^{\gamma-1}$ for a suitable constant $\gamma \in (0,1)$, i. e. key sets of size n are relatively dense in U. In Section 3.2 we prove a lower bound on the failure probability when a uniformly random chosen set $S \subseteq U$ of size no more than $m/2$ is inserted with arbitrarly fixed multiplicative functions. This implies the existence of a "bad set" S of size $m/2$ such that the same lower bound on the failure probability holds if S is inserted with uniformly random chosen multiplicative functions. Similar results for the linear class are established in Section 3.3. Even in case the two linear functions use different prime moduli there is no significant improvement of the performance, as shown in Section 3.4. Section 3.1 deals with a technical condition of the main theorems.

3.1 The Special Case $\frac{m}{N} \geq \frac{1}{2}$

This brief section deals with the special case of very dense key sets, in order to explain a technical condition of the following theorems. It turns out that the performance of cuckoo hashing combined with the multiplicative or linear class is best possible if m/N is at least $1/2$. Note that throughout we focus on m/N (rather than n/N), because it is this ratio that determines the structure of the hash functions.

Proposition 1. *If $m/N \geq 1/2$, then all functions $h_1, h_2 \in \mathcal{H}^{mult}_{k,l}$ are suitable for all sets $S \subseteq U$. The same holds for $\mathcal{H}^{lin*}_{p,m}$.*

Proof. It suffices to show that in these cases the complete cuckoo graph $G = G(U, h_1, h_2)$ has maximum degree of 2, i.e., its components are simple paths and simple cycles. It is clear that in this situation the keys can be arranged as required.

As for $\mathcal{H}^{\text{mult}}_{k,l}$, note that $O_k = \{1, 3, \ldots, 2^k - 1\}$ is an Abelian group with respect to multiplication modulo 2^k. So, for each $a \in O_k$ the mapping

$$x \mapsto ax \bmod 2^k$$

is a permutation of $U = [2^k]$, and its restriction to O_k is a permutation of O_k. The assumption $m/N = 2^{l-k} \geq 1/2$ implies $k - l \in \{0, 1\}$, and hence

$$x \mapsto (ax \bmod 2^k) \text{ div } 2^{k-l}$$

is 1-1 on O_k, and 1-1 on $U - O_k$. Consequently, for all $j \in [m]$ there are at most 2 keys x with $h_a(x) = j$, and hence G has maximum degree 2.

The argumentation for $\mathcal{H}^{\text{lin}*}_{p,m}$ is similar: As $[p]$ is a field w.r.t. addition and multiplication modulo p the mapping

$$x \mapsto (ax + b) \bmod p$$

is a permutation of $U = [p]$. The assumption $m/N \geq 2$ can be read as $m \geq N/2$, and hence for all $j \in [m]$ there are at most 2 keys x with

$$h_{a,b}(x) = (ax + b) \bmod p \bmod m = j\,.$$

□

3.2 High Failure Probability for the Multiplicative Class

We consider the multiplicative class. Recall that, for fixed hash functions h_1, h_2, and $S \subseteq U$ chosen randomly, we denote the probability that h_1 and h_2 are not suitable for S as *failure probability* p_{F}. The purpose of this section is to establish the following theorem.

Theorem 3. *Let $h_{a_1}, h_{a_2} \in \mathcal{H}^{mult}_{k,l}$ be arbitrary, and let a set $S \subseteq U = [2^k]$ of size $m/2 = 2^{l-1}$ be chosen uniformly at random. If $l \leq k - 2$ and $l/k > 11/12$, then $p_F = 1 - o(1)$, for $l \to \infty$.*

Note that the number $m/2$ of keys in S is way below the threshold for cuckoo hashing with random sets in the case of $\Omega(\log n)$-wise independent functions, which permits sizes up to $(1-\delta)m$ for an arbitrary constant $\delta > 0$. The case $l > k-2$ was considered in Proposition 1.

Once we have proved Theorem 3 we directly obtain the existence of a *bad set* S in the following sense.

Corollary 1. *Let $l \leq k-2$ and $l/k > 11/12$. Then there exists a set $S \subseteq U = [2^k]$ of size $m/2$ such that p_F is $1-o(1)$, for $l \to \infty$, if $h_{a_1}, h_{a_2} \in \mathcal{H}^{mult}_{k,l}$ are chosen uniformly at random.*

This is a direct implication of the following lemma, where a universe U and a family of hash functions $\mathcal{H}$ are fixed arbitrarily.

Lemma 4. *If for arbitrary hash functions $h_1, h_2 \in \mathcal{H}$ and a set $S \subseteq U$, $|S| = n$, chosen uniformly at random, and an arbitrarily fixed probability $\rho \in (0,1)$ we have $p_F(S, h_1, h_2) \geq \rho$, then there exists a set $S' \subseteq U$, $|S'| = n$, such that for hash functions $h'_1, h'_2 \in \mathcal{H}$ chosen uniformly at random we have $p_F(S', h'_1, h'_2) \geq \rho$.*

Proof (of Lemma 4). Consider a matrix M_F with components in $\{0,1\}$ where each row corresponds to a unique choice of a hash function pair (h_1, h_2), and each column corresponds to a unique choice of a set $S \subseteq U$. Let the component in row (h_1, h_2) and column S be 1 iff the insertion of S by means of the hash functions (h_1, h_2) fails. Then, $p_F = \rho$ is equivalent to a ratio of at least ρ 1s in M_F. This implies, there must be a column such that the ratio of 1s in it is at least ρ. In other words, there exists a set S' such that for uniformly random chosen hash functions in $\mathcal{H}$ the failure probability is at least ρ. □

Proof (of Theorem 3). The general idea is to show that the complete cuckoo graph $G(U, h_{a_1}, h_{a_2})$, referred to as G in the following, contains many bad edge sets of constant size, of which any two do not overlap too much, and to conclude that the subgraph of G that corresponds to a randomly chosen set $S \subseteq U$ is very likely to contain one of these bad edge sets.

Lemma 5. *The complete cuckoo graph $G(U, h_{a_1}, h_{a_2})$ contains a set $\{K_i \mid i \in [m]\}$ of m distinct bad edge sets of size ≤ 10 such that for all $i \in [m]$ we have*

$$|\{j \in [m] \mid K_j \cap K_i \neq \emptyset\}| \leq 13\,.$$

Proof. For analyzing the structure of G we may assume that $a_1 = 1$, as the following lemma shows.

Lemma 6. *The set $\{G(U, h_{a_1}, h_{a_2}) \mid a_2 \in O_k\}$ of complete cuckoo graphs for fixed a_1 and variable a_2 does not depend on a_1. The same holds for $\{G(O_k, h_{a_1}, h_{a_2}) \mid a_2 \in O_k\}$.*

Proof. Let r denote the mapping

$$x \mapsto a_1 x \bmod 2^k .$$

As a_1 is odd, we have for the edge set E of $G(D, h_{a_1}, h_{a_2})$, $D \in \{U, O_k\}$:

$$\begin{aligned} E &= \{(h_{a_1}(x), h_{a_2}(x)) \mid x \in D\} \\ &= \{(h_1(r(x)), h_{a_2 a_1^{-1}}(r(x))) \mid x \in D\} , \end{aligned}$$

where the mapping

$$a_2 \mapsto a_2 a_1^{-1} \bmod 2^k$$

is a permutation of O_k. It remains to observe that r is a permutation of U, and its restriction to O_k is a permutation of O_k. □

So, for the proof of Lemma 5 we assume $a_1 = 1$, and consider $G = G(U, h_1, h_{a_2})$. We partition U into "grid sets" $G_m(c)$, $c \in [2^{k-l}]$, that are defined as follows:

$$G_m(c) := \{x_i(c) \mid i \in [m]\} , \tag{3.2.1}$$

where $x_i(c) := (c + i \cdot 2^{k-l}) \bmod 2^k$. A straightforward calculation proves the following.

Lemma 7. *For each $h_a \in \mathcal{H}^{mult}_{k,l}$ and for all $i \in [m]$ we have*

$$h_a(x_i(c)) = (h_a(c) + i \cdot a) \bmod 2^l .$$

Proof. The calculation makes use of the following three equations, which hold for arbitrary natural numbers x and y, and whose correctness is immediate when the numbers are represented in binary:

$$(x \cdot 2^{k-l}) \bmod 2^k = (x \bmod 2^l) \cdot 2^{k-l} , \tag{3.2.2}$$

$$(x \bmod 2^k) \operatorname{div} 2^{k-l} = (x \operatorname{div} 2^{k-l}) \bmod 2^l , \tag{3.2.3}$$

$$x \operatorname{div} 2^{k-l} + (y \cdot 2^{k-l}) \operatorname{div} 2^{k-l} = (x + y \cdot 2^{k-l}) \operatorname{div} 2^{k-l} . \tag{3.2.4}$$

Now, the calculation is as follows, where the fourth equation makes use of (3.2.2) and (3.2.3), and the fifth equation applies (3.2.4).

$$\begin{aligned} h_a(x_i(c)) &= (a \cdot x_i(c)) \bmod 2^k \operatorname{div} 2^{k-l} \\ &= (a \cdot (c + i \cdot 2^{k-l})) \bmod 2^k \operatorname{div} 2^{k-l} \\ &= (ac \bmod 2^k + (ai \cdot 2^{k-l}) \bmod 2^k) \bmod 2^k \operatorname{div} 2^{k-l} \\ &= (ac \bmod 2^k + (ai \bmod 2^l) \cdot 2^{k-l}) \operatorname{div} 2^{k-l} \bmod 2^l \\ &= (h_a(c) + ai \bmod 2^l) \bmod 2^l \\ &= (h_a(c) + i \cdot a) \bmod 2^l \,. \end{aligned}$$

□

In other words, the image of $G_m(c)$ under h_a is also a grid set and the increment is the same for all c. Moreover, we shall see that the edge set corresponding to $G_m(c)$ in $G = (V_1, V_2, E)$ is a perfect matching with a structure that makes it possible to find many bad edge sets. If we denote the edge that corresponds to key $x_i(c)$ by $e_i(c)$, then $e_i(c)$ is incident with $i \in V_1$, as shown by the following calculation which uses Lemma 7, (3.2.3) and the fact that $c < 2^{k-l}$ implies $h_1(c) = 0$:

$$\begin{aligned} e_i(c) &= (h_1(x_i(c)), h_{a_2}(x_i(c))) \\ &= ((h_1(c) + i) \bmod 2^l, (h_{a_2}(c) + i \cdot a_2) \bmod 2^l) \\ &= (i, (a_2 c \operatorname{div} 2^{k-l} + i \cdot a_2) \bmod 2^l) \,. \end{aligned}$$

For notational convenience, we consider a graph $\tilde{G} = (V_1, V_2, \tilde{E})$ derived from G by permuting V_2. Precisely,

$$\tilde{E} = \{(h_1(x), (h_{a_2}(x) \cdot a_2^{-1}) \bmod 2^l) \mid x \in U\} \,,$$

where a_2^{-1} denotes the multiplicative inverse of a_2 modulo 2^k (and hence in particular we have $a_2 a_2^{-1} \bmod 2^l = 1$ for $l < k$). Obviously, $\tilde{G}$ is isomorphic to G, and hence bad edge sets in $\tilde{G}$ correspond to bad edge sets in G. In $\tilde{G}$, edge $\tilde{e}_i(c)$ of key $x_i(c)$ is incident with $i \in V_1$ and with $i + o_c \in V_2$ for a constant

$$o_c := (a_2 c \operatorname{div} 2^{k-l} \cdot a_2^{-1}) \bmod 2^l \,,$$

$i \in [m]$. So, the edge set corresponding to $G_m(c)$ forms a perfect matching in $\tilde{G}$, and hence also in G, which in $\tilde{G}$ can be considered as a wheel (cyclic modulo m) of parallel line segments with slope o_c (Fig. 3.1).

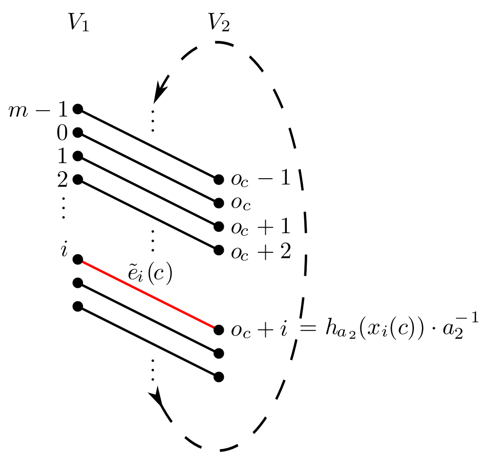


Figure 3.1: The edges corresponding to $G_m(c)$ form a perfect matching in $\tilde{G}$ which can be considered as a wheel of parallel line segments.

For any two distinct keys $c, c' < 2^{k-l}$, consider the edge set $C_i(c, c')$ defined as

$$\{\tilde{e}_i(c), \tilde{e}_{o_c+i}(0), \tilde{e}_{o_c+i}(c'), \tilde{e}_{o_{c'}+i}(c), \tilde{e}_{o_{c'}+i}(0), \tilde{e}_i(c')\} ,$$

of size at most 6, where arithmetic modulo m has to be applied to all indices (Fig. 3.2).

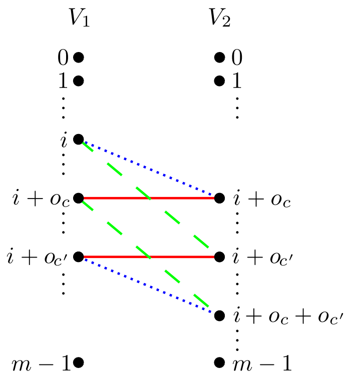
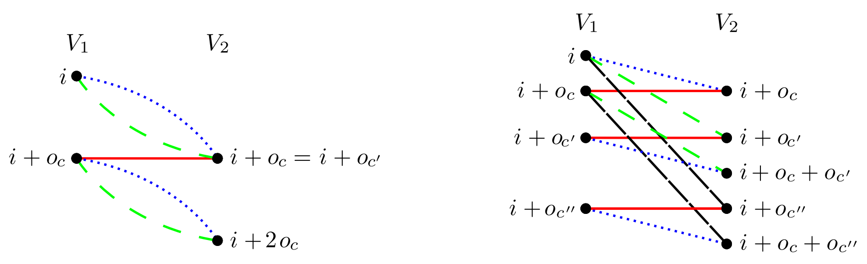


Figure 3.2: A cycle $C_i(c, c')$ in $\tilde{G}$

Now fix any three distinct elements $c, c', c'' \in [2^{k-l}]$. This is possible since $k - l$ is at least 2. If two of the offsets $o_c, o_{c'}, o_{c''}$ are equal, say $o_c = o_{c'}$, then $C_i(c, c')$ is a bad edge set $\tilde{K}_i$ of size 5, $i \in [m]$, since $\tilde{e}_{o_c+i}(0) = \tilde{e}_{o_{c'}+i}(0)$ (Fig. 3.3(a)). Otherwise, $C_i(c, c')$ and $C_i(c, c'')$ are cycles of size 6 that overlap in two edges, and their union is a bad edge set $\tilde{K}_i$ of size 10, $i \in [m]$ (Fig. 3.3(b)).

Note that each of the bad edge sets $\tilde{K}_i$, $i \in [m]$, contains at most four distinct vertices in V_1, and $\tilde{K}_j$, $j \in [m]$, is a copy of $\tilde{K}_i$, shifted modulo m. Therefore, the number of distinct bad edge sets $\tilde{K}_j$, $j \in [m]$, (where $j = i$ is included) overlapping with $\tilde{K}_i$ is at most 13, because a necessary condition for $\tilde{K}_j \cap \tilde{K}_i \neq \emptyset$



(a) $o_c = o_{c'}$ (b) $o_c, o_{c'}, o_{c''}$ distinct

Figure 3.3: Bad edge sets for the multiplicative class

is a common vertex in V_1. Now, letting K_i denote the bad edge set in G that corresponds to $\tilde{K}_i$, respectively, completes the proof of Lemma 5. □

By Lemma 5, we may fix a set $\{K_i \mid i \in [m]\}$ of m distinct bad edge sets of size at most 10 such that for all $i \in [m]$ we have $|\{j \in [m] \mid K_j \cap K_i \neq \emptyset\}| \leq 13$. Let $S \subseteq U$, $|S| = m/2$, be chosen uniformly at random. Choosing S directly corresponds to choosing n edges in G at random. Let the random variable X_i, $i \in [m]$, take the value 1 if all edges of K_i have been chosen, 0 otherwise, and define $X := \sum_{i \in [m]} X_i$. Then

$$p_{\mathrm{F}} \geq \Pr(X > 0)\,.$$

Note that the X_i are not independent.

In order to establish a lower bound on $\Pr(X > 0)$, we invoke Lemma 3 and apply it for $(Y_1, \ldots, Y_t) = (X_0, \ldots, X_{m-1})$ and $Y = X$. As $\Pr(X_i = 1)$ only depends on $|K_i|$, we obtain under the given conditions $l \leq k - 2$ and $l/k > 11/12$:

$$\begin{aligned}\Pr(X_i = 1) &\geq \binom{2^k - 10}{2^{l-1} - 10} \Big/ \binom{2^k}{2^{l-1}} \\ &> 2^{-10(k-l+2)}\,.\end{aligned}$$

Furthermore, distinguishing between bad edge sets K_j that overlap with K_i and those which do not yields $\mathrm{E}(X) + 13$ as an upper bound for $\mathrm{E}(X \mid X_i = 1)$. Putting it all together, we get

$$p_{\mathrm{F}} > \left(1 + 2^{4+10(k-l+2)-l}\right)^{-1}\,.$$

For $l/k > 11/12$, the latter term is at least $\left(1 + 2^{-(1/12)\cdot k + 24}\right)^{-1}$, and hence $p_{\mathrm{F}} = 1 - o(1)$ for $k, l \to \infty$. This concludes the proof of Theorem 3. □

3.3 High Failure Probability for the Linear Class

In this section, we prove a theorem for the linear class, in analogy to Theorem 3. Note that the linear class is *very* standard.

Theorem 4. *Let $h_{a_1,b_1}, h_{a_2,b_2} \in \mathcal{H}^{lin*}_{p,m}$ be arbitrary, and let a set $S \subseteq U = [p]$ of size $\lceil m/2 \rceil$ be chosen uniformly at random. If $m/p \in [p^{-(1/7-\varepsilon)}, 1/7]$ for a constant $\varepsilon \in (0, 1/7)$ then $p_F = 1 - o(1)$, for $m \to \infty$.*

Together with Lemma 4, Theorem 4 implies the existence of a bad set.

Corollary 2. *Let $m/p \in [p^{-(1/7-\varepsilon)}, 1/7]$ for a constant $\varepsilon \in (0, 1/7)$. Then there exists a set $S \subseteq U = [p]$ of size $\lceil m/2 \rceil$ such that $p_F = 1 - o(1)$, for $m \to \infty$, if $h_{a_1,b_1}, h_{a_2,b_2} \in \mathcal{H}^{lin*}_{p,m}$ are chosen uniformly at random.*

Proof (of Theorem 4). The general approach is the same as in the proof of Theorem 3, but the details differ considerably. Fix $h_{a_1,b_1}, h_{a_2,b_2} \in \mathcal{H}^{\text{lin}*}_{p,m}$ and consider the complete cuckoo graph $G = G(U, h_{a_1,b_1}, h_{a_2,b_2})$.

Lemma 8. *Let $m' = \lceil m/3 \rceil$. If $m/p \leq 1/7$, then G contains a set $\{K_i \mid i \in [m']\}$ of m' distinct bad edge sets of size 6 such that for all $i \in [m']$ we have*

$$|\{j \in [m'] \mid K_j \cap K_i \neq \emptyset\}| \leq 5 .$$

Proof. By a lemma and proof that is similar to Lemma 6 and its proof we may assume w. l. o. g. that $h_{a_1,b_1} = h_{1,0}$, where $h_{1,0}(x) = x \bmod m$:

Lemma 9. *The set $\{G(U, h_{a_1,b_1}, h_{a_2,b_2}) \mid a_2 \in [p] - \{0\}, b_2 \in [p]\}$ of complete cuckoo graphs for fixed a_1, b_1 and variable a_2, b_2 does not depend on a_1, b_1.*

Proof. Let r denote the mapping

$$x \mapsto (a_1 x + b_1) \bmod p .$$

As $a_1 > 0$, we have for the edge set E of $G(U, h_{a_1,b_1}, h_{a_2,b_2})$:

$$\begin{aligned} E &= \{(h_{a_1,b_1}(x), h_{a_2,b_2}(x)) \mid x \in U\} \\ &= \{(h_{1,0}(r(x)), h_{a_2 a_1^{-1}, b_2 - a_2 a_1^{-1} b_1}(r(x))) \mid x \in U\} , \end{aligned}$$

where the mapping

$$a_2 \mapsto a_2 a_1^{-1} \bmod p$$

is a permutation of $[p] - \{0\}$, and

$$b_2 \mapsto b_2 - a_2 a_1^{-1} b_1 \bmod p$$

is a permutation of $[p]$. It remains to observe that r is a permutation of U. $\square$

Thus, consider $G = G(U, h_{1,0}, h_{a,b}) = (V_1, V_2, E)$ for an arbitrary $h_{a,b} \in \mathcal{H}^{\text{lin}*}_{p,m}$. We study the neighborhood $\Gamma_j = h_{a,b}(h_{1,0}^{-1}(j)) \subseteq V_2$ of an arbitrary vertex $j \in V_1$. Every key x whose corresponding edge e_x is incident with $j \in V_1$ is in $h_{1,0}^{-1}(j)$, and hence has the form $x = im + j$ for some $i \in \mathbb{N}$. Note that $i \leq t := \lceil p/m \rceil$, and the degree of any vertex in G is either t or $t-1$. We refer to a key $im + j$ as $x_i(j)$, call the corresponding edge the *i-edge of j*, and refer to its endpoint in V_2 as the *i-neighbor of j*, or as $n_i(j)$ (Fig. 3.4). The following lemma will help us to understand the structure of Γ_j.

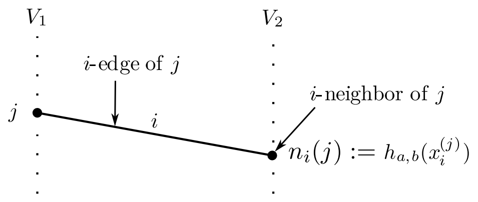


Figure 3.4: the i-edge and the i-neighbor of $j \in V_1$

Lemma 10 (leap effect)**.** *Let r_1 and r_2 be fixed positive integers, and define $\Delta := (-r_1) \bmod r_2$. Then for all $x \in \mathbb{N}$ we have:*

$$(x \bmod r_1) \bmod r_2 = (x + \lfloor x/r_1 \rfloor \cdot \Delta) \bmod r_2 .$$

Proof. Straightforward induction on $x \in \mathbb{N}$:
$\underline{x = 0}$: Clear. $\underline{x \to x+1}$: We distinguish two cases. If r_1 divides $x+1$, then

$$\lfloor (x+1)/r_1 \rfloor = \lfloor x/r_1 \rfloor + 1 \quad \text{and} \quad r_1 = x \bmod r_1 + 1 .$$

Together with the induction hypothesis, this yields

$$\begin{aligned}(x+1) \bmod r_1 \bmod r_2 &= 0 \bmod r_2 \\ &= (r_1 - r_1) \bmod r_2 \\ &= ((x \bmod r_1 + 1) + \Delta) \bmod r_2 \\ &= ((x + \lfloor x/r_1 \rfloor \cdot \Delta) + 1 + \Delta) \bmod r_2 \\ &= ((x+1) + \lfloor (x+1)/r_1 \rfloor \cdot \Delta) \bmod r_2 .\end{aligned}$$

Otherwise, if r_1 does not divide $x+1$, then

$$\lfloor (x+1)/r_1 \rfloor = \lfloor x/r_1 \rfloor \quad \text{and} \quad (x+1) \bmod r_1 = (x \bmod r_1) + 1\,,$$

and an application of the induction hypothesis yields

$$\begin{aligned}(x+1) \bmod r_1 \bmod r_2 &= (x \bmod r_1 + 1) \bmod r_2\\ &= ((x + \lfloor x/r_1 \rfloor \cdot \Delta) + 1) \bmod r_2\\ &= ((x+1) + \lfloor (x+1)/r_1 \rfloor \cdot \Delta) \bmod r_2\,.\end{aligned}$$

□

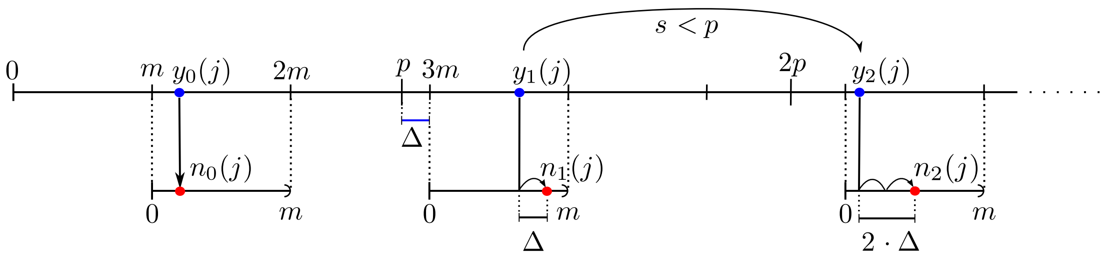


Figure 3.5: Application of the leap effect, where we have $n_i(j) = h_{a,b}(x_i(j)) = (y_i(j) \bmod p) \bmod m$ for $y_i(j) = i \cdot (am \bmod p) + (aj + b) \bmod p$.

We want to simplify the term $n_i(j) = h_{a,b}(x_i(j))$. Applying Lemma 10 for $r_1 = p$ and $r_2 = m$ as depicted in Fig. 3.5 leads to the basic observation that Γ_j is nearly a grid, more precisely: For each $i \in [t-2]$ we have

$$n_{i+1}(j) \in \{(n_i(j) + s') \bmod m, (n_i(j) + s'') \bmod m\}\,,$$

where $s' = s \bmod m$, $s'' = (s + \Delta) \bmod m$, $s = am \bmod p$, and $\Delta = (-p) \bmod m$.

For each i-edge of j with $i \in [t-2]$, there is still an $(i+1)$-edge of j. So, call the former a *predecessor edge*, and the latter its *successor edge*. Furthermore, call a vertex in V_2 *obstructive* if it is incident with at least five predecessor edges. Let l be an obstructive vertex and fix any five of its incident predecessor edges. Their respective successor edges are incident with $(l + s') \bmod m$ or $(l + s'') \bmod m$ in V_2, and therefore at least three of these successor edges must have the same endpoint $k \in V_2$. Fix three of the successor edges with endpoint k. Together with their predecessor edges, they form a bad edge set $K^{(l)}$ of size 6 (Fig. 3.6).

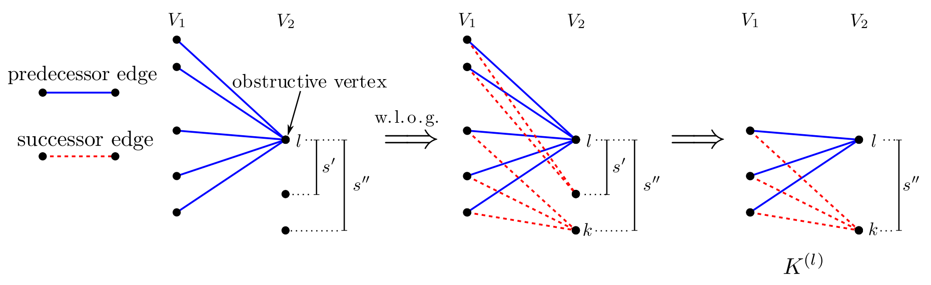


Figure 3.6: Bad edge sets for the linear class

If l and l' are distinct obstructive vertices, then their respective bad edge sets $K^{(l)}$ and $K^{(l')}$ must also be distinct, because otherwise there would be a predecessor edge which is its own successor edge, which is impossible. Moreover, if we fix $K^{(l)}$ for every obstructive vertex l, then for each obstructive vertex l, we have

$$|\{l' \in V_2 \mid l' \text{ obstructive vertex}, K^{(l)} \cap K^{(l')} \neq \emptyset\}| \leq 5\,,$$

as a necessary condition for $K^{(l)} \cap K^{(l')} \neq \emptyset$ is a common vertex in V_2.

It remains to show that there are $m' = \lceil m/3 \rceil$ obstructive vertices. The number of predecessor edges in G is $(t-2)m$, where the assumption $m/p \leq 1/7$ implies $t-2 \geq 5$. With respect to maximizing the number of obstructive vertices, the worst case is $t-2=5$. Now it suffices to use the fact that the degree of each vertex of G is at most t. □

We complete the proof of Theorem 4 in a way similar to the proof of Theorem 3: Lemma 8 guarantees the existence of a set $\{K_i \mid i \in [m']\}$ of bad edge sets in G. We fix such a set. Choose $\lceil m/2 \rceil$ edges from G uniformly at random. Define 0-1 random variables $X_0, \ldots, X_{m'-1}$ as follows: $X_i = 1$ if and only if all edges of K_i are chosen, and let $X = \sum_{i \in [m']} X_i$. Clearly, $p_{\mathrm{F}} \geq \Pr(X > 0)$. Under the given condition $m/p \geq p^{-(1/7-\varepsilon)}$, $\varepsilon \in (0, 1/7)$, an application of the conditional expectation inequality (Lemma 3) yields the following lower bound for $\Pr(X > 0)$:

$$\left(1 + 2^6 \cdot 3 \cdot 5 \cdot \left(1 - \frac{10}{p^{6/7+\varepsilon}}\right)^{-6} \cdot p^{-7\varepsilon}\right)^{-1} = 1 - o(1)\,,$$

for $p, m \to \infty$. □

3.4 High Failure Probability for Two Distinct Linear Classes

Again, we consider the linear class. It might seem plausible that the performance strongly improves if h_1 and h_2 are chosen from linear classes over fields given by distinct prime numbers p_1 and p_2, respectively. It is the purpose of this section to show that this is not the case.

Theorem 5. *Let h_{a_1,b_1} and h_{a_2,b_2} in $\mathcal{H}^{lin*}_{p_1,m}$ and $\mathcal{H}^{lin*}_{p_2,m}$, resp., where $p_1 \le p_2 \le \alpha p_1$ holds for an arbitrary constant $\alpha \ge 1$, and let a set $S \subseteq U = [p_1]$ of size $\lceil m/2 \rceil$ be chosen uniformly at random. If $m/p_1 \in [p_1^{-(1/8-\varepsilon)}, 1/73]$ for a constant $\varepsilon \in (0, 1/8)$, then $p_F = 1 - o(1)$, for $m \to \infty$.*

An application of Lemma 4 leads to the following.

Corollary 3. *Let $p_1 \le p_2 \le \alpha p_1$ and $m/p_1 \in [p_1^{-(1/8-\varepsilon)}, 1/73]$ for arbitrarily fixed constants $\varepsilon \in (0, 1/8)$ and $\alpha \ge 1$. Then there exists a set $S \subseteq U = [p_1]$ of size $\lceil m/2 \rceil$ such that $p_F = 1 - o(1)$, for $m \to \infty$, if h_{a_1,b_1} and h_{a_2,b_2} in $\mathcal{H}^{lin*}_{p_1,m}$ and $\mathcal{H}^{lin*}_{p_2,m}$, respectively, are chosen uniformly at random.*

Proof (of Theorem 5). The basic approach is the same as before. Fix arbitrary hash functions h_{a_1,b_1} and h_{a_2,b_2} in $\mathcal{H}^{\text{lin}*}_{p_1,m}$ and $\mathcal{H}^{\text{lin}*}_{p_2,m}$, respectively. Then there exist many bad edge sets of constant size, of which very few pairs have a common edge.

Lemma 11. *Let $m' := \lceil m/(\alpha(t+1)) \rceil$ for $t := \lceil p_1/m \rceil - 1$. If $m/p_1 \le 1/73$, then $G(U, h_{a_1,b_1}, h_{a_2,b_2})$ contains a set $\{K_i \mid i \in [m']\}$ of m' distinct bad edge sets of size 6 such that for each $i \in [m']$ we have $|\{j \in [m'] \mid K_j \cap K_i \ne \emptyset\}| \le 5$.*

Proof. In contrast to the proof of Lemma 8, we cannot assume that $h_{a_1,b_1} = h_{1,0}$, because of the distinct moduli of h_{a_1,b_1} and h_{a_2,b_2}. However, we may get rid of one modulus with the help of the following lemma, which says that a grid set modulo p contains a long arithmetic sequence in $\mathbb{N}$.

Lemma 12. *Assume p is a prime and $L, \tilde{s} \in [p]$ are arbitrary, where $L > 2$ and $\tilde{s} > 0$. Then there exists an $\hat{s} \in [p]$, $0 < \hat{s} \le p/\sqrt{L/2}$, such that for all $c \in [p]$ the grid set $B := \{x_i \mid i \in [L]\}$, where $x_i := (i \cdot \tilde{s} + c) \bmod p$, of size L contains an arithmetic* *sequence* $(\hat{x}_k)_{k\in[l]}$ *with step size* $\hat{s}$ *and length* $l = \lfloor\sqrt{L/2}\rfloor$*, i. e., for* $k = 0, 1, \ldots, l-2$ *we have* $\hat{x}_{k+1} = \hat{x}_k + \hat{s} = \hat{x}_0 + (k+1)\cdot\hat{s}$*.*

Proof. For each pair of distinct keys $x_i, x_j \in B$, define

$$\begin{aligned}&A_{i,j} := \{x_{i+z\cdot d} \in B \mid d = |j-i|, z \in \mathbb{Z}, i + z\cdot d \in [L]\}\\ &\text{and } s_{i,j} := \min\{|x_j - x_i|, p - |x_j - x_i|\}\ .\end{aligned}$$

The subset $A_{i,j}$ can be viewed as a sequence $(y_k)_{k\in[l']}$ of length $l' \geq \lfloor L/d\rfloor$, which increases by $s_{i,j}$ in one step cyclically modulo p. Moreover, $s_{i,j}$ does not depend on the offset c of B. Now assume for the time being $l' \geq \lfloor\sqrt{2L}\rfloor$, i. e., $l' \geq 2\lfloor\sqrt{L/2}\rfloor = 2l$, and $s_{i,j} \leq p/\sqrt{L/2}$. Then $(y_k)_{k\in[l']}$ contains the desired arithmetic sequence $(\hat{x}_k)_{k\in[l]}$ with step size $\hat{s} := s_{i,j} \leq p/\sqrt{L/2}$: Let w' be the smallest index $w \in [l'] - \{0\}$ such that $y_{w-1} > y_w$. If no such w exists, we are done. Otherwise, for each $k \in [l]$, set

$$\hat{x}_k := \begin{cases} y_k & \text{if } w' \geq l \\ y_{w'+k} & \text{otherwise}\ . \end{cases}$$

Regard B as a point set $B_2 := \{(i, x_i) \mid i \in [L]\}$ in the half-open rectangle $Q := [0, L) \times [0, p) \subseteq \mathbb{R}^2$ (Fig. 3.7). Let $J \times K \subseteq Q$ for cyclic intervals $J := [l_1, r_1)$ and $K := [l_2, r_2)$, i. e., J and K may be wrapped around the boundaries of $[0, L)$ and $[0, p)$, respectively, and hence we have $|J| := (r_1 - l_1) \bmod L$ and $|K| := (r_2 - l_2) \bmod p$ for the lengths of J and K (Fig. 3.8).

Observe the following. If $|J| \leq \sqrt{L/2}$, $|K| \leq p/\sqrt{L/2}$, and $J \times K$ contains $(i, x_i), (j, x_j) \in B_2$, $i \neq j$, then $A_{i,j}$ yields the desired sequence $(y_k)_{k\in[l']}$ of length $l' \geq \lfloor L/|J|\rfloor \geq \lfloor\sqrt{2L}\rfloor$ with a step size $s_{i,j} \leq |K| \leq p/\sqrt{L/2}$ (compare Fig. 3.8).

It remains to show that a rectangle $J \times K$ with the above-mentioned properties exists, i. e., $|J| \leq \sqrt{L/2}$, $|K| \leq p/\sqrt{L/2}$, and $J \times K$ contains $(i, x_i), (j, x_j) \in B_2$, $i \neq j$. Assume this is not the case. Then define half-open rectangles

$$Q_i := [i, (i + \sqrt{L/2}) \bmod L) \times [x_i, (x_i + p/\sqrt{L/2}) \bmod p)$$

for $i \in [L]$. By our assumption, the rectangles $Q_0, \ldots, Q_{L-1}$ are pairwise disjoint. This implies that the area of $\bigcup_{i\in[L]} Q_i$ is equal to the area of Q, and hence $Q = \bigcup_{i\in[L]} Q_i$. Consider Q_0. Its bottom left corner is $(0, x_0) \in \mathbb{N}^2$.

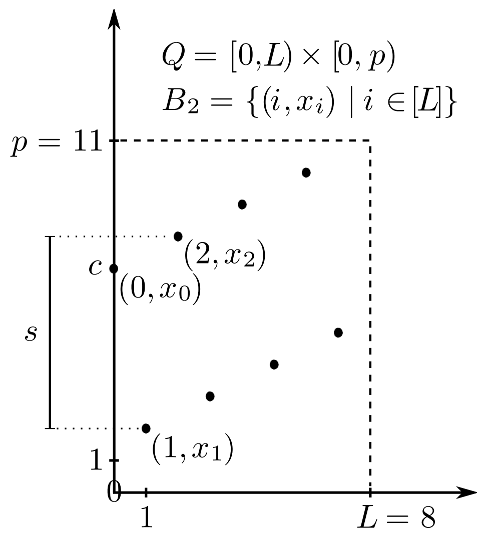


Figure 3.7: B_2

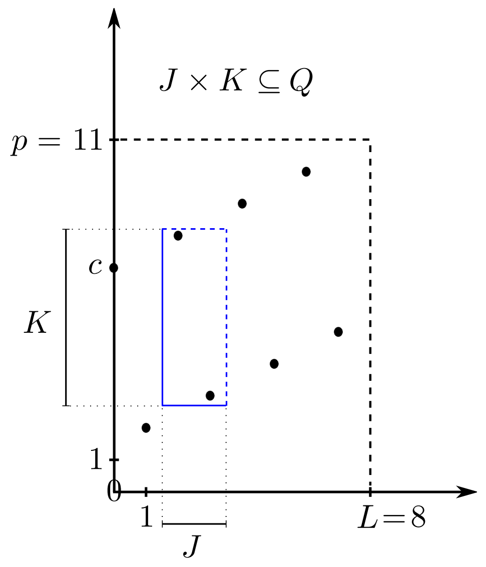


Figure 3.8: Suitable rectangle $J \times K \subseteq Q$ yields sufficiently long sequence (y_k)

Our observation implies that, cyclically modulo L and p, there must be neighboring rectangles all around Q_0 that on the one hand do not overlap with Q_0 and on the other hand touch its borders. That is, there must be a rectangle with bottom left corner $(\sqrt{L/2}, \cdot)$ and another one with bottom left corner $(\cdot, (x_0 + p/\sqrt{L/2}) \bmod p)$, which in particular implies that $\sqrt{L/2}$ and $p/\sqrt{L/2}$ are in $\mathbb{N}$. For $L > 2$ and a prime number p, this is impossible. This completes the proof of Lemma 12. $\square$

As for the proof of Lemma 11, Lemma 12 makes it possible to show for the complete cuckoo graph $G := (V_1, V_2, E) := G(U, h_{a_1,b_1}, h_{a_2,b_2})$ that the neighborhood $\Gamma_j \subseteq V_2$ of a vertex $j \in V_1$ contains a subset $\hat{\Gamma}_j$ of size $\lfloor\sqrt{t/2}\rfloor$, $t = \lceil p_1/m \rceil - 1$, which has the same crucial property as the corresponding set in the proof of Lemma 8: $\hat{\Gamma}_j$ can be viewed as a sequence $(n_i(j))_{i \in [\lfloor\sqrt{t/2}\rfloor]}$ that increases modulo m with a step size s' and s'' for fixed values s' and s''. From here, we complete the argumentation in direct analogy to the proof of Lemma 8. The details are as follows.

Consider the complete cuckoo graph $G := G(U, h_{a_1,b_1}, h_{a_2,b_2})$ with vertex sets V_1 and V_2, and with edge set E. Observe (for later use) that each vertex in V_1 has a degree of either t or $t + 1$. We analyze the neighborhood of an arbitrary vertex $j \in V_1$. For the set $B_j := h^{-1}_{a_1,b_1}(j)$ of keys whose corresponding edges are incident with $j \in V_1$, we have

$$\begin{aligned} B_j &= \{x \in U \mid (a_1 x + b_1) \bmod p_1 \bmod m = j\} \\ &= \{x \in U \mid (a_1 x + b_1) \bmod p_1 \in \{im + j \in U \mid i \in \mathbb{N}\}\} . \end{aligned}$$

The equation $(a_1 x + b_1) \bmod p_1 = im + j$ has the unique solution

$$x = (i \cdot a_1^{-1} m + a_1^{-1}(j - b_1)) \bmod p_1 .$$

That is, B_j is a grid set

$$\{x_i^{(j)} \mid i \in \mathbb{N}, im + j \in U\} , x_i^{(j)} := (i \cdot s_1 + c^{(j)}) \bmod p_1 ,$$

with step size $s_1 := (a_1^{-1} m) \bmod p_1$ and offset $c^{(j)} := (a_1^{-1}(j - b_1)) \bmod p_1$. Note that $\{x_i^{(j)} \mid i \in [t]\}$ is a subset of B_j, and that the step size s_1 is independent of j.

If we try to compute the neighborhood $\Gamma_j := h_{a_2,b_2}(B_j)$ of j, then we have to deal with arithmetic w. r. t. three distinct moduli. However, for each $j \in V_1$, we apply Lemma 12 with $L = t$ and $B = \{x_i^{(j)} \mid i \in [t]\}$: There exists a step size $\hat{s} \in [p_1] - \{0\}$ such that each set $\{x_i^{(j)} \mid i \in [t]\} \subseteq B_j$, $j \in V_1$, contains an arithmetic sequence $(\hat{x}_k^{(j)})_{k \in [l]}$ of length $l \geq \lfloor \sqrt{t/2} \rfloor$ with step size $\hat{s}$. Now, if we restrict ourselves to considering the neighbors $\hat{\Gamma}_j$ of j that are given by the subset

$$\hat{B}_j := \{\hat{x}_k^{(j)} \mid k \in [\lfloor \sqrt{t/2} \rfloor]\}$$

of B_j, then arithmetic modulo p_1 simply drops out.

Call the edge that corresponds to a key $\hat{x}_i^{(j)} \in \hat{B}_j$ the *i-edge of j* and call its endpoint in V_2 the *i-neighbor of j*, or $n_i(j)$. Then we have

$$\begin{aligned} n_i(j) &= h_{a_2,b_2}(\hat{x}_i^{(j)}) \\ &= (a_2\hat{x}_i^{(j)} + b_2) \bmod p_2 \bmod m \\ &= (a_2(i \cdot \hat{s} + \hat{x}_0^{(j)}) + b_2) \bmod p_2 \bmod m \\ &= (i \cdot a_2\hat{s} + (a_2\hat{x}_0^{(j)} + b_2)) \bmod p_2 \bmod m\ . \end{aligned}$$

Define s as $(a_2\hat{s}) \bmod p_2$, $o^{(j)}$ as $(a_2\hat{x}_0^{(j)} + b_2) \bmod p_2$, and $y_i^{(j)}$ as $i \cdot s + o^{(j)}$. Then, by Lemma 10 ("leap effect") applied for $r_1 = p_2$ and $r_2 = m$, we have

$$\begin{aligned} n_i(j) &= y_i^{(j)} \bmod p_2 \bmod m \\ &= (y_i^{(j)} + \lfloor y_i^{(j)}/p_2 \rfloor \cdot \Delta) \bmod m\ , \end{aligned}$$

where Δ is $(-p_2) \bmod m$. So, the subsequence $(n_i(j))_{i \in [\lfloor \sqrt{t/2} \rfloor]}$ of Γ_j increases modulo m with steps of size $s' := s \bmod m$ and $s'' := (s + \Delta) \bmod m$.

Call an i-edge of j a *predecessor edge* if $i \leq \lfloor \sqrt{t/2} \rfloor - 2$, and call a vertex $l \in V_2$ *obstructive* if it is incident with at least five predecessor edges. Then the existence of a set $\{K_i \mid i \in [m']\}$ of $m' = \lceil m/(\alpha(t+1)) \rceil$ distinct bad edge sets of size 6 such that for all $i \in [m']$ the number of bad edge sets K_j, $j \in [m']$, overlapping with K_i is at most 5, follows in complete analogy to the proof of Lemma 8:

(i) Each obstructive vertex $l \in V_2$ belongs to a bad edge set $K^{(l)}$ of size 6 (Fig. 3.6), and for any other obstructive vertex $l' \in V_2$, a corresponding bad edge set $K^{(l')}$ is distinct from $K^{(l)}$.

(ii) The graph G contains at least $m' = \lceil m/(\alpha(t+1)) \rceil$ obstructive vertices. Here we use that $m/p_1 \leq 1/73$ implies $\lfloor \sqrt{t/2} \rfloor - 1 \geq 5$, and that $p_2 \leq \alpha p_1$ implies a vertex degree of at most $\alpha(t+1)$ in V_2.

(iii) We use (i) and (ii) to define the desired set $\{K_i \mid i \in [m']\}$.

□

Using the condition $m/p_1 \geq p_1^{-(1/8-\varepsilon)}$ for an arbitrary constant $\varepsilon \in (0, 1/8)$, an application of Lemma 11 that is analogous to the application of Lemma 8 in the proof of Theorem 4, leads to the desired result:

$$\begin{aligned} p_{\mathrm{F}} &\geq \left(1 + 2^7 \cdot 5 \cdot \alpha \cdot \left(1 - \frac{10}{p_1^{7/8+\varepsilon}}\right)^{-6} \cdot p_1^{-8\varepsilon}\right)^{-1} \\ &= 1 - o(1) \end{aligned}$$

for $m, p_1 \to \infty$. This completes the proof of Theorem 5. □

3.5 Experiments

We implemented cuckoo hashing in Java™ in a straightforward way, where generation of pseudo-random numbers was done via the Mersenne Twister from the colt distribution[1]. We carried out some experiments in order to obtain estimates of the failure probability by counting average failure frequencies among 5 independently and uniformly random chosen sets S of size $(1-\delta)m$, each set being inserted 10 times with independently and uniformly random chosen hash functions. This was repeated several times for different settings of the parameters k, l and p, m, respectively, where δ was fixed, as well as for different settings of δ, where k and l were fixed.

Fig. 3.9 depicts the results for the multiplicative class as well as for the reference class $\mathcal{H}^3_{p,m}$, i. e., the class of quadratic polynomials, where we fixed $k = 22$, $p = 8388593 \approx 2^{23}$ and $\delta = 0.1$, and repeated the experiment for table size $m = 2^l$, $l = 1, \ldots, 22$. Fig. 3.10 shows two results for the linear class, for fixed $p = 2097143 \approx 2^{21}$ and $\delta = 0.1$, as well as changing table size $m = \lceil p/t \rceil$, $t = 2, 3, \ldots, 129$: First, only the linear class given by p was used, and second, we used the linear class given by $p_1 = p$ for h_1 and the linear class given by $p_2 = 4194301 \approx 2^{22}$ for h_2.

[1] `http://acs.lbl.gov/~hoschek/colt/`

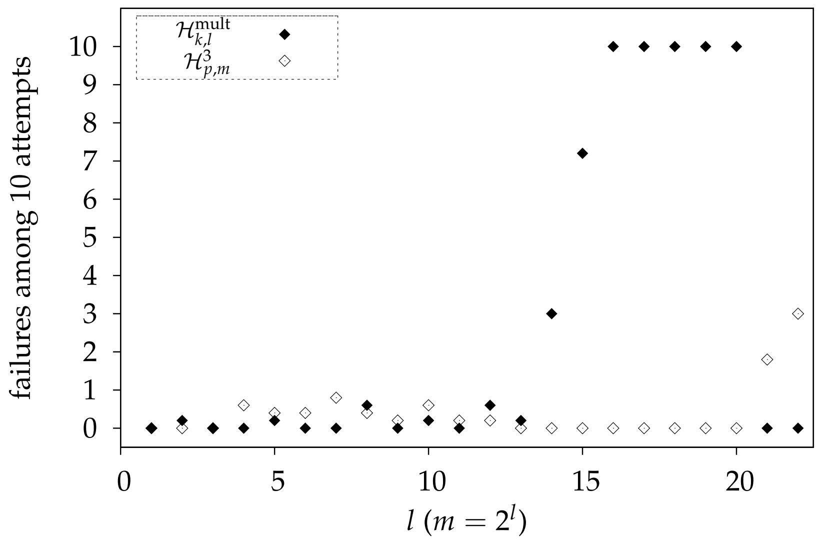


Figure 3.9: Results for the multiplicative class with fixed $k = 22$ and $\delta = 0.1$

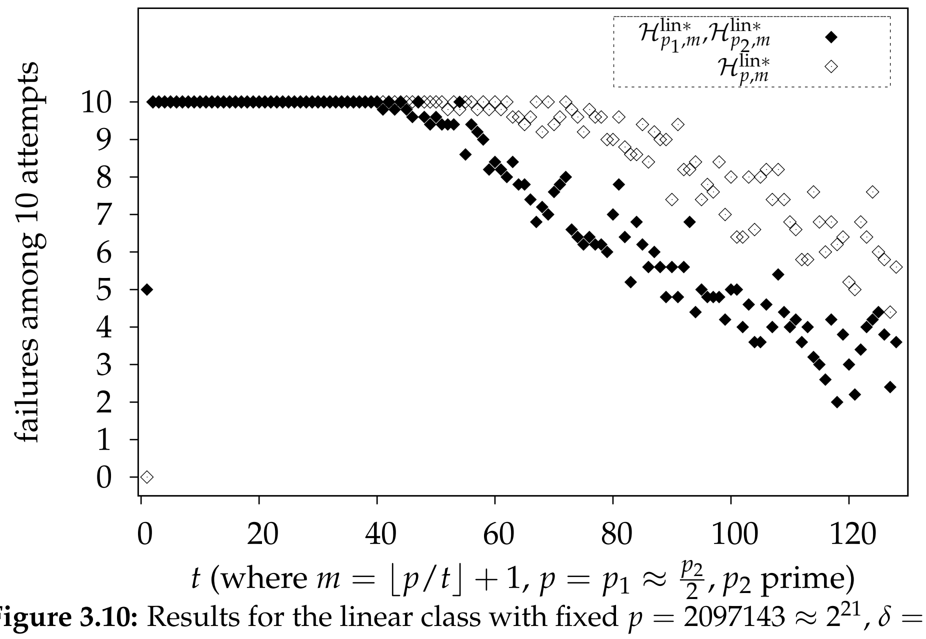


Figure 3.10: Results for the linear class with fixed $p = 2097143 \approx 2^{21}$, $\delta = 0.1$

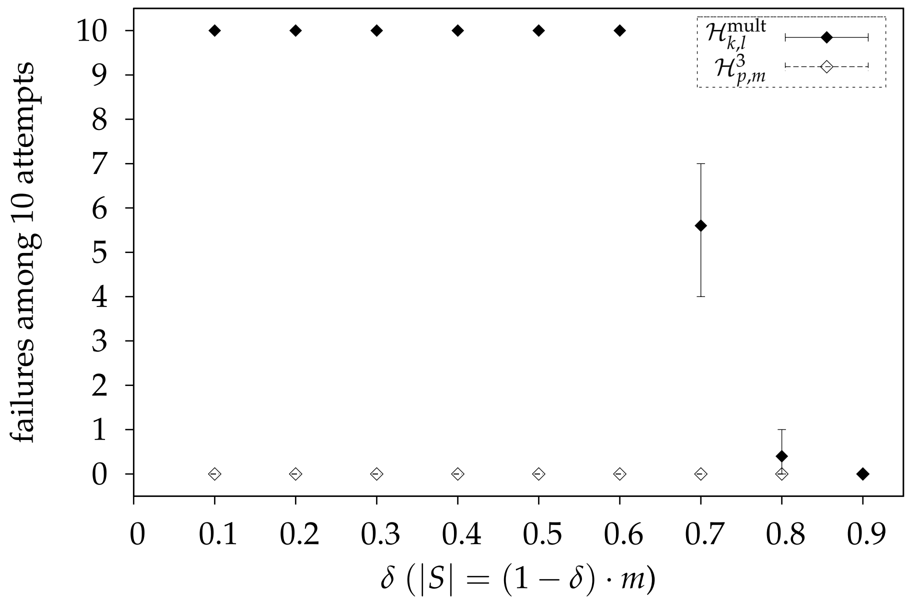


Figure 3.11: Further results for the multiplicative class with fixed $k = 24$ and $l = 21$. (The error bars show the maximum and the minimum number of failures among the five sets chosen for each value of δ.) Observe the good behavior when square polynomials are used.

In case of the hash family being the same for both hash functions, one observes: For the multiplicative class (Fig. 3.9) and the linear class (Fig. 3.10), the failure probability close to 1 for large ratio $m/N < 1/2$ nicely reflects the assertions of Theorems 3 and 4, respectively. In the case of distinct linear classes for h_1 and h_2 (Fig. 3.10) we do not see a significant performance improvement. This corresponds to Theorem 5. Note the good performance when quadratic polynomials are used (Fig. 3.9). (Note that the failure rate is indeed zero for the multiplicative class and the linear class if $m/N \geq 1/2$, as proven in Section 3.1. Astonishingly, quadratic polynomials exhibit an increased failure rate for $m/N \geq 1/2$. We do not have a reasonable explanation for this phenomenon.)

Experiments were carried out also for the multiplicative class with fixed $k = 24$, $l = 21$, and changing $\delta \in \{0.1, 0.2, \ldots, 0.9\}$. The result is depicted in Fig. 3.11. It can be seen that random key sets of relatively small size $0.4m$ still seem to be very unlikely to be inserted successfully, if the key set is dense in the universe.

CHAPTER 4

The Case of Sparse Key Sets

We consider the multiplicative class in the remaining case $m/N < N^{\gamma'-1}$ for some constant $\gamma' \in (0,1)$. We give two examples of relatively small, structured random sets for which the failure probability is high if the hash functions are chosen arbitrarily (Corollary 4) resp. uniformly at random (Theorem 6) from the multiplicative class. These random sets with high failure probability imply the existence of "bad sets." Of course we would have liked to find completely deterministic bad sets. At least the set in Theorem 6 is the result of our search for such a set. However, it seems that the closer one gets to determinism the higher becomes the complexity of bounding the failure probability, as can be seen from the complexity of the following proof.

4.1 Random Choice from a Grid

Here we show that Theorem 3 can be lifted to larger universes U by restricting the choice of S to a small subuniverse U' that has the structure of a grid. Define $U' \subseteq U = [2^k]$ as the grid set

$$\{y \cdot 2^{k-l-2} \mid y \in [2^{l+2}]\} .$$

Observe that for an odd number $a \in O_k$, the mapping $x \mapsto ax \bmod 2^k$ is a permutation of U, and that this mapping preserves the lowest order 1 bit of x. Therefore the mapping is also a permutation of U'. This implies in particular that for the hash value

$$h_a(x) = (ax \bmod 2^k) \text{ div } 2^{k-l}$$

of an element $x \in U'$ only the $l+2$ lowest order bits of the parameter a are relevant, where uniform random choice of $a \in O_k$ yields a uniform random choice of $l+2$ lowest order bits of which the lowest one is 1. So if S is chosen randomly from U' then we are in the case $l = k-2$ of Theorem 3. This leads to the following.

Corollary 4. *Let $h_{a_1}, h_{a_2} \in \mathcal{H}_{k,l}^{mult}$ be arbitrary and let a set $S \subseteq U'$ of size $m/2 = 2^{l-1}$ be chosen uniformly at random. If $l \leq k-2$ and $l/(l+2) > 11/12$, then $p_F = 1 - o(1)$, for $l \to \infty$.*

The existence of a bad set now follows as in case of all theorems in Chapter 3 from Lemma 4.

Corollary 5. *Let $l \leq k-2$ and $l/(l+2) > 11/12$. Then there exists a set $S \subseteq U'$ of size $m/2$ such that $p_F = 1 - o(1)$, for $l \to \infty$, if $h_{a_1}, h_{a_2} \in \mathcal{H}_{k,l}^{mult}$ are chosen uniformly at random.*

4.2 Random Choice of a Grid

For the second kind of "bad" random key set we use arithmetic progressions in U with step size 2^{k-l} ("grid sets") as building blocks that did already play an important role in the proof of Theorem 3 (see (3.2.1)). Let $\delta = 1/8$ and $d := \lceil (1-\delta)m/3 \rceil$. Define $x_i(c) := (c + i \cdot 2^{k-l}) \bmod 2^k$, for $c \in [2^k]$ and $i \in [2^l]$, and the grid sets

$$G_c := \{x_i(c) \mid i \in [d]\}\ , \text{for } c \in [2^k]\ .$$

To get S, we perform the following random experiment: choose c at random from O_k, and choose a random subset R_c of $U - (G_0 \cup G_c)$ of size d. Then

$$S = S(c, R_c) = G_0 \cup G_c \cup R_c\ . \tag{4.2.1}$$

The main purpose of this chapter is to establish the following theorem. Recall that the failure probability $p_F = p_F(S, h_{a_1}, h_{a_2})$ may not only depend on the random choice of S, but also on the random choice of the hash functions h_{a_1} and h_{a_2}.

Theorem 6. *If $l \geq 14$ and $k - \log k \geq 3l + 5$, then for the set $S = G_0 \cup G_c \cup R_c$ of size $3d \leq (7/8) \cdot m + 2$ formed by the random experiment as just described and for h_{a_1}, h_{a_2} chosen from $\mathcal{H}_{k,l}^{mult}$ uniformly at random we have $p_F = \Omega(1)$.*

Our next corollary, derived from Lemma 4, that states the existence of a bad set might seem not much stronger than what we know already from Corollary 5. However, in some sense it has a new quality that is worth mentioning: Corollary 6 holds for $l \geq 14$ and is not an asymptotic result (although we make use of the Ω-notation), whereas Corollary 5 can only be applied for $l \geq 23$ (which is derived from the condition $l/(l+2) > 11/12$), and in addition contains an asymptotic bound on the failure probability.

Corollary 6. *Let $l \geq 14$ and $k - \log k \geq 3l + 5$. Then there exists a set $S \subseteq U = [2^k]$ of size $\leq (7/8) \cdot m + 2$ such that $p_F = \Omega(1)$ if $h_{a_1}, h_{a_2} \in \mathcal{H}_{k,l}^{mult}$ are chosen uniformly at random.*

The following Sections 4.3, 4.4, and 4.5 are devoted to the proof of Theorem 6, where we assume

$$l \geq 14 \text{ and } k - \log k \geq 3l + 5 \tag{4.2.2}$$

throughout, particularly in Lemmas 13, 14, 15, 16, 18, and 22. The constant lower bound we establish for p_{F} is 2^{-24}. Experiments indicate that the failure probability for the sets S constructed here is much larger.

4.3 Basic Structure of the Proof

Apart from the grid structure of the set S, a certain property of hash function pairs is vital in our proof: we say that a pair (h_{a_1}, h_{a_2}) of hash functions from $\mathcal{H}_{k,l}^{\mathrm{mult}}$ has an *almost uniform distribution* of values for the domain $D \in \{U, O_k\}$, if for x chosen uniformly at random from D we have

$$\forall (i,j) \in [2^l]^2: \quad \frac{1}{4} \cdot 2^{-2l} \leq \Pr((h_{a_1}(x), h_{a_2}(x)) = (i,j)) \leq 4 \cdot 2^{-2l}\,. \tag{4.3.1}$$

In Sections 4.4 and 4.5 we prove the following two lemmas, respectively.

Lemma 13. *If (4.2.2) holds, then a fraction of more than $1/7$ of all hash function pairs $(h_{a_1}, h_{a_2}) \in (\mathcal{H}_{k,l}^{mult})^2$ has an almost uniform distribution as in (4.3.1) for $D \in \{U, O_k\}$.*

Lemma 14. *Let $(h_{a_1}, h_{a_2}) \in (\mathcal{H}_{k,l}^{mult})^2$ be a pair with almost uniform distribution for $D \in \{U, O_k\}$, assume (4.2.2), and let $S = S(c, R_c)$ be chosen randomly as in (4.2.1). Then $p_F(S) > 2^{-21}$.*

Once these lemmas are proved, we have proved Theorem 6, because

$$2^{-24} < \frac{1}{7} \cdot 2^{-21} < p_{\mathrm{F}} = p_{\mathrm{F}}(S(c, R_c), h_{a_1}, h_{a_2}) \; .$$

4.4 Proof of Lemma 13: Many Hash Function Pairs Have an Almost Uniform Distribution

The distribution of hash value pairs for $D \in \{U, O_k\}$ is represented by the cuckoo graph $G(D, h_{a_1}, h_{a_2})$. Applying Lemma 6, we can assume w. l. o. g. that $a_1 = 1$, and it remains to identify a suitable set $A_2 \subseteq O_k$ of parameters a_2. We define A_2 as the set

$$\left\{ a \in O_k \mid \exists\, x \in O_{(k-l)-(l+2)} : ax \bmod 2^k \in \left\{ \frac{2^{k-l}}{4}, \ldots, \frac{2^{k-l}}{2} - 1 \right\} \right\} \; . \quad (4.4.1)$$

We will show (Lemma 15) that each pair (h_1, h_{a_2}), $a_2 \in A_2$, has an almost uniform distribution for $D \in \{U, O_k\}$, and that $|A_2|/|O_k| > 1/7$ (Lemma 16), which concludes the proof of Lemma 13.

Lemma 15. *Each pair (h_1, h_{a_2}), $a_2 \in A_2$, has an almost uniform distribution for $D \in \{U, O_k\}$.*

Proof. We define $\min_D$ and $\max_D$ as the *minimum cardinality* and the *maximum cardinality of a preimage with respect to (h_1, h_{a_2}) for the domain D*, respectively, i. e.

$$\min_D := \min\{|\{x \in D \mid (h_1(x), h_{a_2}(x)) = (i, j)\}| : i, j \in [m]\}$$

and $\max_D$ accordingly. If x is chosen uniformly at random from D, then for all $i, j \in [m]$ we have

$$\frac{\min_D}{|D|} \leq \Pr((h_1(x), h_{a_2}(x)) = (i, j)) \leq \frac{\max_D}{|D|} \; . \quad (4.4.2)$$

The cardinality of the preimage of (i, j) with respect to a uniformly distributing hash function pair is $\mathrm{avg}_D := |D|/2^{2l}$, as $m = 2^l$. Assume that

$$\frac{\max_D}{\min_D} \leq 4 \; . \quad (4.4.3)$$

Then we have

$$\frac{\min_D}{|D|} \geq \frac{\max_D}{4|D|} \geq \frac{\mathrm{avg}_D}{4|D|} = \frac{1}{4} \cdot 2^{-2l} \; ,$$

and similarly

$$\frac{\max_D}{|D|} \leq 4 \cdot 2^{-2l} ,$$

and together with (4.4.2) this yields (4.3.1). So, it remains to prove (4.4.3). We consider the matrix $B := (b_{x,y})_{x,y \in U}$ that is given by

$$b_{x,y} := \begin{cases} 1 & \text{if } y = a_2 x \bmod 2^k, \\ 0 & \text{otherwise} . \end{cases}$$

Let $U_i := \{i2^{k-l}, \ldots, (i+1)2^{k-l} - 1\}$ for $i \in [2^l]$. Observe that the number of 1s in the submatrix $B_{i,j} := (b_{x,y})_{x \in U_i, y \in U_j}$ is $|(h_1, h_{a_2})^{-1}(i,j)|$ for $D = U$ (Fig. 4.1), as well as for $D = O_k$ if every second row is counted. This follows from the fact that z div $2^{k-l} = i$ for all $z \in U_i$. Furthermore, observe that the pattern of 1s in all submatrices $B_i := (b_{x,y})_{x \in U_i, y \in U}$ is equal in the sense that $B_{i'}$ is just a shifted version of B_i for arbitrary $i, i' \in [2^l]$. This in turn follows from the fact that each row contains exactly one 1, and from the obvious equivalence

$$b_{x,y} = 1 \Leftrightarrow b_{(x+1) \bmod 2^k, (y+a_2) \bmod 2^k} = 1 ,$$

which holds for all $x, y \in U$.

Thus, for estimating the values of $\max_D$ and $\min_D$, we can restrict ourselves to considering B_0. We have to show that the number of 1s in arbitrary blocks $B_{0,j}$ and $B_{0,j'}$, $j, j' \in [2^l]$, differs by no more than a factor four.

Fix $a_2 \in A_2$ and, according to (4.4.1), $x \in O_{(k-l)-(l+2)}$ with $a_2 x \bmod 2^k \in \{2^{k-l}/4, \ldots, 2^{k-l}/2 - 1\}$ arbitrarily. For each $t \in [x]$, consider the row sequence $(x_s^{(t)})_{0 \leq s \leq d_t}$, where $x_s^{(t)} := (s \cdot x + t)$, and

$$d_t = \left\lfloor \frac{2^{k-l} - (t+1)}{x} \right\rfloor \tag{4.4.4}$$

is the maximum natural number with $(s \cdot x + t) \in U_0$. The 1 in row $x_s^{(t)}$ resides in column $y_s^{(t)} := a_2 x_s^{(t)} \bmod 2^k$, and we refer to the sequence of matrix positions $(x_s^{(t)}, y_s^{(t)})_{0 \leq s \leq d_t}$ which represent the 1s in rows $(x_s^{(t)})$ as $(\text{ones}_s^{(t)})$. Observe that the set of the 1 positions in B_0 is the disjoint union of the sets $\{\text{ones}_s^{(t)} \mid s \in \{0, \ldots, d_t\}\}$ over all $t \in [x]$. Now consider a sequence $(\text{ones}_s^{(t)})$ for a fixed $t \in [x]$. (If for all $j, j' \in [2^l]$ the number of 1s in $B_{0,j}$ and $B_{0,j'}$ given by $(\text{ones}_s^{(t)})$ differs by no more than a factor four, then the same is true if we sum over all $t \in [x]$.)

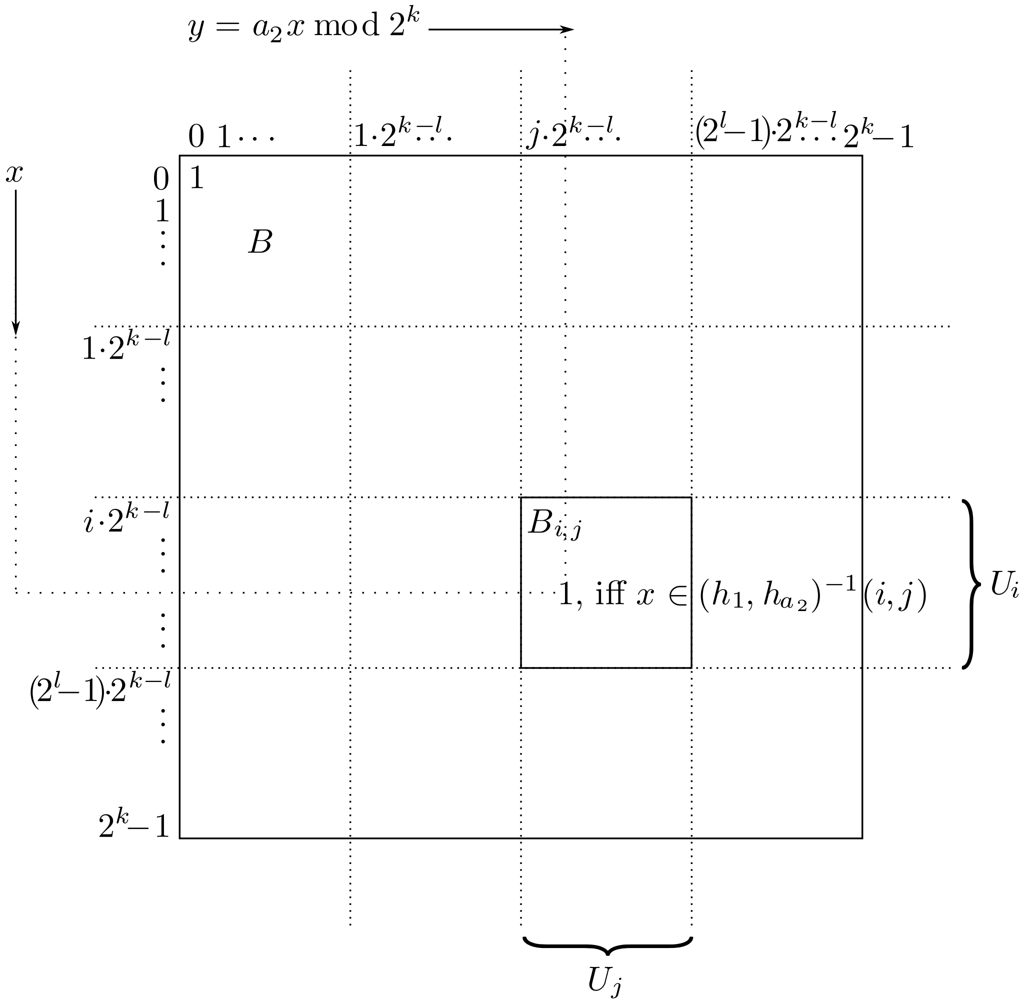


Figure 4.1: The number of 1s in $B_{i,j}$ is $|(h_1, h_{a_2})^{-1}(i,j)|$ for $D = U$, because the single 1 in row x is in row block $i = x \text{ div } 2^{k-l} = h_1(x)$ and in column block $j = h_{a_2}(x)$.

Whenever the sequence $(\text{ones}_s^{(t)})$ *passes* a block $B_{0,j}$, it *hits* this block with at least two and at most four successive elements, because by the definition of A_2 the step size y of the column sequence $(y_s^{(t)})$ is

$$y = a_2 x \bmod 2^k \in \left\{ \frac{2^{k-l}}{4}, \dots, \frac{2^{k-l}}{2} - 1 \right\} . \tag{4.4.5}$$

Furthermore, the sequence $(\text{ones}_s^{(t)})$ passes each block $B_{0,j}$, $j \in [2^l]$, at least once, because the sum of d_t steps of size y is greater than $2^k - 2^{k-l}/2$, by (4.4.4) and (4.4.5).

We obtain an upper bound on $\max_D / \min_D$ as follows. Consider an arbitrary block $B_{0,j}$. This block is passed by the sequence $(\text{ones}_s^{(t)})$ at least once and whenever it is passed it is hit with at least two and at most four elements. Now assume that there exist blocks $B_{0,j}$ and $B_{0,j'}$, $j, j' \in [2^l]$, such that $(\text{ones}_s^{(t)})$ passes $B_{0,j'}$ once and hits it with only two elements, whereas $B_{0,j}$ is passed twice and each time hit with four elements. In this worst case, the number of 1s, given by $(\text{ones}_s^{(t)})$, differs in the two blocks by a factor four, and hence (4.4.3) is proved for $D = U$ (see Fig. 4.2).

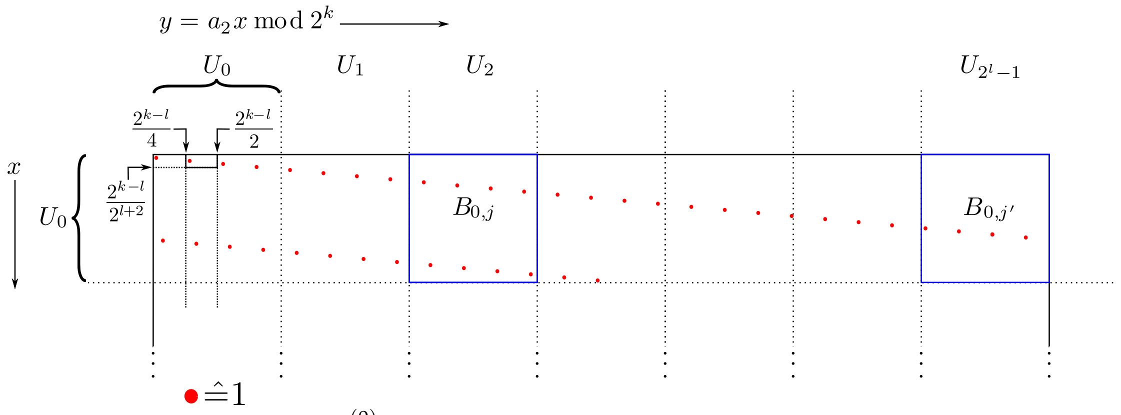


Figure 4.2: $(\text{ones}_s^{(0)})$ in the worst case with respect to $\max_D / \min_D$

For $D = O_k$, the argument is similar, noticing that every second element of $(x_s^{(t)})$ is odd: if t is odd, then $x_s^{(t)}$ is odd for every even s, and vice versa. Thus, the corresponding sequence of matrix positions which contain the 1s passes every block $B_{0,j}$ at least once, and hits it with one or two elements whenever it is passed, and so on. □

Lemma 16. $|A_2| / |O_k| > 1/7$.

Proof. Consider A_2 as in (4.4.1). For all $x \in O_{(k-l)-(l+2)}$ and $i \in [x]$ we define

$$Q_{x,i} := \left[\frac{1}{x}\left(i \cdot 2^k + \frac{2^{k-l}}{4}\right), \frac{1}{x}\left(i \cdot 2^k + \frac{2^{k-l}}{2}\right)\right), \tag{4.4.6}$$

$$Q_x := \bigcup_{i \in [x]} Q_{x,i}, \text{ and} \tag{4.4.7}$$

$$Q := \bigcup_{x \in O_{(k-l)-(l+2)}} Q_x. \tag{4.4.8}$$

Then

$$A_2 = Q \cap O_k,$$

where for our purposes the obvious subset relation $Q \cap O_k \subseteq A_2$ is sufficient. Observe that there exist disjoint half-open intervals $I_1, \ldots, I_t$ of the form $[a', b')$ and of length $\geq 2^l$ such that $Q = \bigcup_{1 \leq j \leq t} I_j$. This in particular implies that $t < |Q|/2^l$. As $|Q| = |I_1| + \cdots + |I_t|$ and each interval I_j contains at least $\lfloor |I_j| \rfloor$ natural numbers, of which at least $\lfloor |I_j|/2 \rfloor$ are odd, we have

$$|A_2| \geq |Q \cap O_k| > \sum_{1 \leq j \leq t} \left(\frac{|I_j|}{2} - 1\right) = \frac{|Q|}{2} - t > (1 - 2^{-l+1}) \cdot \frac{|Q|}{2},$$

and hence

$$\frac{|A_2|}{|O_k|} > (1 - 2^{-l+1}) \cdot \frac{|Q|}{2^k}.$$

We show that $|Q|/2^k > 5/2^5$. Then $(1 - 2^{-l+1}) \cdot |Q|/2^k > 1/7$ for $l \geq 14$—as desired.

A simple inclusion-exclusion bound, Boole's inequalities, turns out to be helpful to establish a lower bound for $|Q|/2^k$.

Lemma 17 (Boole's inequalities). *Let $D_1, \ldots, D_r$, $r \in \mathbb{N}$, be arbitrary events. Then*

$$\sum_{i=1}^{r} \Pr(D_i) - \sum_{1 \leq i < j \leq r} \Pr(D_i \cap D_j) \leq \Pr\left(\bigcup_{i=1}^{r} D_i\right) \leq \sum_{i=1}^{r} \Pr(D_i).$$

In order to apply it, we normalize all values by shrinking the interval $[0, 2^k)$ to $[0, 1)$ and working in the measure space that is given by $[0, 1)$ with the usual Lebesgue measure λ.

Now, in direct correspondence to (4.4.6), (4.4.7), and (4.4.8), we define the sets

$$E_{x,i} := \left[\frac{1}{x}(i+2^{-(l+2)}), \frac{1}{x}(i+2^{-(l+1)})\right) , \tag{4.4.9}$$

$$E_x := \bigcup_{i\in[x]} E_{x,i} \text{ , and} \tag{4.4.10}$$

$$E := \bigcup_{x\in O_{(k-l)-(l+2)}} E_x \; . \tag{4.4.11}$$

Then

$$\frac{|Q|}{2^k} = \lambda(E) \; .$$

The following lemma allows us to apply Boole's lower bound inequality.

Lemma 18. *Let $x, x' \in [2^{(k-l)-(l+2)}] - [2^{(k-l)-(l+3)}]$, $x < x'$, be coprime. Then*

$$\lambda(E_x \cap E_{x'}) < \frac{3}{16} \cdot 2^{-2l} \; .$$

Proof. Consider E_x (see (4.4.10)). The lower interval limits $l_{x,i}$ of the intervals $E_{x,i}$, $i \in [x]$, cyclically divide $[0, 1)$ into x sections of the same size $1/x$, each of which starts with an *x-interval* $E_{x,i}$ (Fig. 4.3).

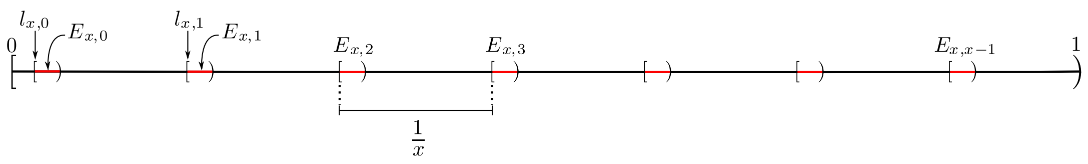


Figure 4.3: event $E_x = \bigcup_{i\in[x]} E_{x,i}$

In addition consider $E_{x'}$, which forms a similar pattern, but with a period of $1/x'$. We show that, for the allowed x and x', $\lambda(E_x \cap E_{x'})$ is $O(\lambda(E_x) \cdot \lambda(E_{x'}))$, and hence approximately the same as in the case of independence between E_x and $E_{x'}$. For this, we overlay the sections $[l_{x,0}, l_{x,1}), [l_{x,1}, l_{x,2}), \ldots, [l_{x,x-1}, 1) \cup [0, l_{x,0})$. Then we get a single half-open interval I_x of length $1/x$, w. l. o. g. $I_x = [0, 1/x)$, such that the lower interval limits of the intervals $E_{x',i}$, $i \in [x']$, form a sequence $(l_i)_{i\in[x']}$ which begins at some position $l_0 \in I_x$, and proceeds cyclically modulo $1/x$ with steps of size $1/x'$. That is, $l_i = (l_0 + i \cdot 1/x') \bmod 1/x$ for $i \in [x']$ (Fig. 4.4).

For coprime numbers x and x', $x < x'$, it is easy to see that the sets $\{x, 2x \bmod x', \ldots, (x'-1)x \bmod x'\}$ and $\{1, \ldots, x'-1\}$ are equal. Adding zero, and nor-

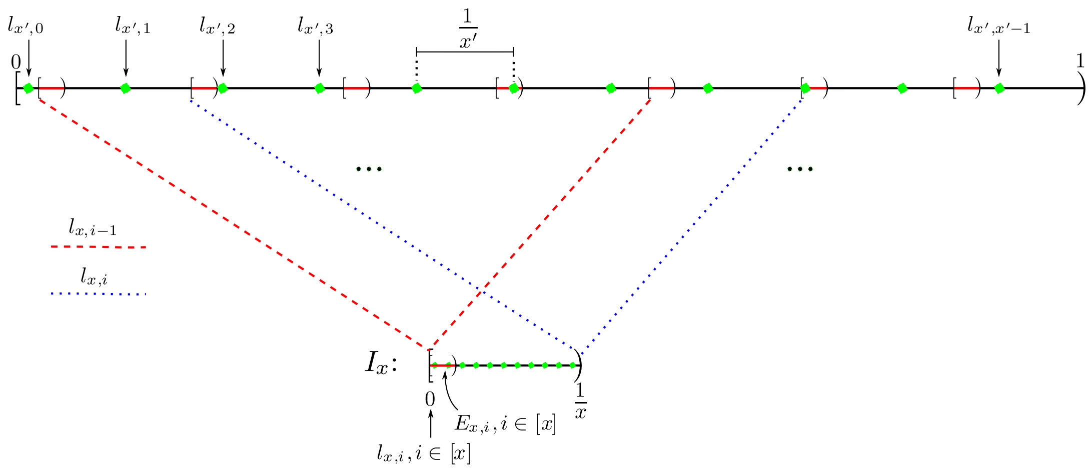


Figure 4.4: lower interval limits $l_{x',i}$ of event $E_{x'}$ and *overlay* I_x

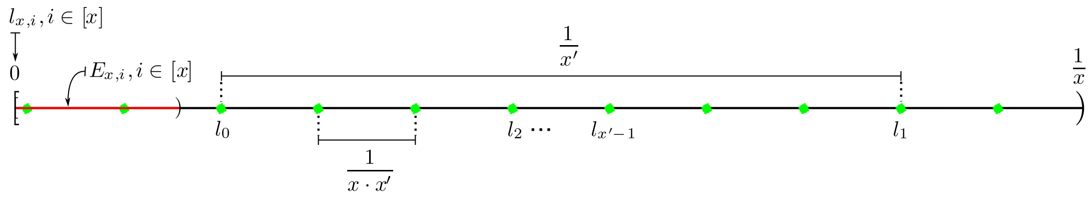


Figure 4.5: lower interval limits of $E_{x'}$ form a cyclic grid $\{l_i \mid i \in [x']\}$ in I_x

malizing all numbers by multiplication with $(x \cdot x')^{-1}$ yields

$$\left\{ \frac{i}{x'} \bmod \frac{1}{x} \mid i \in [x'] \right\} = \left\{ \frac{i}{x \cdot x'} \mid i \in [x'] \right\} ,$$

and hence

$$\left\{ \left(l_0 + \frac{i}{x'} \right) \bmod \frac{1}{x} \mid i \in [x'] \right\} = \left\{ \left(l_0 + \frac{i}{x \cdot x'} \right) \bmod \frac{1}{x} \mid i \in [x'] \right\} .$$

That means $\{l_i \mid i \in [x']\}$ forms a *cyclic grid* in I_x, where neighboring elements have a difference of exactly $1/(x \cdot x')$ (Fig. 4.5).

Obviously, an interval $E_{x',i}$ can only have a nonempty intersection with E_x, if its left interval limit $l_{x',i}$ is either within or shortly before an x-interval, or equivalently if the number $l_i \in I_x$ which corresponds to $l_{x',i}$ lies in

$$\Phi := \left[0, \frac{1}{x \cdot 2^{l+2}} \right) \cup \left[\frac{1}{x} - \frac{1}{x' \cdot 2^{l+2}}, \frac{1}{x} \right) \subseteq I_x .$$

Let $H_\Phi := \{ i \in [x'] \mid l_i \in \Phi \}$. Our knowledge about the regular shape of $\{l_i \mid i \in [x']\}$ within I_x allows us to easily bound the size of H_Φ:

$$\begin{aligned} |H_\Phi| &\leq \left\lceil \frac{\lambda(\Phi)}{(x \cdot x')^{-1}} \right\rceil && (\text{shape of } \{l_i \mid i \in [x']\}) \\ &\leq (x \cdot x') \left((x \cdot 2^{l+2})^{-1} + (x' \cdot 2^{l+2})^{-1} \right) + 1 && (\text{def. } \lambda, \Phi) \\ &< (2x' \cdot 2^{-(l+2)}) + 1 && (\text{as } x < x') \end{aligned}$$

This finally yields the claimed bound on $\lambda(E_x \cap E_{x'})$:

$$\begin{aligned}
\lambda(E_x \cap E_{x'}) &= \lambda\left(\bigcup_{i\in[x']} (E_x \cap E_{x',i})\right) \\
&= \sum_{i\in[x']} \lambda(E_x \cap E_{x',i}) && \text{(pairwise disjoint events)} \\
&\le \sum_{l_i\in H_\Phi} \lambda(E_{x',i}) && \text{(def. } H_\Phi\text{)} \\
&< \left(\frac{2x'}{2^{l+2}}+1\right)\cdot\frac{1}{x'\cdot 2^{l+2}} && \text{(see above)} \\
&= \frac{1}{2^{2l+3}} + \frac{1}{x'\cdot 2^{l+2}} \\
&< \frac{1}{2^{2l+3}} + \frac{1}{2^{k-l-1}} && \text{(as } x' > 2^{(k-l)-(l+3)}\text{)} \\
&\le \frac{3}{16}\cdot 2^{-2l} && \text{(for } k \ge 3l+5\text{)}
\end{aligned}$$

□

The intuitive meaning of Lemma 18 is as follows: for the admitted x and x', the probability $\lambda(E_x \cap E_{x'})$ of $E_x \cap E_{x'}$ is not much larger than $\lambda(E_x)\cdot\lambda(E_{x'}) = (2^{-(l+2)})^2 = (1/16)\cdot 2^{-2l}$, and hence approximately the same as in the case of independence between E_x and $E_{x'}$.

Consider the following fact that was proved by Finsler [22].

Lemma 19 (Finsler's inequalities [22])**.** *Let $n \in \mathbb{N}$, $n > 1$, and define $\pi(n)$ as the number of distinct prime numbers less than or equal to n. Then*

$$\frac{n}{3\ln(2n)} < \pi(2n) - \pi(n) < \frac{7n}{5\ln(n)}\ .$$

By Finsler's inequalities we know that the set

$$[2^{(k-l)-(l+2)}] - [2^{(k-l)-(l+3)}]$$

contains at least $2^{k-2l-3}/(3\ln(2^{k-2l-2}))$ distinct prime numbers. Of course these prime numbers are odd and pairwise coprime. For $k - \log k \ge 3l+5$ we have $2^{k-2l-3}/(3\ln(2^{k-2l-2})) \ge 2^l$. So, let PR be a set of exactly 2^l distinct prime numbers in $[2^{(k-l)-(l+2)}] - [2^{(k-l)-(l+3)}]$. We complete the proof of Lemma 16 as follows:

$$\begin{aligned}
\frac{|Q|}{2^k} = \lambda(E) = \lambda\left(\bigcup_{x\in O_{(k-l)-(l+2)}} E_x\right) && \text{(see (4.4.11))}\\
\geq \lambda\left(\bigcup_{x\in \mathrm{PR}} E_x\right) && (\mathrm{PR} \subseteq O_{(k-l)-(l+2)})\\
> \sum_{x\in\mathrm{PR}} \lambda(E_x) - \sum_{\substack{x,x'\in\mathrm{PR},\\ x\neq x'}} \lambda(E_x \cap E_{x'}) && \text{(Lemma 17)}\\
> 2^l \cdot 2^{-(l+2)} - \binom{2^l}{2} \cdot \frac{3}{16} \cdot 2^{-2l} && \text{(Lemma 18)}\\
> \frac{5}{2^5}\,.
\end{aligned}$$

□

4.5 Proof of Lemma 14: $p_\mathrm{F}(S)$ under the Condition of an Almost Uniform Distribution

For the proof of Lemma 14 we consider the cuckoo graph $G = (V_1, V_2, E) = G(S, h_{a_1}, h_{a_2})$ of the set $S = S(c, R_c)$. We show that if c is suitably chosen then a large subset of the edges corresponding to G_0 and G_c in G form a set of simple paths with disjoint vertex sets. The number of these paths is a random variable Δ. Then we prove a lower bound for the probability that we chose a suitable c and that under the condition of a suitable c choosing R_c yields $\geq \Delta + 1$ edges with endpoints on the Δ paths, and hence yields a bad edge set. This will conclude the proof of Lemma 14.

In the following we refer to the edge that corresponds to a key $x_i(c') \in G_{c'}$ as $e_i(c')$ for arbitrary $c' \in U$, and we say that the keys $x \neq y$ *collide under* the hash function h if $h(x) = h(y)$.

Lemma 20. *Each hash function $h_a \in \mathcal{H}^{mult}_{k,l}$ maps $G_{c'}$ one-to-one into $[m]$ for arbitrary $c' \in U$.*

Proof. Let $x, y \in G_{c'}$, $x \neq y$, be arbitrary. We have to show that $h_a(x) \neq h_a(y)$. Let i and j be the unique numbers in $[d]$ with $x = (c' + i \cdot 2^{k-l}) \bmod 2^k$ and $y = (c' + j \cdot 2^{k-l}) \bmod 2^k$, and assume w. l. o. g. that $i < j$. Then we have

$y = (x + t \cdot 2^{k-l}) \bmod 2^k$ for the positive integer $t := j - i < 2^l/3$. Now, on the one hand we have

$$h_a(x) = (ax \bmod 2^k) \operatorname{div} 2^{k-l} ,$$

and on the other hand we derive

$$h_a(y) = ((ax \bmod 2^k + at \bmod 2^k \cdot 2^{k-l}) \bmod 2^k) \operatorname{div} 2^{k-l} .$$

As $0 < t < 2^l$ and a is odd, $t' := at \bmod 2^k$ is neither zero nor a multiple of 2^l. This implies that $t' \cdot 2^{k-l} \bmod 2^k \geq 2^{k-l}$, and hence $h_a(x) \neq h_a(y)$. $\square$

Lemma 20 applied for $c' = 0$ and $c' = c$, respectively, yields that the edges corresponding to G_0 and the edges corresponding to G_c each form a matching of size d. Imagine each of them as a set of d parallel lines in increasing order w. r. t. the indices $i \in [d]$ of the edges $e_i(0)$ and $e_i(c)$, respectively (Fig. 4.6).

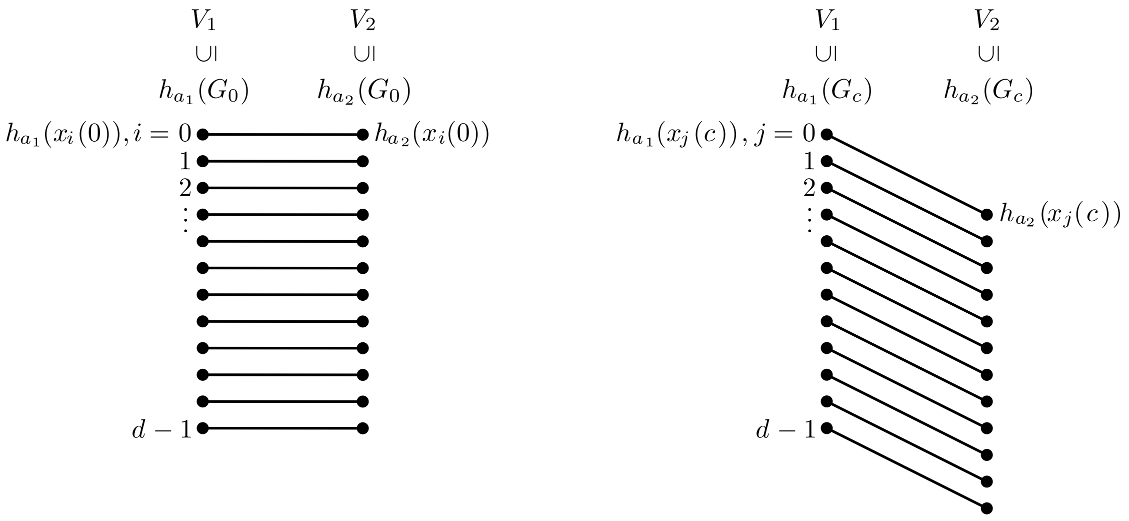


Figure 4.6: The two matchings corresponding to G_0 and G_c

Consider $e_0(c) = (h_{a_1}(x_0(c)), h_{a_2}(x_0(c)))$ and assume that $x_{i_1}(0)$ and $x_0(c)$ collide under h_{a_1}, and that $x_{i_2}(0)$ and $x_0(c)$ collide under h_{a_2}, respectively, for $i_1 \neq i_2$, w. l. o. g. $i_1 < i_2$. The following Lemma, applied for $h_a = h_{a_1}$, $\alpha = 0$, $\beta = c$, $i = i_1$, and $i' = 0$ as well as for $h_a = h_{a_2}$, $\alpha = 0$, $\beta = c$, $i = i_2$, and $i' = 0$ says that under this assumption there is a sequence of collisions between keys in G_0 and G_c both with respect to h_{a_1} and h_{a_2}.

Lemma 21. *Let $h_a \in \mathcal{H}^{mult}_{k,l}$, as well as offsets $\alpha, \beta \in U$, and indices $i, i' \in [2^l]$ be arbitrary. If $x_i(\alpha)$ and $x_{i'}(\beta)$ collide under h_a then $x_j(\alpha)$ and $x_{j'}(\beta)$ collide under h_a for all $j, j' \in [2^l]$ with $j - j' = i - i'$.*

Proof. Assume that $x_i(\alpha)$ and $x_{i'}(\beta)$ collide under h_a, and let $j, j' \in [2^l]$ with $j - j' = i - i'$ be arbitrary. Then there exists a unique integer z with $j = i + z$ and $j' = i' + z$. An elementary calculation proves that $x_j(\alpha)$ and $x_{j'}(\beta)$ collide under h_a:

$$\begin{aligned}
h_a(x_j(\alpha)) &= (a(\alpha + j2^{k-l})) \bmod 2^k \operatorname{div} 2^{k-l} && \text{(by def.)}\\
&= (a(\alpha + (i+z)2^{k-l})) \bmod 2^k \operatorname{div} 2^{k-l} && (j = i + z)\\
&= (ax_i(\alpha) + az2^{k-l}) \bmod 2^k \operatorname{div} 2^{k-l}\\
&= (ax_{i'}(\beta) + az2^{k-l}) \bmod 2^k \operatorname{div} 2^{k-l}\\
&\ \vdots\\
&= h_a(x_{j'}(\beta))
\end{aligned}$$

The second to last equation is proved as follows: as we assumed that $x_i(\alpha)$ and $x_{i'}(\beta)$ collide under h_a, the bits at positions $k-1$ down to $k-l$ of $ax_i(\alpha)$ and $ax_{i'}(\beta)$ must be equal. Furthermore, as the $k-l$ lowest order bits of $az2^{k-l}$ are zero, the carry from position $k-l-1$, when we perform a binary addition of $az2^{k-l}$ with $ax_i(\alpha)$ and $ax_{i'}(\beta)$, respectively, is zero, too. Summing up, $(ax_i(\alpha) + az2^{k-l})$ and $(ax_{i'}(\beta) + az2^{k-l})$ have the same bits at positions $k-1$ down to $k-l$. □

So we have

$$h_{a_t}(x_j(c)) = h_{a_t}(x_{i_t+j}(0)) \text{ for } 0 \le j \le d - i_t - 1\,, t \in \{1,2\}\,,$$

and hence the two matchings given by the edges of G_0 and G_c can be merged as depicted in Fig. 4.7. This reveals the existence of $\Delta := i_2 - i_1$ simple paths $P_0, \ldots, P_{\Delta-1}$ in G with disjoint vertex sets. Furthermore, the total number $|V_1'|$ and $|V_2'|$ of vertices in V_1 and in V_2 covered by these paths is at least $d - i_1$, respectively.

Let $d' := \lceil d/2^7 \rceil + 1$, and

$$Z := \{(h_{a_1}(x_{i_1}(0)), h_{a_2}(x_{i_2}(0))) \mid i_1, i_2 \in [d'], i_1 \neq i_2\}\,. \tag{4.5.1}$$

For $e_0(c) \in Z$ we have just proven the existence of $\Delta \le \lceil d/2^7 \rceil$ simple paths $P_0, \ldots, P_{\Delta-1}$ in G with disjoint vertex sets, which in total cover the vertex set $V_1' \subseteq V_1$ and $V_2' \subseteq V_2$ of size $\ge d - \lceil d/2^7 \rceil$, respectively. We assume w. l. o. g.

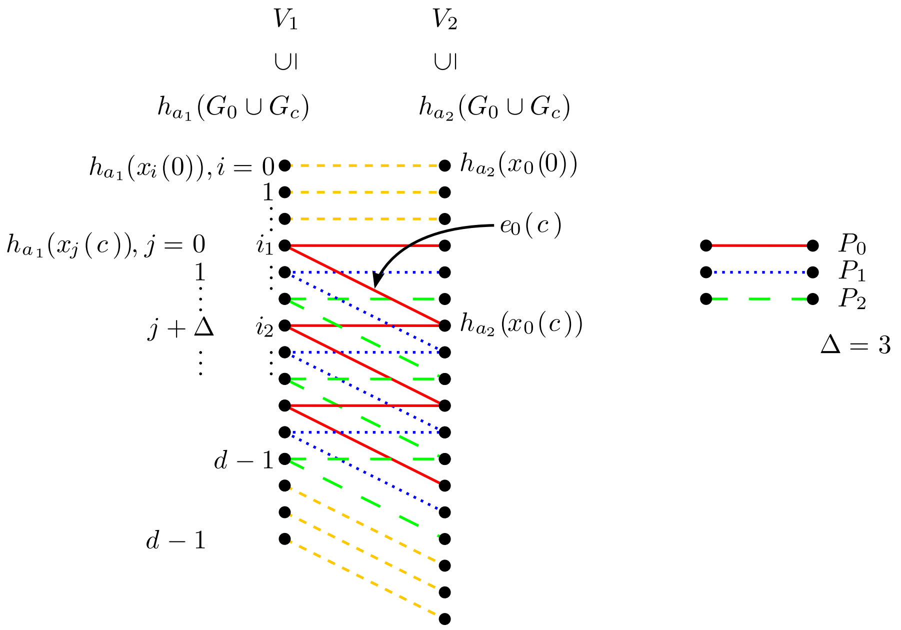


Figure 4.7: The Δ paths in G for $e_0(c) = (h_{a_1}(x_{i_1}(0)), h_{a_2}(x_{i_2}(0)))$

that S is the result of a random experiment where first $c \in O_k$, and then $R_c \subseteq U - (G_0 \cup G_c)$ is chosen uniformly at random. So,

$$\begin{aligned}\Pr(e_0(c) \in Z) &= \sum_{(i,j)\in Z} \Pr((h_{a_1}, h_{a_2})(c) = (i,j)) \\ &\geq |Z| \cdot \frac{1}{4} \cdot 2^{-2l} && \text{(Lemma 13)} \\ &\geq \left(\frac{d}{2^7}\right)^2 \cdot 2^{-2l-2}\,, && \text{((4.5.1), Lemma 20)}\end{aligned}$$

and thus, as $d = \lceil (1-\delta)2^l/3 \rceil$, we have

$$\Pr(e_0(c) \in Z) \geq \frac{(1-\delta)^2}{9 \cdot 2^{16}}\,. \tag{4.5.2}$$

If $e_0(c) \in Z$ and if the uniform random choice of $R_c \subseteq U - (G_0 \cup G_c)$ yields $\geq \Delta + 1$ edges in $V_1' \times V_2'$, then these edges together with the edges of $P_0, \ldots, P_{\Delta-1}$ obviously form a bad edge set. We refer to the event that choosing R_c yields $\geq \Delta + 1$ edges in $V_1' \times V_2'$ as F.

Lemma 22. $\Pr(F \mid e_0(c) \in Z) \geq 1 - \exp\left(-\left(\frac{(1-\delta)}{3}\right)^3 2^{l-9}\right)$.

Proof. The proof is a straightforward application of a Chernoff bound. We assume w. l. o. g. that the keys in R_c are chosen one by one from $U - (G_0 \cup G_c)$ without repetition. Define the 0-1-random variables X_i, $1 \leq i \leq d$, as 1, if the ith chosen key yields an edge in $V_1' \times V_2'$, 0 otherwise. Then the random variable $X := \sum_{i=1}^{d} X_i$ counts the number of keys in R_c which yield edges in $V_1' \times V_2'$, and

$$\Pr(F \mid e_0(c) \in Z) = \Pr(X > \Delta) .$$

For the number N' of keys in U that are mapped on $V_1' \times V_2'$ with respect to an almost uniformly distributing hash function pair (h_{a_1}, h_{a_2}) we have

$$N' \geq |V_1' \times V_2'| \cdot \frac{1}{4} \cdot 2^{-2l} \cdot |U| ,$$

where the condition $e_0(c) \in Z$ yields

$$\begin{aligned} |V_1' \times V_2'| &\geq (d - i_1)^2 \\ &\geq \left(d - \left\lceil \frac{d}{2^7} \right\rceil\right)^2 \\ &\geq \left(d - \frac{d}{2^6}\right)^2 && (l \geq 14, \delta = 1/8) \\ &> \left(\frac{5}{16}(1-\delta)\right)^2 2^{2l} , \end{aligned}$$

and hence,

$$N' \geq \left(\frac{5}{16}(1-\delta)\right)^2 2^{k-2} .$$

This implies under the given constraints $l \geq 14$ and $k - \log k \geq 3l + 5$ that

$$\Pr(X_i = 1) \geq \frac{N' - 3d}{|U|} > \frac{(1-\delta)^2}{2^6} .$$

We underestimate $\Pr(X > \Delta)$ by assuming that the X_i are independent and identically distributed with $\Pr(X_i = 1) = q := (1-\delta)^2/2^6$. Then $\mathrm{E}[X] = dq$, and we may apply the following Chernoff bound [24] for $t = d$ and $(Y_0, \ldots, Y_{d-1}) = (X_1, \ldots, X_d)$.

Lemma 23 (Chernoff bound). *Let $Y_0, \ldots, Y_{t-1}$ be independent 0-1-random variables such that $\Pr(Y_i = 1) = q_i$. Let $Y := \sum_{i=0}^{t-1} Y_i$ and $\mu := \mathrm{E}[Y]$. Then, for $0 < \varepsilon < 1$, we have*

$$\Pr(Y \leq (1-\varepsilon)\mu) \leq e^{-\frac{\mu\varepsilon^2}{2}} .$$

This yields

$$
\begin{aligned}
\Pr(F \mid e_0(c) \in Z) &\geq \Pr(X > \Delta) \\
&= 1 - \Pr(X \leq \Delta) \\
&\geq 1 - \Pr\left(X \leq \left\lceil \frac{d}{2^7} \right\rceil\right) && \left(\Delta < d' = \left\lceil \frac{d}{2^7} \right\rceil\right) \\
&\geq 1 - \Pr\left(X \leq \left(1 - \frac{1}{6}\right) \mathrm{E}[X]\right) && \left(l \geq 14, \delta = \frac{1}{8}\right) \\
&\geq 1 - \exp\left(-\frac{\mathrm{E}[X] \left(\frac{1}{6}\right)^2}{2}\right) && (\text{Lemma } 23) \\
&= 1 - \exp\left(-\frac{dq}{72}\right) \\
&\geq 1 - \exp\left(-\left(\frac{(1-\delta)}{3}\right)^3 2^{l-9}\right) ,
\end{aligned}
$$

which completes the proof of Lemma 22. □

Applying Lemma 22, we complete the proof of Lemma 14 as follows.

$$
\begin{aligned}
p_{\mathrm{F}}(S) &\geq \Pr(e_0(c) \in Z) \cdot \Pr(F \mid e_0(c) \in Z) \\
&\geq \frac{(1-\delta)^2}{9 \cdot 2^{16}} \cdot \left(1 - \exp\left(-\left(\frac{(1-\delta)}{3}\right)^3 2^{l-9}\right)\right) && ((4.5.2), \text{Lemma } 22) \\
&\geq \left(\frac{7}{3}\right)^2 2^{-22} \left(1 - \exp\left(-\left(\frac{7}{3}\right)^3 2^{l-18}\right)\right) && (\delta = 1/8) \\
&> 2^{-21} . && (l \geq 14)
\end{aligned}
$$

4.6 Experiments

Using the same implementation of cuckoo hashing as in the case of dense key sets (see Chapter 3, Section 3.5) we carried out experiments that were meant to obtain estimates of the failure probability p_{F} by counting the average failure frequency among 100 independently and uniformly random chosen grid based sets $S(c, R_c)$, as considered in the proof of Theorem 6, of size $(1-\delta)m$, each set inserted 1000 times with independently and uniformly random chosen hash functions. This was repeated several times for fixed $k = 126$ and $\delta = 0.1$, and

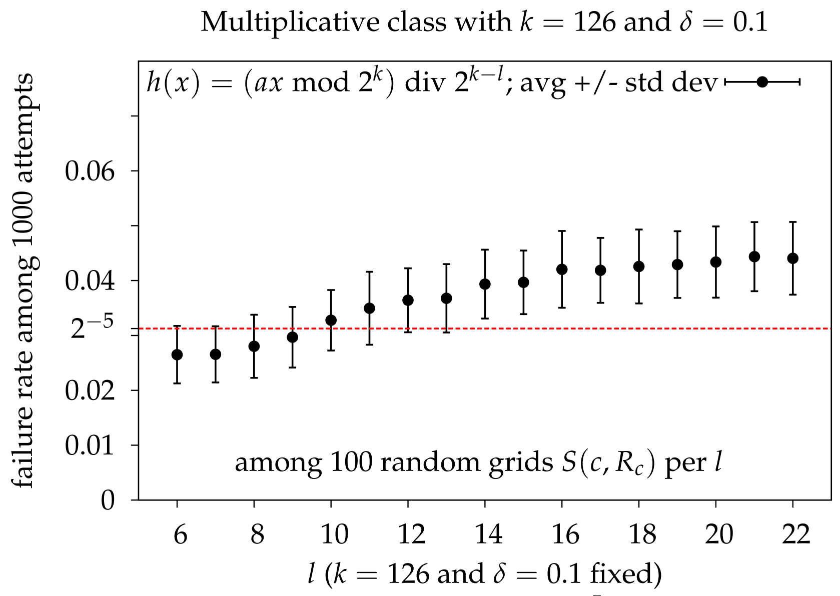


Figure 4.8: Constant failure probability of about 2^{-5} for the multiplicative class

for each $l \in \{6, \ldots, 22\}$. The result is depicted in Fig. 4.8, where a dot represents the average failure rate, and an error bar is given by addition and subtraction of the "normalized" sample standard deviation from the average. (There is a bit confusion in the literature about the notion "sample standard deviation." Some just call it "standard deviation.") The sample standard deviation s with respect to failure frequencies for a given value of l is computed according to the following well known formula:

$$s = \sqrt{\frac{1}{t-1}\sum_{i=1}^{t}(x_i - \bar{x})^2}\,,$$

where in our case $t = 100$ denotes the number of randomly chosen sets $S(c, R_c)$ for a given table size 2^l, x_i is the number of failures among 1000 independent attempts to insert the i-th chosen set, and $\bar{x}$ is the average failure frequency among the t sets.

It can be seen from Fig. 4.8 that for a randomly chosen grid based set S and hash functions h_1, h_2 chosen uniformly at random from the multiplicative class p_{F} appears to be much larger than the constant from Theorem 6: the bound of Theorem 6 is 2^{-24}, whereas we see a failure rate of about 2^{-5}.

CHAPTER 5

Conclusion and Open Problems

In the case $m/N \geq N^{\gamma-1}$ for a suitable constant $\gamma \in (0,1)$, i.e. for relatively dense key sets, we have answered the question of why cuckoo hashing does not work well with the multiplicative class. We have proved that even for a moderate load factor of $1/4$ the failure probability for highly random key sets is $1 - o(1)$ if arbitrary multiplicative functions are used. This implies, for a load factor of no more than $1/4$, the existence of a "bad set" S such that the failure probability is $1 - o(1)$ if S is inserted with multiplicative functions that are chosen uniformly at random. We further showed that in the same sense cuckoo hashing performs badly when combined with the common linear class, even if h_1 and h_2 are chosen from distinct linear classes. Cuckoo hashing should not be used with these classes if U is not sufficiently large compared to m.

In the remaining case of sparse key sets we have shown that cuckoo hashing combined with the multiplicative class has a failure probability of $\Omega(1)$, or even $1 - o(1)$, if relatively small random key sets that exhibit a certain structure are inserted. Therefore, summing up the results for the multiplicative class, we generally recommend to avoid using cuckoo hashing with the multiplicative class.

Our results point out that care must be taken when interpreting the result by Mitzenmacher and Vadhan [32]—one has to check carefully that the hypotheses of the theorems are satisfied: Not only "sufficient randomness of the data" is required, but also has the universe U to be large compared to the table size m.

Recently Mitzenmacher, Kirsch, and Wieder [27] proposed a simple method to reduce the failure probability. Under the full randomness assumption they prove: If during the sequential insertion of n items, given $m = (1+\varepsilon)n$, each item that would cause the insertion to fail is put into a *stash*, i. e. into an extra storage apart from T_1 and T_2, then the size σ of the stash after all items have been inserted satisfies

$$\Pr(\sigma \geq s) = O(n^{-s})$$

for all $s \geq 1$. So, using a small stash of constant size leads to a drastic decrease of the failure probability. It is an intriguing open question how much the failure probability drops in the situation that we investigated in this thesis, i. e. if h_1 and h_2 are chosen from a simple universal class in a case where the result of [32] cannot be applied. Maybe this simple trick is a way out of the dilemma of a high failure probability. However, we suppose that a small stash of constant size will not help in the situation where N is a (small) constant times m, because then it seems to be inevitable that one of the $\Omega(m)$ bad edge sets is "switched on" during the random choice of $m/2$ edges (compare the proofs of our Theorems 3, 4, and 5).

Another open question of great interest is the behavior of d-ary and blocked cuckoo hashing for $d > 2$ ([23] and [17, 42], respectively, see Section 1.1.5.2) in combination with simple universal hash classes. We suppose that they will do better than standard cuckoo hashing, as a growing number of choices for each key should reduce the failure probability.

We have implicitly proved that pairwise independence of the hash functions does not suffice to guarantee a good behavior of cuckoo hashing. All is not lost however. In our experiments cuckoo hashing seemed to work very well with the $(2,3)$-universal class $\mathcal{H}^3_{p,m}$ of quadratic polynomials over a prime field. You may agree upon these polynomials being fairly simple and efficient. A mathematical proof of this good behavior remains an open problem. This seems to be a difficult problem.

Finally, one may argue that our results appear to be negative. However, as every cloud has a silver lining, we strongly believe that our work will enhance the practical use of hashing in the long term. A deeper understanding of a high performance hashing scheme like cuckoo hashing combined with simple and efficient universal hash functions surely is a step towards this goal.

References

[1] Tuğkan Batu, Petra Berenbrink, and Colin Cooper. Balanced allocations: Balls-into-bins revisited and chains-into-bins. In *CDAM Research Report LSE-CDAM-2007-34*. London School of Economics, 2007.

[2] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. *SIAM J. Comput.*, 17(2):210–229, 1988.

[3] Fabiano C. Botelho, Rasmus Pagh, and Nivio Ziviani. Simple and space-efficient minimal perfect hash functions. In *Workshop on Algorithms and Data Structures (WADS '07)*, pages 139–150. Springer LNCS, 2007.

[4] Julie A. Cain, Peter Sanders, and Nicholas C. Wormald. The random graph threshold for *k*-orientiability and a fast algorithm for optimal multiple-choice allocation. In *Proc. 18th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA '07)*, pages 469–476, 2007.

[5] Neil J. Calkin. Dependent sets of constant weight binary vectors. *Comb. Probab. Comput.*, 6(3):263–271, 1997.

[6] Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

[7] Bernard Chazelle, Joe Kilian, Ronitt Rubinfeld, and Ayellet Tal. The bloomier filter: an efficient data structure for static support lookup tables. In *Proc. 15th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA '04)*, pages 30–39. SIAM, 2004.

[8] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988.

[9] Kai-Min Chung and Salil P. Vadhan. Tight bounds for hashing block sources. In *APPROX-RANDOM*, pages 357–370, 2008.

[10] Zbigniew J. Czech, George Havas, and Bohdan S. Majewski. Perfect hashing. *Theoretical Computer Science*, 182(1–2):1–143, 1997.

[11] Luc Devroye and Pat Morin. Cuckoo hashing: Further analysis. *Information Processing Letters*, 86:215–219, 2003.

[12] Martin Dietzfelbinger. Design strategies for minimal perfect hash functions. In *Proc. 4th Int. Symp. on Stochastic Algorithms: Foundations and Applications (SAGA '07)*, volume 4665, 2007.

[13] Martin Dietzfelbinger and Rasmus Pagh. Succinct data structures for retrieval and approximate membership (extended abstract). In *ICALP (1)*, pages 385–396, 2008.

[14] Martin Dietzfelbinger and Michael Rink. Applications of a splitting trick. In *Proc. 36th Int. Colloquium on Automata, Languages and Programming (ICALP '09)*, pages 354–365. Springer-Verlag, 2009.

[15] Martin Dietzfelbinger and Ulf Schellbach. On risks of using cuckoo hashing with simple universal hash classes. In *Proc. 19th Annual ACM -SIAM Symp. on Discrete Algorithms (SODA '09)*, pages 795–804, 2009.

[16] Martin Dietzfelbinger and Ulf Schellbach. Weaknesses of cuckoo hashing with a simple universal hash class: The case of large universes. In *Proc. 35th Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM '09)*, pages 217–228, 2009.

[17] Martin Dietzfelbinger and Christoph Weidling. Balanced allocation and dictionaries with tightly packed constant size bins. *Theor. Comput. Sci.*, 380 (1-2):47–68, 2007.

[18] Martin Dietzfelbinger and Philipp Woelfel. Almost random graphs with simple hash functions. In *Proc. 35th Annual ACM Symp. on Theory of Computing (STOC '03)*, pages 629–638, 2003.

[19] Martin Dietzfelbinger, Torben Hagerup, Jyrki Katajainen, and Martti Penttonen. A reliable randomized algorithm for the closest-pair problem. *J. Algorithms*, 25(1):19–51, 1997.

[20] Michael Drmota and Reinhard Kutzelnigg. A precise analysis of cuckoo hashing. In *preprint*, 2008.

[21] Daniel Fernholz and Vijaya Ramachandran. The *k*-orientability thresholds for gn, p. In *Proc. 18th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA '07)*, pages 459–468. SIAM, 2007.

[22] P. Finsler. Über die Primzahlen zwischen n und 2n. In *Festschrift zum 60. Geburtstag von Prof. Dr. Andreas Speiser*, pages 118–122. Füssli, 1945.

[23] Dimitris Fotakis, Rasmus Pagh, Peter Sanders, and Paul G. Spirakis. Space efficient hash tables with worst case constant access time. In *Proc. 20th Annual Symp. on Theoretical Aspects of Computer Science (STACS '03)*, pages 271–282. Springer-Verlag, 2003.

[24] Torben Hagerup and C. Rüb. A guided tour of chernoff bounds. *Inf. Process. Lett.*, 33(6):305–308, 1990.

[25] Torben Hagerup and Torsten Tholey. Efficient minimal perfect hashing in nearly minimal space. In *Proc. 18th Annual Symp. on Theoretical Aspects of Computer Science (STACS '01)*, pages 317–326. Springer-Verlag, 2001.

[26] Russell Impagliazzo, Leonid A. Levin, and Michael G. Luby. Pseudo-random generation from one-way functions. In *Proc. 21st Annual ACM Symp. on Theory of Computing (STOC '89)*, pages 12–24, 1989.

[27] Adam Kirsch, Michael Mitzenmacher, and Udi Wieder. More robust hashing: Cuckoo hashing with a stash. In *Proc. 16th Annual European Symp. on Algorithms (ESA '08)*, pages 611–622. Springer-Verlag, 2008.

[28] Donald E. Knuth. *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, 2 edition, 1982.

[29] Reinhard Kutzelnigg. Bipartite random graphs and cuckoo hashing. In *Proc. 4th Colloquium on Mathematics and Computer Science*, pages 403–406, 2006.

[30] Michael Luby and Avi Wigderson. Pairwise independence and derandomization. *Found. Trends Theor. Comput. Sci.*, 1(4):237–301, 2006.

[31] Michael Mitzenmacher and Eli Upfal. *Probability and Computing*. Cambridge University Press, 2005.

[32] Michael Mitzenmacher and Salil Vadhan. Why simple hash functions work: exploiting the entropy in a data stream. In *Proc. 19th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA '08)*, pages 746–755, 2008.

[33] Noam Nisan and Amnon Ta-Shma. Extracting randomness: A survey and new constructions. *J. Comput. Syst. Sci.*, 58(1):148–173, 1999.

[34] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[35] Anna Östlin and Rasmus Pagh. Uniform hashing in constant time and linear space. In *Proc. 35th Annual ACM Symp. on Theory of Computing (STOC '03)*, pages 622–628, 2003.

[36] Andrea Ott. Studienarbeit: Experimenteller Vergleich einiger neuer Verfahren zur Konstruktion von dynamischen Wörterbüchern mit fixer Zugriffszeit. 2004.

[37] Anna Pagh and Rasmus Pagh. Uniform hashing in constant time and optimal space. *SIAM J. Comput.*, 38(1):85–96, 2008.

[38] Anna Pagh, Rasmus Pagh, and Milan Ružić. Linear probing with constant independence. In *Proc. 39th Annual ACM Symp. on Theory of Computing (STOC '07)*, pages 318–327, 2007.

[39] Rasmus Pagh. On the cell probe complexity of membership and perfect hashing. In *Proc. 33rd Annual ACM Symp. on Theory of Computing (STOC '01)*, pages 425–432, 2001.

[40] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *Springer LNCS*, 2161:121–133, 2001.

[41] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *J. Algorithms*, 51(2):122–144, 2004.

[42] Rina Panigrahy. Efficient hashing with lookups in two memory accesses. In *Proc. 16th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA '05)*, pages 830–839, 2005.

[43] Ronen Shaltiel. Recent developements in explicit constructions of extractors. *EATCS Bulletin*, 77:67–95, 2002.

[44] Alan Siegel. On universal classes of fast high performance hash functions, their time-space tradeoff, and their applications. In *Proc. 30th Annual Symp. on Foundations of Computer Science (SFCS '89)*, pages 20–25. IEEE, 1989.

[45] Alan Siegel. On universal classes of extremely random constant-time hash functions. *SIAM J. Comput.*, 33(3):505–543, 2004.

[46] David Zuckerman. Simulating BPP Using a General Weak Random Source. *Algorithmica*, 16(4/5):367–391, 1996.