Entropy Suffices for Key Guessing

Timo Glaser, Alexander May, and Julian Nowakowski

Ruhr-University Bochum, Bochum, Germany
{timo.glaser,alex.may,julian.nowakowski}@rub.de

Abstract. Modern (lattice-based) cryptosystems typically sample their secret keys *component-wise* and independently from a discrete probability distribution χ. For instance, Kyber has secret key entries from the centered binomial distribution, Dilithium from the uniform distribution, and Falcon from the discrete Gaussian. As attacks may require guessing of a subset of the secret key coordinates, the complexity of enumerating such sub-keys is of fundamental importance.
Any length-n sub-key with entries sampled from χ has entropy $\mathrm{H}(\chi)n$, where $\mathrm{H}(\chi)$ denotes the entropy of χ. If χ is the uniform distribution, then it is easy to see that any length-n sub-key can be enumerated with $2^{\mathrm{H}(\chi)n}$ trials. However, for sub-keys sampled from general probability distributions, Massey (1994) ruled out that the number of key guesses can be upper bounded by a function of the entropy alone.
In this work, we bypass Massey's impossibility result by focussing on the typical cryptographic setting, where key entries are sampled independently component-wise from some distribution χ, i.e., we focus on χ^n.
We study the optimal key guessing algorithm that enumerates keys in χ^n in descending order of probability, but we abort at a certain probability. As our main result, we show that for *any discrete probability distribution* χ our aborted key guessing algorithm tries at most $2^{\mathrm{H}(\chi)n}$ keys, and its success probability asymptotically converges to $\frac{1}{2}$. Our algorithm allows for a quantum version with at most $2^{\mathrm{H}(\chi)n/2}$ key guesses. In other words, for any distribution χ, we achieve a Grover-type square root speedup, which we show to be optimal.
For the underlying key distributions of Kyber and Falcon, we explicitly compute the expected number of key guesses and their success probabilities for our aborted key guessing for all sub-key lengths n of practical interest. Our experiments strongly indicate that our aborted key guessing, while sacrificing only a factor of two in success probability, improves over the usual (non-aborted) key guessing by a run time factor exponential in n.

1 Introduction

The security of any cryptosystem has to be based on a proper choice of its secret key, which at the bare minimum protects against key guessing. As a counter example for a proper choice, the widely used DES standard had to be replaced by AES not because of structural weaknesses, but primarily due to the fact

that its keys did no longer provide sufficient security against key guessing attacks [Fou98]. Thus, a crypto designer's first check is to validate that the secret key is sufficiently hard to guess.

Sub-Key Guessing in Hybrid Attacks. Modern lattice-based cryptosystems like NTRU [HPS06,CDH+21], Kyber [BDK+18], Dilithium [DKL+18], and Falcon [FHK+18] certainly have hard to guess secret keys. However, in general there exist more sophisticated hybrid attacks on these schemes that combine guessing of sub-keys with other techniques like lattice reduction.

These more sophisticated lattice attacks include e.g. the hybrid Meet-in-the-Middle attack of Howgrave-Graham [How07] that combines guessing of sub-keys with lattice reduction on the primal lattice. This hybrid attack was used to analyze NTRU's security [CDH+21], and Nguyen [Ngu21] showed how to boost the attack by providing a more efficient sub-key guessing technique.

The more recent attack of Guo-Johannson [GJ21] considers LWE's dual lattice, and balances the complexity of sub-key guessing and lattice reduction, similarly to Howgrave-Graham's hybrid attack. The Guo-Johannson attack is extended and refined in MATZOV [IDF22]. MATZOV claims improved attack complexities, presumably reducing the security of e.g. Kyber and Dilithium below the required NIST security level. The recent work of Ducas and Pulles [DP23], however, heavily questions some of the heuristics used in the analysis of [GJ21,IDF22].

The Complexity of Key Guessing. Among the heuristics used in [IDF22] is the estimation of the complexity of key guessing. Let a length-n sub-key be sampled coordinate-wise, independently from some probability distribution χ. The authors of [IDF22] estimate the key guessing complexity as $2^{\mathrm{H}(\chi)n}$, whereas Ducas and Pulles [DP23] criticize that this estimation lacks theoretical justification.

In this work, we provide the missing theoretical justification for the $2^{\mathrm{H}(\chi)n}$ estimate. [1] Before we dive into our results, let us first provide some intuition as to why on the one hand $2^{\mathrm{H}(\chi)n}$ appears to be a natural bound, and on the other hand such an upper bound for key guessing has not yet been proven, despite its fundamental nature and importance for cryptography as a whole.

In the past, most cryptographic schemes chose their secret keys from the uniform distribution χ^n. Let χ take m values with probability $\frac{1}{m}$ each, then $\mathrm{H}(\chi) = \log m$ and $\mathrm{H}(\chi^n) = n \log m$. Enumerating all keys costs $m^n = 2^{\mathrm{H}(\chi^n)}$ trials. Thus, for the uniform distribution χ^n we can (trivially) upper bound key guessing as a function of $\mathrm{H}(\chi^n)$.

Massey [Mas94] showed that any key guessing algorithm has to make at least $\frac{1}{4}2^{\mathrm{H}(\chi)n} - 1$ trials on average. However, Massey also showed that in general there is no matching upper bound. More precisely, he constructed counter-example distributions, for which key guessing cannot be upper bounded by a function of their entropy alone. The latter observation of Massey is usually taken as an

[1] This however neither contradicts the results of Ducas and Pulles [DP23], nor does it heal other issues in the analysis of [GJ21,IDF22].

impossibility result in the cryptographic community. However, Massey's counter-example does not rule out that for the most relevant crytographic setting of distributions χ^n, where every component is independently sampled from χ, key guessing might be upper bounded by a function of $\mathrm{H}(\chi^n)$ after all.

The existence of an entropy-dependent upper bound is supported by the following compression argument from information theory. We know that the output of a source that samples n times from a probability distribution χ can asymptotically be compressed into $(1+\delta)\,\mathrm{H}(\chi)n$ bits for any constant $\delta > 0$, see e.g. [MU17]. This already gives us a key guessing algorithm that enumerates the compressed keys with $2^{(1+\delta)\,\mathrm{H}(\chi)n}$ trials. However, this argument only reaches the desired bound of $2^{\mathrm{H}(\chi)n}$ up to a term $2^{\delta n}$ exponential in n.

In the light of our information theoretic argument, it does not come as a surprise that recent upper bounds for key guessing from a discrete Gaussian distribution [AS22] are exponentially in n away from the entropy bound. Similarly, experiments for the centered binomial distribution [DP23] indicate that key guessing requires an additional exponential factor in n as well.

On quantum computers, the famous Grover search [Gro96a] allows to achieve (up to) square root speedups over classical key guessing. However, a generalization of Grover search by Montanaro [Mon11] opened the door for even larger speedups for key guessing. Namely, Montanaro explicitly constructed distributions (different from product distributions χ^n), for which his quantum algorithm achieves exponential speedups over any classical key guessing algorithm. The algorithm of Montanaro was used in [AS22] for quantum key guessing for the discrete Gaussian distribution, using QRAM access.

Our results. We study the most common setting of cryptographic keys from an n-fold product distribution χ^n, where our result holds for *any* probability distribution χ.

Similar to previous work, we study the optimal key guessing algorithm that enumerates keys in descending order of probability. To this end, we extend the enumeration strategy of Budroni and Mårtenson [BM23]. However, in contrast to previous work, we drop the limiting restriction that our key guessing algorithm has to succeed with probability 1. Instead, we abort our key guessing when the success probability for the remaining keys drops below the threshold $2^{-\,\mathrm{H}(\chi)n}$.

This enables us to show that our aborted key guessing uses at most $T = 2^{\mathrm{H}(\chi)n}$ trials. Moreover, using the Central Limit Theorem, we show that the success probability ε (taken over all keys) of our aborted key guessing converges to $\frac{1}{2}$. Thus, with our aborted key guessing, we provide an algorithm that, for the first time, achieves a ratio $\frac{T}{\varepsilon}$ that is upper bounded by $2^{\mathrm{H}(\chi)n+1}$, whereas a previous result of Albrecht, Shen [AS22] achieved a ratio $\frac{T}{\varepsilon}$ for discrete Gaussians that is inferior by a factor exponential in n.

In a nutshell, our results show that for all n-fold key distributions χ^n, it does not pay off to guess unlikely keys[2]. Namely, in order to go significantly

[2] For the uniform distribution, in our terminology all keys are likely.

above success probability $\frac{1}{2}$, one has to pay an excessive, disproportionally large exponential overhead.

We also provide a quantum Grover-type version of our classical aborted key guessing algorithm that achieves a square root speedup, i.e, we quantumly achieve key guessing within $2^{\mathrm{H}(\chi)n/2}$ trials and success probability $\frac{1}{2}$. As opposed to the quantum algorithm of [AS22], our quantum algorithm does not require QRAM. Moreover, we prove a matching lower bound for the number of trials for any quantum key guessing algorithm of $\frac{1}{\mathrm{poly}(n)}2^{\mathrm{H}(\chi)n/2}$. This shows that our quantum algorithm is optimal up to polynomial factors. Moreover, our matching lower bound also rules out that we can achieve more than polynomial speedups with Montanaro's algorithm for the cryptographic setting of probability distributions χ^n.

We also study the impacts of our asymptotic results for concrete cryptographic settings. To this end, we study the centered binomial distributions used in Kyber, and the discrete Gaussian distributions used in Falcon for reasonably sized n. Our experiments show that, for these cryptographically relevant distributions,

(1) the convergence to success probability $\varepsilon = \frac{1}{2}$ is from above, i.e., for almost all n we achieve $\varepsilon \geq \frac{1}{2}$,
(2) our entropy-based upper bound $2^{\mathrm{H}(\chi)n}$ tightly matches the maximum number of trials of our aborted key guessing (up to a $\sqrt{n}$ factor), i.e., the entropy bound is essentially optimal,
(3) our aborted key guessing outperforms the usual key guessing (without aborts) by a factor exponential in n, i.e., we observe an exponential factor speedup with our aborted key guessing,
(4) the number of trials in our Grover-type version of aborted key guessing tightly matches the number of trials in Montanoro's key guessing algorithm (again up to roughly a factor of $\sqrt{n}$), i.e., Montanaro's algorithm does not provide significant speedups for probability distributions χ^n.

Organization of our paper. After fixing some preliminaries in Section 2, we introduce and analyze our aborted key guessing algorithm in Section 3. Its quantum version is provided in Section 4. The experimental results for the centered binomial distribution of Kyber and the centered Gaussian distribution of Falcon are presented in Section 5.

Source code. We provide the source code for our experimental results from Section 5 via `https://anonymous.4open.science/r/Entropy`.

2 Preliminaries

Throughout the paper, all probability distributions are discrete. We write $X \leftarrow \chi$ to denote that a random variable X is drawn from some probability distribution χ. Expected value and variance of X are denoted by $\mathbb{E}[X]$ and $\mathrm{Var}[X]$,

respectively. For a probability distribution χ over some set A, the *probablity mass function* of χ is defined as

$$P : A \to [0,1],\ a \mapsto \Pr_{X \leftarrow \chi}[X = a].$$

The *support* of χ is defined as $\mathrm{supp}(\chi) := \{a \in A \mid P(a) > 0\}$. The base-2 logarithm is denoted by $\log(\cdot)$.

Definition 2.1 (Entropy). *Let χ be a probability distribution with support A and probability mass function $P : A \to (0,1]$. The* entropy *of χ is defined as*

$$\mathrm{H}(\chi) := -\sum_{a \in A} P(a) \log P(a) = \mathop{\mathbb{E}}_{X \leftarrow \chi}[-\log P(X)].$$

We note that entropy is usually defined with respect to random variables. However, for our purposes, Definition 2.1 is more convenient.

We use the following variant of the Central Limit Theorem.

Lemma 2.2 (Berry-Esseen Theorem [Ber41,Ess45]). *Let $X_1, X_2, \ldots$ be a sequence of i.i.d random variables with $\mathbb{E}[X_i] < \infty$, $0 < \mathrm{Var}[X_i] < \infty$ and $\mathbb{E}[|X_i|^3] < \infty$. Define $\mu := \mathbb{E}[X_i]$, $\sigma^2 := \mathrm{Var}[X_i]$ and $\overline{X_n} := \frac{1}{n}\sum_{i=1}^{n} X_i$. Then the distribution of $\sqrt{n}(\overline{X_n} - \mu)$ converges to a Gaussian distribution with mean 0 and variance σ^2 at rate $\mathcal{O}(1/\sqrt{n})$. That is, for every $t \in \mathbb{R}$ it holds true that*

$$\Pr[\sqrt{n}(\overline{X_n} - \mu) \le t] = \int_{-\infty}^{t} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \mathrm{d}x \pm \mathcal{O}\left(\frac{1}{\sqrt{n}}\right).$$

Lemma 2.3 (Grover's Algorithm [Gro96b,Høy00,BHMT02]). *Let $|\Psi\rangle$ be a uniform superposition over some finite set A, and let $\tau : A \to \{0,1\}$ be a function, such that $\tau(a) = 1$ for at most one $a \in A$. Given $|\Psi\rangle$ and oracle access to τ, Grover's algorithm outputs $a \in A$ with $\tau(a) = 1$, if it exists, and an error symbol $\perp$ otherwise. Grover's algorithm achieves this, using $\lceil \frac{\pi}{4}\sqrt{|A|} \rceil + 1$ queries to τ.*

3 Classical Key Guessing

In this section, we introduce our new classical key guessing algorithm, which has *worst case* runtime at most $2^{\mathrm{H}(\chi)n}$ and asymptotic success probability $\frac{1}{2}$. We start by defining the *key guessing problem*, and provide a high-level description of our aborted key guessing algorithm in Section 3.1. Its runtime and success probability are analyzed in Section 3.2.

Additionally, in Section 3.3, we show how to efficiently instantiate the algorithm in practice, by building on an approach by Budroni and Mårtenson [BM23]. For the typical scenario arising in practice, where the underlying probability distribution is symmetric, we furthermore show that the Budroni-Mårtenson approach admits for a significant speedup.

3.1 Key Guessing Problem and Algorithms

Let us start by defining the *key guessing problem.*

Definition 3.1 (Key Guessing Problem). *Let χ be a probability distribution with finite support A and probability mass function $P : A \to (0, 1]$. Let $\mathbf{X} \leftarrow \chi^n$, and let $\tau : A^n \to \{0, 1\}$ be a predicate such that*

$$\tau(\mathbf{a}) := \begin{cases} 1 \text{ , } \textit{if } \mathbf{a} = \mathbf{X}, \\ 0 \text{ , } \textit{else.} \end{cases}$$

An instance of the key guessing problem *is to find the* key $\mathbf{X}$ *on input n, a description of P and A, and oracle access to τ.*

We note that for typical cryptographic distributions, the condition $|A| < \infty$ imposes essentially no constraint, e.g., even when χ is a discrete Gaussian, then χ is close to a distribution χ' with finite support, and we can simply switch to the key guessing problem for χ'. In Section 5, we show how to approximate in practice the discrete Gaussians used in Falcon512 and Falcon1024.

Key Enumeration. Obviously, the optimal strategy for solving the key guessing problem *with success probability* $\varepsilon = 1$ is to enumerate all possible keys in decreasing order of probability, until the correct key is found. This strategy requires access to an algorithm KeyEnumeration with the following properties.

1. KeyEnumeration outputs all keys $\mathbf{a}[1], \mathbf{a}[2], \ldots$ in order of decreasing probability, i.e., we have $P(\mathbf{a}[i]) \geq P(\mathbf{a}[i + 1])$ for $1 \leq i < |A^n|$.
2. KeyEnumeration outputs the keys only on demand, as a stream. That it, on its first invocation it outputs $\mathbf{a}[1]$, and keeps its state. On its second invocation, KeyEnumeration outputs $\mathbf{a}[2]$, etc.
   In particular, KeyEnumeration's runtime solely depends on the number of invocations and a polynomial in n, but not on $|A^n|$.

For the moment, we simply assume the existence of such an algorithm KeyEnumeration. Thus, the complexity of all key guess algorithms is the number of invocations of KeyEnumeration, i.e., the number of key trials.

In Section 3.3, we will show how to efficiently instantiate KeyEnumeration. We also efficiently implemented KeyEnumeration as the basis for our experimental validations in Section 5.

Key Guess Algorithm. The ordinary key guess algorithm KeyGuess is provided in Algorithm 1.

Let us denote by $p_i := P(\mathbf{a}[i])$ the probability of the i-th most likely key in A^n. Then KeyGuess has expected number of key trials

$$\mathbb{E}[T_{\mathsf{KG}}] = \sum_{i=1}^{|A^n|} p_i \cdot i. \tag{1}$$

Tightly bounding Eq. (1) for distributions of cryptographic interest like the centered binomial or discrete Gaussian distribution is an open problem, let alone for arbitrary distributions χ.

Algorithm 1: KeyGuess

Input: Key guessing instance (n, P, A, τ),
access to algorithm KeyEnumeration(n, P, A)
Output: Key $\mathbf{X} \in A^n$ satisfying $\tau(\mathbf{X}) = 1$

```
1 i ← 1; a[1] ← KeyEnumeration(n, P, A);
2 while τ(a[i]) ≠ 1 do
3 |  i ← i + 1; a[i] ← KeyEnumeration(n, P, A)
4 end
5 return a[i]
```

The Core Set. Our algorithm builds on top of KeyGuess, but aborts once keys become too unlikely. More precisely, we only enumerate keys lying in the *core set*, as defined below in Definition 3.2.

This simple modification slightly lowers our success probability asymptotically to $\varepsilon = \frac{1}{2}$. In turn, it allows us to easily bound the number of key guesses for *any* distribution χ^n by $2^{\mathrm{H}(\chi^n)}$. As we demonstrate in Section 5 experimentally, in comparison to Eq. (1), we save an exponential factor of key guesses by sacrificing only a factor of 2 of success probability.

Definition 3.2 (Core Set). *Let χ be a probability distribution with support A and probability mass function $P : A \to [0, 1]$. The* core set *of χ^n is defined as*

$$\mathcal{C}_\chi^n := \left\{ (a_1, \ldots, a_n) \in A^n \mid \prod_{i=1}^{n} P(a_i) \geq 2^{-\mathrm{H}(\chi)n} \right\}.$$

Notice that the product $\prod_{i=1}^{n} P(a_i)$ in Definition 3.2 is the probability

$$\Pr_{\mathbf{X} \leftarrow \chi^n}[\mathbf{X} = (a_1, \ldots, a_n)] = \prod_{i=1}^{n} P(a_i).$$

Aborted Key Guessing. Our modified key guessing algorithm AbortedKeyGuess (Algorithm 2) now simply aborts once we have exhausted all keys from the core set $\mathcal{C}_\chi^n$.

We first show that KeyGuess and AbortedKeyGuess behave identical on the uniform distribution χ, and succeed to recover the key in the desired amount of key trials.

Theorem 3.3. *Let χ be the uniform distribution with probability mass function $P : A \to (0, 1]$. Then* KeyGuess *and* AbortedKeyGuess *solve any key guessing instance (n, P, A, τ) with probability 1 using at most $2^{\mathrm{H}(\chi)n}$ key trials.*

Proof. Since χ has finite support A, we have $m = |A| < \infty$,

$$\mathrm{H}(\chi) = \sum_{i=1}^{m} \frac{1}{m} \cdot \log(m) = \log(m),$$

Algorithm 2: AbortedKeyGuess (high level description)

Input: Key guessing instance (n, P, A, τ),
access to algorithm KeyEnumeration(n, P, A)
Output: Key $\mathbf{X} \in A^n$ satisfying $\tau(\mathbf{X}) = 1$ or $\perp$ (abort).

```
1 i ← 1; a[1] ← KeyEnumeration(n, P, A);
2 while (τ(a[i]) ≠ 1) and (P(a[i]) ≥ 2^{-H(χ)n}) do
3 |  i ← i + 1; a[i] ← KeyEnumeration(n, P, A)
4 end
5 if τ(a[i]) = 1 then return a[i];
6 else return ⊥;
```

and therefore

$$2^{-\mathrm{H}(\chi)n} = m^{-n}.$$

Since all $|A^n| = m^n$ key have the same probability m^{-n}, the core set condition $P(\mathbf{a}[i]) \geq 2^{-\mathrm{H}(\chi)n}$ in line 2 of AbortedKeyGuess is always satisfied. Therefore KeyGuess and AbortedKeyGuess behave identical, and both succeed to recover the key with probability 1 after at most $|A^n| = 2^{\mathrm{H}(\chi)n}$ key trials. □

3.2 Analysis of AbortedKeyGuess for any distribution

Recall that AbortedKeyGuess aborts after exhausting all keys from the core set $\mathcal{C}_\chi^n$. In our main theorem, we show that the following two statements hold for any distribution χ:

(1) We have $|\mathcal{C}_\chi^n| \leq 2^{\mathrm{H}(\chi)n}$, bounding the number of key trials as desired.
(2) Asymptotically, a random key $\mathbf{X} \leftarrow \chi^n$ lies in $\mathcal{C}_\chi^n$ with probability $\frac{1}{2}$.

Theorem 3.4 (Main Theorem). *Let χ be any (but the uniform[3]) distribution with probability mass function $P : A \to (0, 1]$. Then* AbortedKeyGuess *solves a random key guessing instance (n, P, A, τ), taken over the random key choice $\mathbf{X} \leftarrow \chi^n$, with probability $\frac{1}{2} \pm \mathcal{O}(\frac{1}{\sqrt{n}})$, using at most $2^{\mathrm{H}(\chi)n}$ key trials.*

Proof. The number of key trials in AbortedKeyGuess is at most $|\mathcal{C}_\chi^n|$. Let us first show that $|\mathcal{C}_\chi^n| \leq 2^{\mathrm{H}(\chi)n}$. Let $P_n : A^n \to (0, 1]$ denote the probability mass function of χ^n. By definition of $\mathcal{C}_\chi^n$, it holds that

$$1 = \sum_{\mathbf{a} \in A^n} P_n(\mathbf{a}) \geq \sum_{\mathbf{a} \in \mathcal{C}_\chi^n} P_n(\mathbf{a}) \geq \sum_{\mathbf{a} \in \mathcal{C}_\chi^n} 2^{-\mathrm{H}(\chi)n} = |\mathcal{C}_\chi^n| 2^{-\mathrm{H}(\chi)n}.$$

Multiplying the above inequality by $2^{\mathrm{H}(\chi)n}$, we obtain $|\mathcal{C}_\chi^n| \leq 2^{\mathrm{H}(\chi)n}$.

It remains to show that a random key $\mathbf{X} = (X_1, \ldots, X_n) \leftarrow \chi^n$ lies in the core set with probability $\Pr_{\mathbf{X} \leftarrow \chi^n}\left[\mathbf{X} \in \mathcal{C}_\chi^n\right] = \frac{1}{2} \pm \mathcal{O}(\frac{1}{\sqrt{n}})$. Since P is the probability

[3] For the uniform distribution see the stronger statement from Theorem 3.3.

mass function of χ, by definition of $\mathcal{C}_\chi^n$ it holds that

$$\Pr\left[\mathbf{X} \in \mathcal{C}_\chi^n\right] = \Pr\left[\prod_{i=1}^{n} P(X_i) \geq 2^{-\mathrm{H}(\chi)n}\right].$$

W.l.o.g we assume $P > 0$, which allows us to define $Y_i := -\log P(X_i)$. We set $\overline{Y_n} := \frac{1}{n}\sum_{i=1}^{n} Y_i$, and rewrite the above probability as

$$\Pr\left[\mathbf{X} \in \mathcal{C}_\chi^n\right] = \Pr\left[-\sum_{i=1}^{n} Y_i \geq -\mathrm{H}(\chi)n\right] = \Pr\left[\overline{Y_n} - \mathrm{H}(\chi) \leq 0\right].$$

We now make three important observations:

1. By definition of entropy, $\mathbb{E}[Y_i] = \mathrm{H}(\chi) < \infty$.
2. Since χ is not the uniform distribution, Y_i is not constant and thus we have $\mathrm{Var}[Y_i] > 0$.
3. Since χ has finite support, both $\mathrm{Var}[Y_i]$ and $\mathbb{E}[|Y_i|^3]$ are finite.

By the Berry-Esseen Theorem (Lemma 2.2), the distribution of $\sqrt{n}(\overline{Y_n} - \mathrm{H}(\chi))$ thus converges at rate $\mathcal{O}(1/\sqrt{n})$ to a Gaussian distribution with mean 0 and variance $\sigma^2 := \mathrm{Var}[Y_i]$. Hence,

$$\begin{aligned}\Pr\left[\mathbf{X} \in \mathcal{C}_\chi^n\right] &= \Pr\left[\overline{Y_n} - \mathrm{H}(\chi) \leq 0\right] = \Pr\left[\sqrt{n}(\overline{Y_n} - \mathrm{H}(\chi)) \leq 0\right] \\ &= \int_{-\infty}^{0} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \mathrm{d}x \pm \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) \\ &= \frac{1}{2} \pm \mathcal{O}\left(\frac{1}{\sqrt{n}}\right),\end{aligned}$$

which proves our main theorem. □

3.3 Efficiently Enumerating the Core Set

Let us now show how to construct an efficient algorithm KeyEnumeration that outputs all keys $\mathbf{a}[1], \mathbf{a}[2], \ldots$ in decreasing order of probability, as required for algorithms KeyGuess and AbortedKeyGuess. We build on an approach suggested by Budroni and Mårtenson [BM23].

Budroni-Mårtenson Revisited. Let (n, P, A, τ) be a key guessing instance, where P is the probability mass function of some distribution to χ. Recall that for $\mathbf{X} \leftarrow \chi^n$ and $\mathbf{a} = (a_1, \ldots, a_n)$, it holds that

$$\Pr[\mathbf{X} = \mathbf{a}] = \prod_{i=1}^{n} P(a_i).$$

If $\mathbf{a}' \in A^n$ is a permutation of $\mathbf{a}$, then it follows that

$$\Pr[\mathbf{X} = \mathbf{a}] = \Pr[\mathbf{X} = \mathbf{a}'].$$

Based on this simple, yet important observation, Budroni and Mårtenson suggest to represent χ^n via the following set, which we call a *compact dictionary* of χ^n.

Definition 3.5 (Compact Dictionary). *Let χ be a probability distribution with finite support A and probability mass function $P : A \to (0, 1]$, and let $n \in \mathbb{N}$. Let $\widetilde{A^n}$ denote a largest subset of A^n, such that no distinct $\mathbf{a}, \mathbf{a}' \in \widetilde{A^n}$ are permutations of each other. Then we call the following set*

$$\mathcal{D}_\chi^n := \left\{ \left((a_1, \ldots, a_n), \prod_{i=1}^{n} P(a_i) \right) \mid (a_1, \ldots, a_n) \in \widetilde{A^n} \right\}$$

a compact dictionary *of χ^n.*

By construction, any compact dictionary $\mathcal{D}_\chi^n$ contains all probabilities, that the probability mass function of χ^n assumes. This allows us to easily enumerate all elements of A^n in order of decreasing probability as follows: Given A and P, we construct a compact dictionary $\mathcal{D}_\chi^n$ and then sort it by its second component in decreasing order. After that, we simply iterate over (the sorted) $\mathcal{D}_\chi^n$ and enumerate for every tuple $(\mathbf{a}, p) \in \mathcal{D}_\chi^n$ all permutations of $\mathbf{a}$. A formal description of this approach is given in KeyEnumeration (Algorithm 3).

Algorithm 3: KeyEnumeration

Input: n, P, A
Output: Keys $\mathbf{a}[1], \mathbf{a}[2], \ldots$ with $P(\mathbf{a}[i]) \geq P(\mathbf{a}[i+1])$ for all $i < |A^n|$

```
Construct a compact dictionary D_χ^n = {(a, ∏_{i=1}^n P(a_i)) | a ∈ A^n} .
Sort D_χ^n by 2nd component in decreasing order of probabilities ∏_{i=1}^n P(a_i).
i ← 1
foreach (a, p) ∈ D_χ^n do
    foreach permutation a' of a do
        Output a[i] := a'; i ← i + 1;
    end
end
```

As the following Theorem 3.6 shows, the size $|\mathcal{D}_\chi^n|$ of a compact dictionary is polynomial in n, and $\mathcal{D}_\chi^n$ can be efficiently constructed. The Budroni-Mårtenson approach for representing χ^n compactly via $\mathcal{D}_\chi^n$ thus significantly improves over the naive approach of storing $\prod_{i=1}^n P(a_i)$ for *all* $(a_1, \ldots, a_n) \in A^n$, since it reduces the required runtime and amount of memory from exponential in n to polynomial in n.

Theorem 3.6 (Budroni, Mårtenson [BM23]). *Let χ be a probability distribution with finite support A and probability mass function $P : A \to [0, 1]$. For every compact dictionary $\mathcal{D}_\chi^n$, it holds that*

$$|\mathcal{D}_\chi^n| = \binom{n + |A| - 1}{n} = \mathcal{O}(n^{|A|-1}). \tag{2}$$

Furthermore, there exists an algorithm with runtime $\mathcal{O}(n^{|A|-1})$, that on input A and P outputs $\mathcal{D}_\chi^n$.

Proof. Let $\widetilde{A^n}$ be defined as in Definition 3.5, and write $A = \{a_1, \ldots, a_m\}$, where $m := |A|$. For any $\mathbf{a} \in \widetilde{A^n}$, let $\omega_i(\mathbf{a})$ denote the number of coordinates of $\mathbf{a}$, that are equal to a_i.

By definition of $\widetilde{A^n}$, the following map φ is a bijection

$$\begin{array}{rcl} \varphi : \widetilde{A^n} & \to & \{(\alpha_1, \ldots, \alpha_m) \in \mathbb{N}^m \mid \sum_{i=1}^m \alpha_i = n\}, \\ \mathbf{a} & \mapsto & (\omega_1(\mathbf{a}), \ldots, \omega_m(\mathbf{a})). \end{array} \tag{3}$$

Recall that there are exactly $\binom{n+m-1}{n}$ ways to write n as the sum of m non-negative integers. Hence, $|\widetilde{A^n}| = \binom{n+m-1}{n}$.

Together with $|\mathcal{D}_\chi^n| = |\widetilde{A^n}|$ and

$$\binom{n+m-1}{n} = \frac{1}{(m-1)!} \prod_{i=1}^{m-1} (n+i) = \mathcal{O}(n^{m-1}),$$

this proves Equation (2).

To prove that there exists an $\mathcal{O}(n^{|A|-1})$-time algorithm for constructing $\mathcal{D}_\chi^n$, simply observe that the bijection φ from Equation (3) allows us to efficiently construct $\widetilde{A^n}$, from which we then easily construct $\mathcal{D}_\chi^n$. □

Improvement for Symmetric Distributions. In practice, χ usually is a symmetric distribution over $\mathbb{Z}$ with center 0, i.e., for $X \leftarrow \chi$ and $a \in \mathbb{Z}$ it holds that

$$\Pr[X = a] = \Pr[X = -a].$$

Consequently, for $\mathbf{X} \leftarrow \chi^n$ and $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{Z}^n$, it then holds that

$$\Pr[\mathbf{X} = \mathbf{a}] = \Pr[\mathbf{X} = (|a_1|, \ldots, |a_n|)],$$

i.e., the probability of $\mathbf{a}$ does not depend on the signs of the a_i's.

In that case, we may safely remove all tuples $(\mathbf{a}, p)$ from our compact dictionary $\mathcal{D}_\chi^n$ in which $\mathbf{a}$ has at least one negative coordinate, and still obtain a dictionary that fully describes the distribution χ^n.[4] As the following theorem shows, this allows us to save an additional square root in runtime and memory for constructing a compact dictionary.

Theorem 3.7. *Let χ be a symmetric probability distribution with finite support $A \subset \mathbb{Z}$ and probability mass function $P : A \to (0, 1]$. Let $\mathcal{D}_\chi^{n,\geq 0}$ denote the set, that is obtained by removing all tuples $(\mathbf{a}, p)$ from a compact dictionary $\mathcal{D}_\chi^n$, where $\mathbf{a}_i$ has at least one negative coordinate. Then it holds that*

$$|\mathcal{D}_\chi^{n,\geq 0}| = \binom{n + \frac{|A|-1}{2}}{n} = \mathcal{O}\left(n^{\frac{|A|-1}{2}}\right).$$

Furthermore, there exists an algorithm with runtime $\mathcal{O}\left(n^{\frac{|A|-1}{2}}\right)$, that on input A and p outputs $\mathcal{D}_\chi^{n,\geq 0}$.

[4] In this case, we modify line 5 of Algorithm 3 to iterate over all *signed* permutations of $\mathbf{a}'$ of $\mathbf{a}$.

Proof. As the proof is very similar to that of Theorem 3.6, we only give a rough outline. Let $B := \{a \in A \mid a \geq 0\}$ and $m := |B|$. Let $\widetilde{B^n}$ denote a largest subset of B^n, such that no distinct $\mathbf{b}, \mathbf{b}' \in \widetilde{B^n}$ are permutations of each other. Then there is a natural bijection between $\widetilde{B^n}$ and the following set

$$\{(\alpha_1, \ldots, \alpha_m) \in \mathbb{N}^m \mid \textstyle\sum_{i=1}^m \alpha_i = n\},$$

similar to the bijection from Equation (3). Now arguing exactly as in the proof of Theorem 3.6, it easily follows that

$$|\mathcal{D}_\chi^{n,\geq 0}| = |\widetilde{B^n}| = \binom{n+m-1}{n} = \binom{n+\frac{|A|-1}{2}}{n} = \mathcal{O}\left(n^{\frac{|A|-1}{2}}\right),$$

and that there exists an $\mathcal{O}\left(n^{\frac{|A|-1}{2}}\right)$-time algorithm for constructing $\mathcal{D}_\chi^{n,\geq 0}$. □

4 Quantum Key Guessing

In this section, we study the quantum complexity of the key guessing problem. In Section 4.1, we give a simple quantum key guessing algorithm, which achieves a square root speedup over the runtime of our classical algorithm from Section 3, while maintaining its asymptotic success probability of $\frac{1}{2}$. We explicitly show in Section 4.2 how to efficiently instantiate the required superpositions in our algorithm, *without* QRAM. Here, we build on techniques developed in Section 3.3. Finally, we show in Section 4.3 that the runtime of our algorithm is optimal up to polynomial factors.

4.1 A Simple $2^{\mathbf{H}(\chi)n/2}$-Time Quantum Algorithm

Recall that our classical algorithm AbortedKeyGuess (Algorithm 2) from Section 3 simply enumerates all keys that lie in the core set $\mathcal{C}_\chi^n$, as defined in Definition 3.2. A natural quantum version of our algorithm is to run Grover's algorithm on the core set, as depicted in Algorithm 4.

Algorithm 4: QuantumKeyGuess (high level description)

Input: Key guessing instance (n, P, A, τ).
Output: Key $\mathbf{X} \in A^n$ satisfying $\tau(\mathbf{X}) = 1$ or $\perp$.
1 Instantiate a uniform superposition $|\Psi\rangle$ over $\mathcal{C}_\chi^n$.
2 Run Grover's algorithm on $|\Psi\rangle$ with oracle access to τ and return the result.

In the subsequent Section 4.2, we show that Step 1 of QuantumKeyGuess has only polynomial complexity, even without QRAM. Together with Lemma 2.3 and Theorem 3.4, this shows that QuantumKeyGuess achieves a square root speedup over the runtime of AbortedKeyGuess, while still having asymptotic success probability of $\frac{1}{2}$. In particular, we have the following quantum version of Theorem 3.4.

Theorem 4.1. *Let χ be any (but the uniform) distribution with probability mass function $P : A \to (0, 1]$. Then* QuantumKeyGuess *solves a random key guessing instance (n, P, A, τ), taken over the random key choice $\mathbf{X} \leftarrow \chi^n$, with probability $\frac{1}{2} \pm \mathcal{O}(\frac{1}{\sqrt{n}})$ using at most $\lceil \frac{\pi}{4} 2^{\mathrm{H}(\chi)n/2} \rceil + 1$ key trials.*

4.2 Efficiently Instantiating a Superposition Over the Core Set

Using the techniques from Section 3.3, we now show how to efficiently construct a uniform superposition over the core set $\mathcal{C}_\chi^n$ in Step 1 of QuantumKeyGuess, without QRAM.

High-Level Description. Let us first fix some notation. Let (n, P, A, τ) be an instance of the key guessing problem, where P is the probability mass function of some distribution χ, and write $A = \{a_1, \ldots, a_m\}$.

The main idea for efficiently enumerating the core set $\mathcal{C}_\chi^n$ in Section 3.3 is to represent the distribution χ via a compact dictionary $\mathcal{D}_\chi^n$ (see Definition 3.5). Then, for every tuple $(\mathbf{a}, p) \in \mathcal{D}_\chi^n$ with $p \geq 2^{-\mathrm{H}(\chi)n}$, enumerate all permutations of $\mathbf{a}$. To construct a uniform superposition over $\mathcal{C}_\chi^n$, we now follow a similar approach. Loosely speaking, we first create a non-uniform superposition $|\Phi\rangle$, that represents all $(\mathbf{a}, p) \in \mathcal{D}_\chi^n$ with $p \geq 2^{-\mathrm{H}(\chi)n}$, and then apply a unitary operator Perm_{A^n} to $|\Phi\rangle$, that turns $|\Phi\rangle$ into a uniform superposition over all permutations of such $\mathbf{a}$'s.

We proceed as follows: Let $\mathbf{a} \in A^n$. Following the notation of the proof of Theorem 3.6, we denote by $\omega_i(\mathbf{a})$ the number of coordinates of $\mathbf{a}$ that are equal to a_i. We compute a compact dictionary $\mathcal{D}_\chi^n$ of χ in polynomial time (see Theorem 3.6), and initialize the following superposition

$$|\Phi\rangle := \sum_{\substack{(\mathbf{a},p)\in\mathcal{D}_\chi^n,\\ p\geq 2^{-\mathrm{H}(\chi)n}}} \sqrt{\frac{1}{|\mathcal{C}_\chi^n|}\binom{n}{\omega_1(\mathbf{a}),\ldots,\omega_m(\mathbf{a})}}\,|(\omega_1(\mathbf{a}),\ldots,\omega_m(\mathbf{a}))\rangle \otimes |\mathbf{0}^n\rangle\,.$$

Notice that by Theorem 3.6, the size of $\mathcal{D}_\chi^n$ is polynomial in n. We can therefore efficiently initialize the above superposition $|\Phi\rangle$ without QRAM.

Next, we apply an operator Perm_{A^n} to $|\Phi\rangle$, which on input of a base state

$$|(i_1, \ldots, i_m)\rangle \otimes |\mathbf{0}^n\rangle\,,$$

where $i_1, \ldots, i_m \in \mathbb{N}$ and $i_1 + \ldots + i_m = n$, results in

$$|\mathbf{0}^m\rangle \otimes |\phi\rangle\,,$$

where $|\phi\rangle$ is a uniform superposition over the following set

$$\{\mathbf{a} \in A^n \mid \omega_j(\mathbf{a}) = i_j, \text{ for all } 1 \leq j \leq m.\}\,.$$

Before we dive into the technicalities of construction of $\textsc{Perm}_{A^n}$, we note that the resulting quantum state then is a uniform superposition over the core set $\mathcal{C}^n_\chi$, i.e.,

$$\textsc{Perm}_{A^n}(|\Phi\rangle) = |\mathbf{0}^m\rangle \otimes \sum_{\mathbf{a}\in\mathcal{C}^n_\chi} \frac{1}{\sqrt{|\mathcal{C}^n_\chi|}} \cdot |\mathbf{a}\rangle\,,$$

as required.

A formal description of this approach is given in Algorithm 5.

Algorithm 5: QuantumKeyGuess (detailed description)

Input: Key guessing instance (n, P, A, τ).
Output: Key $\mathbf{X} \in A^n$ satisfying $\tau(\mathbf{X}) = 1$ or $\perp$ (abort).

1 $\mathrm{H}(\chi) := -\sum_{a\in A} P(a) \log P(a)$.
2 Construct a compact dictionary $\mathcal{D}^n_\chi$.
3 $|\Phi\rangle := \sum_{\substack{(\mathbf{a},p)\in\mathcal{D}^n_\chi,\\ p\geq 2^{-\mathrm{H}(\chi)n}}} \sqrt{\frac{1}{|\mathcal{C}^n_\chi|\binom{n}{\omega_1(\mathbf{a}),\ldots,\omega_m(\mathbf{a})}}}\,|(\omega_1(\mathbf{a}),\ldots,\omega_m(\mathbf{a}))\rangle \otimes |\mathbf{0}^n\rangle$.
4 $|\mathbf{0}^{|A|}\rangle \otimes |\Psi\rangle := \textsc{Perm}_{A^n}|\Phi\rangle$.
5 Run Grover's algorithm on $|\Psi\rangle$ with oracle access to τ and return the result.

The Perm operator. We construct $\textsc{Perm}_{A^n}$ as a product of the following operators $U_{A,k}$, using ideas from [Sch22, Chapter 5].

Definition 4.2 ($U_{A,k}$). *Let $A = \{a_1, \ldots, a_m\}$ and $n, k \in \mathbb{N}$ with $1 \leq k \leq n$. We define $U_{A,k}$ as a unitary operator, that maps*

$$|(i_1, \ldots, i_m)\rangle \otimes |\mathbf{a}, \mathbf{0}^{n-k+1}\rangle\,,$$

where $\mathbf{a} \in A^{k-1}$, and $i_1, \ldots, i_m \in \mathbb{N}$ with $i_1 + \ldots + i_m = n - k + 1$, to

$$\sum_{j=1}^{m} \sqrt{\frac{i_j}{n-k+1}}\,|(i_1, \ldots, i_{j-1}, i_j - 1, i_{j+1}, \ldots, i_m)\rangle \otimes |\mathbf{a}, a_j, \mathbf{0}^{n-k}\rangle\,.$$

It is easy to see that for fixed A, we can implement $U_{A,k}$ efficiently, even without QRAM.

We now define $\textsc{Perm}_{A^n}$ as follows.

Definition 4.3 ($\textsc{Perm}_{A^n}$). *Let A be a finite set and $n \in \mathbb{N}$. We define $\textsc{Perm}_{A^n}$ as*

$$\textsc{Perm}_{A^n}|\psi\rangle := U_{A,n} \cdot U_{A,n-1} \cdot \ldots \cdot U_{A,1}\,|\psi\rangle\,.$$

As the following theorem shows, $\textsc{Perm}_{A^n}$ yields the desired superposition.

Theorem 4.4. *Let $A = \{a_1, \ldots, a_m\}$, and let $i_1, \ldots, i_m \in \mathbb{N}$ be such that $i_1 + \ldots + i_m = n$. Define*

$$\mathcal{P} := \{\mathbf{a} \in A^n \mid \omega_j(\mathbf{a}) = i_j, \text{ for all } 1 \leq j \leq m\},$$

where $\omega_j(\mathbf{a})$ denotes the number of coordinates of $\mathbf{a}$, that are equal to a_j. Then it holds that

$$\textsc{Perm}_{A^n}\left(|(i_1, \ldots, i_m)\rangle \otimes |\mathbf{0}^n\rangle\right) = |(\mathbf{0}^m)\rangle \otimes \sum_{\mathbf{a} \in \mathcal{P}} \frac{1}{\sqrt{|\mathcal{P}|}} |\mathbf{a}\rangle .$$

Before we prove Theorem 4.4, let us illustrate the intuition behind the $\textsc{Perm}_{A^n}$ operator with an example. Suppose we have $A = \{\circ, \bullet, *\}$. We apply $\textsc{Perm}_{A^4}$ to

$$|\psi\rangle := |(2, 1, 1\rangle \otimes |\mathbf{0}^4\rangle .$$

One can easily verify that $|\psi\rangle$ then evolves as shown in Figure 1.

Notice that in every base state $|(\alpha, \beta, \gamma)\rangle\, |\mathbf{a}, \mathbf{0}^{n-k}\rangle$ in Figure 1, α, β and γ denote the amount of $\circ$'s, $\bullet$'s and $*$'s that we can append to $\mathbf{a}$ to obtain a 4-tuple, where two coordinates equal $\circ$, one coordinate equals $\bullet$ and one coordinate equals $*$. As Figure 1 shows, after application of $U_{A,4}$, the quantum state is a superposition over all such 4-tuples. Furthermore, each of the 12 base states has amplitude

$$\sqrt{\frac{1 \cdot 1 \cdot 1 \cdot 2}{4 \cdot 3 \cdot 2 \cdot 1}} = \sqrt{\frac{1}{12}}.$$

Hence, as required, the quantum state is a *uniform* superposition over the set $\mathcal{P}$ (as defined in Theorem 4.4).

Proof (Theorem 4.4). Let $k \in \mathbb{N}$ with $1 \leq k \leq n$, and fix an arbitrary $\mathbf{a} \in A^{n-k}$ with $\omega_j(\mathbf{a}) \leq i_j$, for all $1 \leq j \leq m$, where $i_1, \ldots, i_m \in \mathbb{N}$ with $i_1 + \ldots + i_m = n$. We consider the following quantum state

$$|\phi\rangle := U_{A,k} \cdot U_{A,k-1} \cdot \ldots \cdot U_{A,1}\left(|(i_1, \ldots, i_m)\rangle \otimes |\mathbf{0}^n\rangle\right).$$

By induction over k, it easily follows that the amplitude of

$$|(i_1 - \omega_1(\mathbf{a}), \ldots, i_1 - \omega_m(\mathbf{a}))\rangle \otimes |\mathbf{a}, \mathbf{0}^{n-k}\rangle ,$$

in $|\psi\rangle$ is

$$\sqrt{\frac{i_1^{\underline{\omega_1(\mathbf{a})}} \cdot \ldots \cdot i_m^{\underline{\omega_m(\mathbf{a})}}}{n^{\underline{k}}}},$$

where $x^{\underline{y}} := x \cdot (x-1) \cdot \ldots \cdot (x-y+1)$ denotes the falling factorial.

In particular, for $\mathbf{a} \in \mathcal{P}$, i.e., for $k = n$ and $\omega_j(\mathbf{a}) = i_j$, it follows that the amplitude of

$$|(\mathbf{0}^m)\rangle \otimes |\mathbf{a}\rangle ,$$

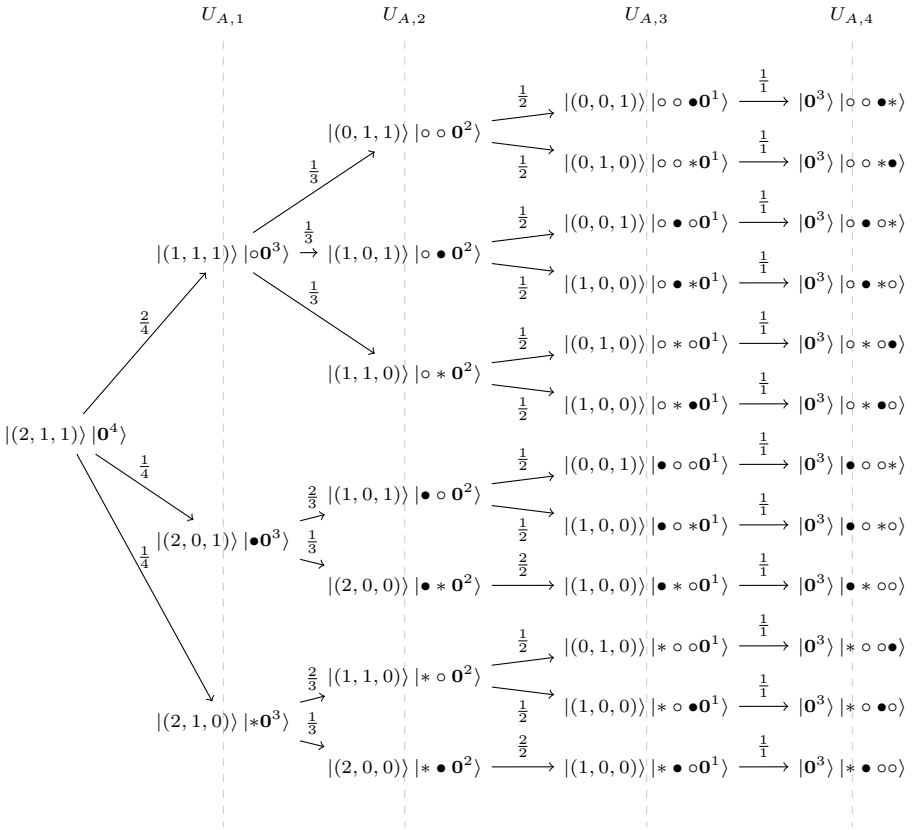

Fig. 1. Evolution of $|\psi\rangle = |(2,1,1\rangle \otimes |\mathbf{0}^4\rangle$, when applying $\textsc{Perm}^4_A$, where $A = \{\circ, \bullet, *\}$. The fractions on the edges of the graph denote the squared amplitudes.

in

$$\begin{aligned} |\psi\rangle &= U_{A,n} \cdot U_{A,n-1} \cdot \ldots \cdot U_{A,1} \left(|(i_1, \ldots, i_m)\rangle \otimes |\mathbf{0}^n\rangle \right) \\ &= \textsc{Perm}_{A^n} \left(|(i_1, \ldots, i_m)\rangle \otimes |\mathbf{0}^n\rangle \right). \end{aligned}$$

is

$$\sqrt{\frac{i_1^{\underline{i_1}} \cdot \ldots \cdot i_m^{\underline{i_m}}}{n^{\underline{n}}}} = \sqrt{\frac{i_1! \cdot \ldots \cdot i_m!}{n!}} = \sqrt{\binom{n}{i_1, \ldots, i_m}^{-1}} = \frac{1}{\sqrt{|\mathcal{P}|}}.$$

Hence, $\textsc{Perm}_{A^n} \left(|(i_1, \ldots, i_m)\rangle \otimes |\mathbf{0}^n\rangle \right)$ is a uniform superposition over $\mathcal{P}$, as required. □

4.3 A $2^{\mathrm{H}(\chi)n/2}$ Lower Bound for Quantum Algorithms

Montanaro [Mon11] proved the following lower bound on the complexity of any quantum algorithm that solves the key guessing problem with success probability 1.

Theorem 4.5 (Proposition 2.4 in [Mon11]). *Let (n, P, A, τ) be a key guessing problem where P is the probability mass function of some distribution χ with support A. Let $p_1 \geq p_2 \geq \ldots \geq p_{|A^n|}$ denote the values that the probability mass function of χ^n assumes. Every quantum algorithm that solves the key guessing instance (n, P, A, τ) with success probability 1 makes at least*

$$0.206 \cdot \sum_{i=1}^{|A^n|} p_i \sqrt{i} - 1$$

oracle queries to τ on expectation.

In [Mon11], Montanaro gave a quantum algorithm with query complexity matching the lower bound from Theorem 4.5 (up to a constant factor). Furthermore, for the special case of $n = 1$, Montanaro showed [Mon11, Corollary 2.6] that there exist distributions for which the best classical algorithm requires at least $\Omega(|A|^{1/2-\varepsilon})$ queries to τ, whereas the best quantum algorithm requires only $\Theta(1)$ – suggesting that the key guessing problem admits for a significantly better speedup than the generic Grover square root bound. However, as the following Theorem 4.6 shows, when n is not fixed to 1, then the lower bound from Theorem 4.5 is at most a polynomial factor better than the Grover bound of $2^{\mathrm{H}(\chi)n/2}$. Hence, our simple algorithm QuantumKeyGuess from Section 4.1 essentially is optimal.

We point out that one may view our Theorem 4.6 as a quantum variant of Massey's [Mas94] lower bound for the classical complexity.

Theorem 4.6. *Let (n, P, A, τ) be a key guessing instance, where $P : A \to (0, 1]$ is the probability mass function of some distribution χ with support A. Let $p_1 \geq p_2 \geq \ldots \geq p_{|A^n|}$ denote the values, that the probability mass function of χ^n assumes. Then it holds that*

$$\sum_{i=1}^{|A^n|} p_i \sqrt{i} > \frac{1}{\mathsf{poly}(n)} \cdot 2^{\mathrm{H}(\chi)n/2}.$$

Proof. Let $A := \{a_1, \ldots, a_m\}$ and $q_i := P(a_i)$. We construct an $\mathbf{a} \in A^n$, such that $q_i n$ coordinates of $\mathbf{a}$ are equal to a_i for every $i = 1, \ldots, m$. (We deliberately ignore rounding issues here, since they contribute only to polynomial factors.)

It is easy to see that for $\mathbf{X} \leftarrow \chi^n$ it holds that

$$\Pr[\mathbf{X} = \mathbf{a}] = 2^{-\mathrm{H}(\chi)n}.$$

and that A^n contains

$$\beta := \binom{n}{q_1 n, q_2 n, \ldots, q_m n} > \frac{1}{\mathsf{poly}(n)} \cdot 2^{\mathrm{H}(\chi)n}$$

permutations of $\mathbf{a}$.

It follows that there are at least $\beta/2$ terms $p_i\sqrt{i}$ in the sum $\sum_{i=1}^{|A^n|} p_i\sqrt{i}$, such that

$$p_i\sqrt{i} \geq 2^{-\mathrm{H}(\chi)n}\sqrt{\beta/2} > \frac{1}{\mathsf{poly}(n)} \cdot 2^{-\mathrm{H}(\chi)n/2}$$

Hence, the sum is lower bounded by

$$\sum_{i=1}^{|A^n|} p_i\sqrt{i} > \frac{\beta}{2} \cdot \frac{1}{\mathsf{poly}(n)} \cdot 2^{-\mathrm{H}(\chi)n/2} > \frac{1}{\mathsf{poly}(n)} \cdot 2^{\mathrm{H}(\chi)n/2},$$

proving the theorem. □

5 AbortedKeyGuess Applied to Kyber and Dilithium

In this Section, we apply our AbortedKeyGuess algorithm to the distributions χ chosen in Kyber and Falcon. Kyber [BDK+18] takes the following centered binomial distribution.

Definition 5.1. *Let $\eta \in \mathbb{N}$. We denote as* centered binomial distribution *the probability distribution over $\{-\eta, \ldots, \eta\}$ with probability distribution function*

$$p_i = \frac{\binom{2\eta}{\eta+i}}{2^{2\eta}}.$$

Sampling from this distribution is denoted by $\mathbf{X} \leftarrow \mathcal{B}(\eta)$.

Kyber512 chooses its keys from $B(3)^{512}$, whereas Kyber768 and Kyber1024 choose their keys from $\mathcal{B}(2)^{768}$ and $\mathcal{B}(2)^{1024}$, respectively.

Falcon [FHK+18] takes the following discrete Gaussian distribution.

Definition 5.2. *Let $\sigma \in \mathbb{R}_{>0}$. We denote as* discrete gaussian distribution *(centered around 0) the probability distribution over $\mathbb{Z}$ with probability distribution function*

$$p_i = \frac{\exp(\frac{-i^2}{2\sigma^2})}{\sum_{j\in\mathbb{Z}} \exp(\frac{-j^2}{2\sigma^2})}.$$

Sampling from this distribution is denoted by $\mathbf{X} \leftarrow \mathcal{D}(\sigma)$.

Falcon uses $\mathcal{D}(\sigma)$ with $\sigma = 1.17\sqrt{\frac{q}{2n}}$ for modulus q and secret key dimension n. This leads to Falcon512 keys being chosen from $\mathcal{D}(4.05)^{512}$ and Falcon1024 keys being chosen from $\mathcal{D}(2.87)^{1024}$.

To make calculations feasible, instead of using $\mathcal{D}(\sigma)$, we opted to use an approximation of $\mathcal{D}(\sigma)$ instead, denoted with $\overline{\mathcal{D}}(\sigma)$. Such an approximate distribution was obtained by sampling 2^{19} times from $\mathcal{D}(\sigma)$ and using the resulting mass as probability distribution. $\overline{\mathcal{D}}(4.05)$ and $\overline{\mathcal{D}}(2.87)$ for Falcon512 and Falcon1024 are provided in Figure 12, Appendix A, having supports $\{-20, \ldots, 20\}$ and $\{-13, \ldots, 13\}$, respectively.

n	$\mathcal{B}(2)$ ε	$\mathcal{B}(3)$ ε	$\overline{\mathcal{D}}(4.05)$ ε	$\overline{\mathcal{D}}(2.87)$ ε	n	$\mathcal{B}(2)$ ε	$\mathcal{B}(3)$ ε	$\overline{\mathcal{D}}(4.05)$ ε	$\overline{\mathcal{D}}(2.87)$ ε
1	0.88	0.78	0.73	0.62	26	0.53	0.52	0.54	0.54
2	0.77	0.61	0.62	0.61	27	0.53	0.54	0.54	0.54
3	0.67	0.53	0.60	0.59	28	0.52	0.52	0.54	0.54
4	0.59	0.65	0.59	0.59	29	0.52	0.53		0.54
5	0.53	0.60	0.58	0.58	30	0.51	0.53		0.53
6	0.49	0.59	0.58	0.58	31	0.51	0.53		0.53
7	0.48	0.54	0.57	0.57	32	0.54	0.53		0.53
8	0.60	0.58	0.57	0.57	33	0.53	0.53		0.53
9	0.59	0.54	0.56	0.56	34	0.53	0.53		0.53
10	0.57	0.56	0.56	0.56	35	0.52	0.52		0.53
11	0.56	0.53	0.56	0.56	36	0.52	0.53		0.53
12	0.54	0.55	0.55	0.55	37	0.52	0.52		0.53
13	0.52	0.53	0.55	0.55	38	0.51	0.53		0.53
14	0.51	0.55	0.55	0.55	39	0.51	0.52		0.53
15	0.50	0.53	0.55	0.55	40	0.53	0.53		
16	0.56	0.54	0.55	0.55	41	0.53	0.52		
17	0.55	0.55	0.55	0.55	42	0.52	0.53		
18	0.55	0.54	0.54	0.54	43	0.52	0.52		
19	0.54	0.55	0.54	0.54	44	0.52	0.53		
20	0.53	0.53	0.54	0.54	45	0.51	0.52		
21	0.52	0.55	0.54	0.54	46	0.51	0.52		
22	0.51	0.53	0.54	0.54	47	0.51	0.52		
23	0.51	0.54	0.54	0.54	48	0.53	0.52		
24	0.54	0.52	0.54	0.54	49	0.52	0.52		
25	0.54	0.54	0.54	0.54	50	0.52	0.52		

Fig. 2. Success probabilities ε of AbortedKeyGuess for Kyber and Falcon distributions.

5.1 Success Probability Larger Than $\frac{1}{2}$

We first study the success probabilities of our AbortedKeyGuess for distributions $\mathcal{B}(2), \mathcal{B}(3), \overline{\mathcal{D}}(2.87)$, and $\overline{\mathcal{D}}(4.05)$. For the binomial distributions we went up to key length $n = 50$, whereas we stopped earlier for the computationally heavy approximate discrete Gaussians. The success probabilities $\varepsilon = \Pr[\mathbf{X} \in \mathcal{C}_\chi^n]$ are provided in Fig. 2, the convergence towards $\frac{1}{2}$ is visualized in Fig. 3.

The binomial distribution $\mathcal{B}(2)$ has the most narrow support $\{-2, \ldots, 2\}$. This results in a large amount of keys (relative to $|A^n|$) that share the same probability. As a consequence, we see in Fig. 3 the largest oscillation bumps, reflecting the fact that at each upward jump keys from the next smaller probability level were taken into account. As one can see, this oscillation effect disappears when we broaden the support.

While Theorem 3.4 only guarantees the convergence towards $\frac{1}{2}$, our experiments show that, for our cryptographic distributions, the convergence is from above, thereby lower bounding our success probability by $\varepsilon \geq \frac{1}{2}$ (for all but very few exceptions in $\mathcal{B}(2)$).

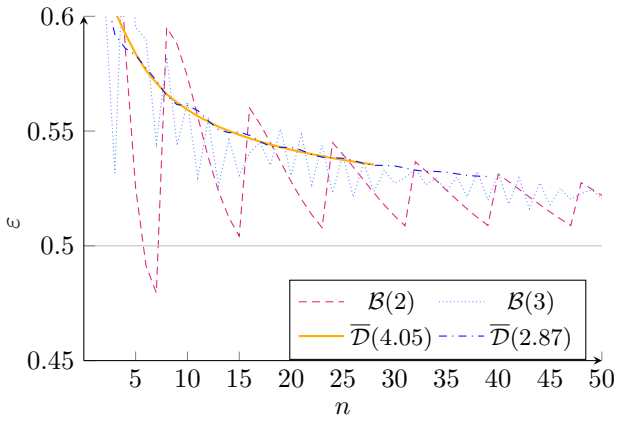

Fig. 3. Convergence of Success Probabilities ε of AbortedKeyGuess for Kyber and Falcon distributions.

5.2 AbortedKeyGuess Trials Tightly Match the Entropy Bound

Recall that our AbortedKeyGuess algorithm enumerates all keys from the core set $\mathcal{C}_\chi^n$, and therefore makes at most $|\mathcal{C}_\chi^n|$ key guesses. Our main theorem (Theorem 3.4) upper bounds $|\mathcal{C}_\chi^n| \leq 2^{\mathrm{H}(\chi)n}$. In this section, we study how tight this upper bound actually is.

To this end, in Fig. 6, we compare the bit complexity $b_{\mathsf{KG}} := \log(|\mathcal{C}_\chi^n|)$ of the maximum number of key guesses $|\mathcal{C}_\chi^n|$ in AbortedKeyGuess to $\mathrm{H}(\chi)n$. Their difference $\Delta = \mathrm{H}(\chi)n - b_{\mathsf{KG}}$ is visualized in Fig. 4. We see that Δ seems to grow slightly faster than $\frac{1}{2}\log n$, implying that the quotient $2^{\mathrm{H}(\chi)n}/|\mathcal{C}_\chi^n|$ grows at least as fast as $\sqrt{n}$, independently of the underlying distribution χ.

In Fig. 5, we express $2^{\mathrm{H}(\chi)n}/|\mathcal{C}_\chi^n| = n^c$ for some c. The polynomial exponent c seems to converge to $\frac{1}{2}$, which lets us conclude that the maximal amount of

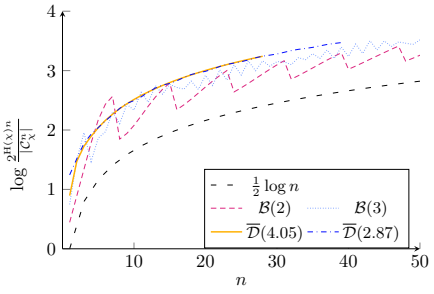

Fig. 4. Difference between $\mathrm{H}(\chi)n$ and bit complexity $\log(|\mathcal{C}_\chi^n|)$ of AbortedKeyGuess.

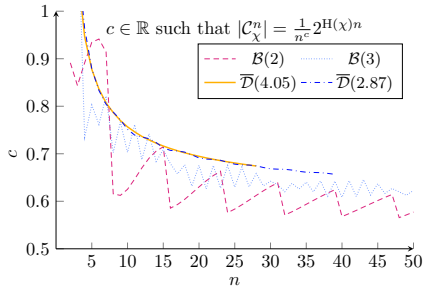

Fig. 5. Polynomial exponent $c \in \mathbb{R}$ such that $|\mathcal{C}_\chi^n| = \frac{1}{n^c}2^{\mathrm{H}(\chi)n}$.

	$\mathcal{B}(2)$			$\mathcal{B}(3)$			$\overline{\mathcal{D}}(4.05)$			$\overline{\mathcal{D}}(2.87)$		
n	$\mathrm{H}(\chi)n$	b_{KG}	Δ	$\mathrm{H}(\chi)n$	b_{KG}	Δ	$\mathrm{H}(\chi)n$	b_{KG}	Δ	$\mathrm{H}(\chi)n$	b_{KG}	Δ
1	2.0	1.6	0.4	2.3	1.6	0.7	4.1	3.2	0.9	3.6	2.3	1.2
2	4.1	3.2	0.9	4.7	3.2	1.5	8.1	6.7	1.5	7.1	5.6	1.5
3	6.1	4.8	1.3	7.0	5.0	2.0	12.2	10.5	1.7	10.7	8.9	1.8
4	8.1	6.3	1.8	9.3	7.9	1.5	16.3	14.4	1.9	14.3	12.3	1.9
5	10.2	8.0	2.2	11.7	9.8	1.9	20.3	18.3	2.0	17.8	15.8	2.0
6	12.2	9.7	2.4	14.0	12.0	2.0	24.4	22.2	2.2	21.4	19.2	2.2
7	14.2	11.6	2.6	16.3	14.0	2.3	28.5	26.2	2.3	25.0	22.7	2.3
8	16.2	14.4	1.8	18.7	16.6	2.1	32.5	30.2	2.4	28.5	26.2	2.4
9	18.3	16.3	1.9	21.0	18.6	2.4	36.6	34.1	2.4	32.1	29.6	2.4
10	20.3	18.2	2.1	23.3	21.0	2.3	40.7	38.1	2.5	35.7	33.2	2.5
11	22.3	20.1	2.2	25.7	23.0	2.6	44.7	42.1	2.6	39.2	36.7	2.6
12	24.4	22.0	2.4	28.0	25.5	2.5	48.8	46.1	2.6	42.8	40.2	2.6
13	26.4	23.8	2.6	30.3	27.6	2.7	52.9	50.2	2.7	46.4	43.6	2.7
14	28.4	25.7	2.7	32.7	30.1	2.6	56.9	54.2	2.8	49.9	47.2	2.8
15	30.5	27.7	2.8	35.0	32.2	2.8	61.0	58.2	2.8	53.5	50.7	2.8
16	32.5	30.1	2.3	37.3	34.6	2.7	65.0	62.2	2.8	57.0	54.2	2.8
17	34.5	32.1	2.4	39.7	37.0	2.7	69.1	66.2	2.9	60.6	57.7	2.9
18	36.6	34.0	2.5	42.0	39.2	2.8	73.2	70.3	2.9	64.2	61.2	2.9
19	38.6	36.0	2.6	44.3	41.6	2.7	77.2	74.3	3.0	67.7	64.8	3.0
20	40.6	37.9	2.7	46.7	43.7	2.9	81.3	78.3	3.0	71.3	68.3	3.0
21	42.6	39.8	2.8	49.0	46.3	2.8	85.4	82.3	3.0	74.9	71.9	3.0
22	44.7	41.8	2.9	51.3	48.3	3.0	89.4	86.4	3.1	78.4	75.4	3.1
23	46.7	43.7	3.0	53.7	50.8	2.8	93.5	90.4	3.1	82.0	78.9	3.1
24	48.7	46.1	2.6	56.0	52.9	3.1	97.6	94.4	3.1	85.6	82.4	3.1
25	50.8	48.1	2.7	58.3	55.4	2.9	101.6	98.5	3.2	89.1	86.0	3.1
26	52.8	50.0	2.8	60.7	57.5	3.2	105.7	102.5	3.2	92.7	89.5	3.2
27	54.8	52.0	2.9	63.0	60.0	3.0	109.8	106.6	3.2	96.3	93.1	3.2
28	56.9	53.9	2.9	65.3	62.2	3.2	113.8	110.6	3.2	99.8	96.6	3.2
29	58.9	55.9	3.0	67.7	64.6	3.1	117.9			103.4	100.1	3.3
30	60.9	57.8	3.1	70.0	66.8	3.2	122.0			107.0	103.7	3.3
31	62.9	59.8	3.2	72.3	69.2	3.1	126.0			110.5	107.2	3.3
32	65.0	62.1	2.9	74.7	71.6	3.1	130.1			114.1	110.8	3.3
33	67.0	64.1	2.9	77.0	73.8	3.2	134.2			117.7	114.3	3.4
34	69.0	66.1	3.0	79.3	76.1	3.2	138.2			121.2	117.9	3.4
35	71.1	68.0	3.0	81.7	78.4	3.3	142.3			124.8	121.4	3.4
36	73.1	70.0	3.1	84.0	80.8	3.2	146.4			128.4	124.9	3.4
37	75.1	72.0	3.2	86.3	83.0	3.3	150.4			131.9	128.5	3.4
38	77.2	73.9	3.2	88.7	85.5	3.2	154.5			135.5	132.0	3.5
39	79.2	75.9	3.3	91.0	87.6	3.4	158.6			139.1	135.6	3.5
40	81.2	78.2	3.0	93.3	90.1	3.2	162.6			142.6		
41	83.3	80.2	3.1	95.7	92.2	3.4	166.7			146.2		
42	85.3	82.2	3.1	98.0	94.7	3.3	170.7			149.8		
43	87.3	84.1	3.2	100.3	96.8	3.5	174.8			153.3		
44	89.3	86.1	3.2	102.7	99.3	3.3	178.9			156.9		
45	91.4	88.1	3.3	105.0	101.5	3.5	182.9			160.4		
46	93.4	90.0	3.4	107.3	103.9	3.4	187.0			164.0		
47	95.4	92.0	3.4	109.7	106.2	3.5	191.1			167.6		
48	97.5	94.3	3.2	112.0	108.5	3.5	195.1			171.1		
49	99.5	96.3	3.2	114.3	110.9	3.4	199.2			174.7		
50	101.5	98.3	3.3	116.7	113.2	3.5	203.3			178.3		

Fig. 6. Bit complexity $b_{\mathsf{KG}} := \log(|\mathcal{C}_\chi^n|)$ of the maximal amount of key guesses $|\mathcal{C}_\chi^n|$ in AbortedKeyGuess in comparison to $\mathrm{H}(\chi)n$, and their difference $\Delta := \mathrm{H}(\chi)n - b_{\mathsf{KG}}$.

key guesses $|\mathcal{C}_\chi^n|$ is tightly upper bounded by $2^{\mathrm{H}(\chi)n}$ (up to a small polynomial factor that tends to $\sqrt{n}$).

	$\mathcal{B}(2)$			$\mathcal{B}(3)$				$\mathcal{B}(2)$			$\mathcal{B}(3)$		
n	$b_{\mathbb{E}[\mathsf{KG}]}$	$b_{\mathbb{E}[\mathsf{AKG}]}$	Δ	$b_{\mathbb{E}[\mathsf{KG}]}$	$b_{\mathbb{E}[\mathsf{AKG}]}$	Δ	n	$b_{\mathbb{E}[\mathsf{KG}]}$	$b_{\mathbb{E}[\mathsf{AKG}]}$	Δ	$b_{\mathbb{E}[\mathsf{KG}]}$	$b_{\mathbb{E}[\mathsf{AKG}]}$	Δ
1	1.1	1.0	0.1	1.3	1.1	0.2	26	53.3	49.2	4.1	62.2	56.7	5.5
2	2.8	2.4	0.4	3.3	2.7	0.7	27	55.4	51.2	4.2	64.7	59.2	5.5
3	4.7	4.1	0.7	5.6	4.5	1.1	28	57.6	53.2	4.4	67.2	61.4	5.8
4	6.8	5.7	1.0	8.0	7.1	0.9	29	59.7	55.1	4.6	69.7	63.8	5.9
5	8.8	7.4	1.4	10.4	9.1	1.3	30	61.9	57.1	4.8	72.2	66.1	6.1
6	10.9	9.2	1.7	12.8	11.3	1.5	31	64.0	59.0	4.9	74.7	68.4	6.3
7	13.0	11.1	1.9	15.2	13.4	1.8	32	66.1	61.3	4.8	77.2	70.8	6.4
8	15.1	13.6	1.4	17.7	15.8	1.9	33	68.3	63.3	5.0	79.7	73.0	6.7
9	17.2	15.6	1.6	20.1	17.9	2.2	34	70.4	65.3	5.1	82.2	75.3	6.8
10	19.3	17.5	1.8	22.6	20.3	2.3	35	72.6	67.2	5.3	84.7	77.6	7.1
11	21.4	19.4	2.0	25.0	22.4	2.7	36	74.7	69.2	5.5	87.2	80.0	7.2
12	23.5	21.3	2.2	27.5	24.8	2.7	37	76.8	71.2	5.6	89.7	82.2	7.5
13	25.6	23.2	2.5	30.0	26.9	3.1	38	79.0	73.2	5.8	92.1	84.6	7.5
14	27.8	25.1	2.7	32.4	29.3	3.1	39	81.1	75.1	6.0	94.6	86.8	7.8
15	29.9	27.0	2.9	34.9	31.5	3.4	40	83.3	77.4	5.9	97.1	89.3	7.9
16	32.0	29.4	2.6	37.4	33.9	3.5	41	85.4	79.4	6.0	99.6	91.4	8.2
17	34.1	31.3	2.8	39.9	36.2	3.7	42	87.5	81.4	6.2	102.1	93.9	8.3
18	36.3	33.3	3.0	42.3	38.4	3.9	43	89.7	83.3	6.4	104.6	96.0	8.6
19	38.4	35.2	3.2	44.8	40.9	4.0	44	91.8	85.3	6.5	107.1	98.5	8.6
20	40.5	37.2	3.4	47.3	43.0	4.3	45	94.0	87.3	6.7	109.6	100.7	8.9
21	42.6	39.1	3.5	49.8	45.5	4.3	46	96.1	89.3	6.8	112.1	103.1	9.0
22	44.8	41.0	3.7	52.3	47.6	4.7	47	98.3	91.3	7.0	114.6	105.4	9.3
23	46.9	43.0	3.9	54.8	50.0	4.7	48	100.4	93.5	6.9	117.1	107.7	9.4
24	49.0	45.3	3.7	57.3	52.1	5.1	49	102.6	95.5	7.1	119.6	110.1	9.6
25	51.2	47.3	3.9	59.7	54.6	5.1	50	104.7	97.5	7.2	122.1	112.3	9.8

Fig. 7. Bit complexities $b_{\mathbb{E}[\mathsf{KG}]} := \log(\mathbb{E}[T_{\mathsf{KG}}])$ and $b_{\mathbb{E}[\mathsf{AKG}]} := \log(\mathbb{E}[T_{\mathsf{AKG}}])$ of expected amount of key trials of KeyGuess and AbortedKeyGuess, respectively. Δ denotes the difference in bit complexities.

5.3 The Benefit of AbortedKeyGuess over KeyGuess

Let the i-th key candidate $\mathbf{a}_i$ have success probability p_i. Then the expected number of trials in KeyGuess is

$$\mathbb{E}[T_{\mathsf{KG}}] = \sum_{i=1}^{|A^n|} p_i \cdot i.$$

In AbortedKeyGuess we only enumerate keys from the core set $\mathcal{C}_\chi^n$. In case that our key is not in the core set, our algorithm aborts after $|\mathcal{C}_\chi^n|$ key trials. Therefore, the expected number of trials of AbortedKeyGuess is

$$\mathbb{E}[T_{\mathsf{AKG}}] = \sum_{i=1}^{|\mathcal{C}_\chi^n|} p_i \cdot i + (1 - \Pr[\mathbf{X} \in \mathcal{C}_\chi^n]) \cdot |\mathcal{C}_\chi^n|.$$

In this section, we study the gain $\mathbb{E}[T_{\mathsf{KG}}]/\mathbb{E}[T_{\mathsf{AKG}}]$ achieved by our aborted guessing algorithm that comes at the mild cost of losing a factor of 2 in the success probability ε.

Since the computation of $\mathbb{E}[T_{\mathsf{AKG}}]$ requires the full enumeration of the compact dictionary $\mathcal{D}_\chi^n$ from Definition 3.5, we are unable to perform this computation for distributions with a large support like $\overline{\mathcal{D}}(2.87)$ and $\overline{\mathcal{D}}(4.05)$. Therefore,

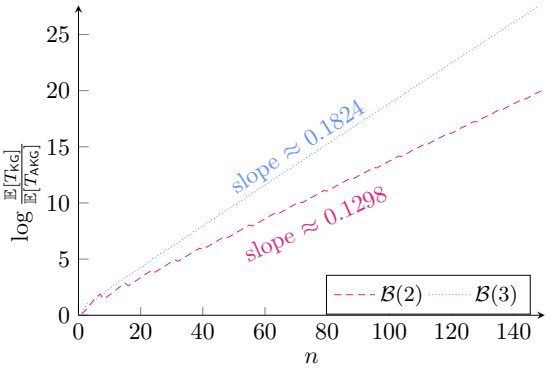

Fig. 8. Logarithm of our gain $\mathbb{E}[T_{\mathsf{KG}}]/\mathbb{E}[T_{\mathsf{AKG}}]$ as a function of n.

in this section we solely consider $\mathcal{B}(2)$ and $\mathcal{B}(3)$. The computations of their bit complexities for $\mathbb{E}[T_{\mathsf{KG}}]$ and $\mathbb{E}[T_{\mathsf{AKG}}]$ are depicted in Fig 7.

In Fig. 8, we plot the logarithm of our gain $\mathbb{E}[T_{\mathsf{KG}}]/\mathbb{E}[T_{\mathsf{AKG}}]$. For $\mathcal{B}(2)$ this logarithmic gain is $0.13n$, whereas for $\mathcal{B}(3)$ the logarithmic gain is $0.18n$. The logarithmic gain in turn implies that we save exponential factors of $2^{0.13n}$, respectively $2^{0.18n}$, for the expected amount of key trials when using AbortedKeyGuess rather than KeyGuess.

Our experiments are in line with Ducas and Pulles [DP23], who experimentally observed an exponential factor between $\mathbb{E}[T_{\mathsf{KG}}]$ and $2^{\mathrm{H}(\chi)n}$. Furthermore, Albrecht and Shen [AS22] provided an upper bound for $\mathbb{E}[T_{\mathsf{KG}}]$ for the discrete Gaussian distribution of the form $2^{\Theta(n)} \cdot 2^{\mathrm{H}(\chi)n}$.

5.4 QuantumKeyGuess compares well to Montanaro's algorithm

Our QuantumKeyGuess from Section 4 requires $2^{\mathrm{H}(\chi)n/2}$ many key guesses. Although our algorithm is a comparatively simple Grover-type application, we already showed in Theorem 4.6 for Montanaro's more involved algorithm a lower bound that matches our number of key trials by a polynomial factor.

This section is devoted to experimentally evaluate the limitations of the speedup that can be achieved by using Montanaro's algorithm. Our QuantumKeyGuess does not only have worst case complexity $2^{\mathrm{H}(\chi)n/2}$, but we also expect $\mathbb{E}[T_{\mathsf{QKG}}] = 2^{\mathrm{H}(\chi)n/2}$ many key trials. Montanaro's algorithm —applied to our core set $\mathcal{C}_\chi^n$ to enable fair comparison— instead achieves an amount of key trials of

$$\mathbb{E}[T_{\mathsf{Mon}}] = \sum_{i=1}^{|\mathcal{C}_\chi^n|} p_i \sqrt{i}.$$

on expectation.

The bit complexities of $\mathbb{E}[T_{\mathsf{QKG}}]$ and $\mathbb{E}[T_{\mathsf{Mon}}]$ are provided in Fig. 9. Their differences are visualized in Fig. 10. Independent of the distribution, on this

	$\mathcal{B}(2)$			$\mathcal{B}(3)$			$\overline{\mathcal{D}}(4.05)$			$\overline{\mathcal{D}}(2.87)$		
n	$b_{\mathbb{E}[\mathsf{QKG}]}$	$b_{\mathbb{E}[\mathsf{Mon}]}$	Δ	$b_{\mathbb{E}[\mathsf{QKG}]}$	$b_{\mathbb{E}[\mathsf{Mon}]}$	Δ	$b_{\mathbb{E}[\mathsf{QKG}]}$	$b_{\mathbb{E}[\mathsf{Mon}]}$	Δ	$b_{\mathbb{E}[\mathsf{QKG}]}$	$b_{\mathbb{E}[\mathsf{Mon}]}$	Δ
1	0.8	0.2	0.6	0.8	0.1	0.7	1.6	0.6	1.0	1.2	0.0	1.1
2	1.6	0.6	1.0	1.6	0.3	1.3	3.3	1.9	1.4	2.8	1.4	1.4
3	2.4	1.1	1.3	2.5	0.9	1.6	5.2	3.7	1.5	4.5	2.9	1.6
4	3.2	1.6	1.5	3.9	2.4	1.5	7.2	5.6	1.6	6.2	4.5	1.6
5	4.0	2.2	1.8	4.9	3.3	1.6	9.1	7.4	1.7	7.9	6.2	1.7
6	4.9	3.0	1.9	6.0	4.3	1.7	11.1	9.3	1.8	9.6	7.8	1.8
7	5.8	3.8	2.0	7.0	5.2	1.8	13.1	11.3	1.8	11.4	9.5	1.8
8	7.2	5.4	1.8	8.3	6.5	1.8	15.1	13.2	1.9	13.1	11.2	1.9
9	8.2	6.3	1.8	9.3	7.4	1.9	17.1	15.1	1.9	14.8	12.9	1.9
10	9.1	7.2	1.9	10.5	8.6	1.9	19.1	17.1	2.0	16.6	14.6	2.0
11	10.0	8.1	1.9	11.5	9.5	2.0	21.1	19.0	2.0	18.3	16.3	2.0
12	11.0	9.0	2.0	12.8	10.8	2.0	23.1	21.0	2.1	20.1	18.0	2.1
13	11.9	9.9	2.1	13.8	11.7	2.1	25.1	23.0	2.1	21.8	19.7	2.1
14	12.9	10.8	2.1	15.0	12.9	2.1	27.1	24.9	2.1	23.6	21.4	2.1
15	13.8	11.7	2.1	16.1	14.0	2.1	29.1	26.9	2.2	25.3	23.2	2.2
16	15.1	13.0	2.1	17.3	15.2	2.1	31.1	28.9	2.2	27.1	24.9	2.2
17	16.0	14.0	2.1	18.5	16.3	2.2	33.1	30.9	2.2	28.9	26.6	2.2
18	17.0	14.9	2.1	19.6	17.4	2.2	35.1	32.9	2.3	30.6	28.4	2.3
19	18.0	15.8	2.2	20.8	18.6	2.2	37.1	34.8	2.3	32.4	30.1	2.3
20	18.9	16.7	2.2	21.9	19.6	2.3	39.2	36.8	2.3	34.2	31.8	2.3
21	19.9	17.7	2.2	23.1	20.9	2.2	41.2	38.8	2.3	35.9	33.6	2.3
22	20.9	18.6	2.3	24.2	21.9	2.3	43.2	40.8	2.4	37.7	35.3	2.4
23	21.9	19.6	2.3	25.4	23.1	2.3	45.2	42.8	2.4	39.5	37.1	2.4
24	23.0	20.8	2.2	26.5	24.1	2.3	47.2	44.8	2.4	41.2	38.8	2.4
25	24.0	21.8	2.3	27.7	25.4	2.3	49.2	46.8	2.4	43.0	40.6	2.4
26	25.0	22.7	2.3	28.8	26.4	2.4	51.3	48.8	2.5	44.8	42.3	2.5
27	26.0	23.7	2.3	30.0	27.6	2.4	53.3	50.8	2.5	46.5	44.1	2.5
28	27.0	24.6	2.3	31.1	28.7	2.4	55.3	52.8	2.5	48.3	45.8	2.5
29	27.9	25.6	2.4	32.3	29.9	2.4				50.1	47.6	2.5
30	28.9	26.5	2.4	33.4	31.0	2.4				51.8	49.3	2.5
31	29.9	27.5	2.4	34.6	32.1	2.5				53.6	51.1	2.5
32	31.1	28.7	2.4	35.8	33.3	2.5				55.4	52.8	2.6
33	32.0	29.7	2.4	36.9	34.4	2.5				57.2	54.6	2.6
34	33.0	30.6	2.4	38.1	35.6	2.5				58.9	56.3	2.6
35	34.0	31.6	2.4	39.2	36.7	2.5				60.7	58.1	2.6
36	35.0	32.5	2.5	40.4	37.9	2.5				62.5	59.8	2.6
37	36.0	33.5	2.5	41.5	38.9	2.6				64.2	61.6	2.6
38	37.0	34.5	2.5	42.7	40.2	2.5				66.0	63.4	2.7
39	37.9	35.4	2.5	43.8	41.2	2.6				67.8	65.1	2.7
40	39.1	36.6	2.5	45.0	42.5	2.6						
41	40.1	37.6	2.5	46.1	43.5	2.6						
42	41.1	38.6	2.5	47.4	44.8	2.6						
43	42.1	39.5	2.5	48.4	45.8	2.6						
44	43.1	40.5	2.6	49.7	47.0	2.6						
45	44.0	41.5	2.6	50.8	48.1	2.7						
46	45.0	42.4	2.6	52.0	49.3	2.7						
47	46.0	43.4	2.6	53.1	50.4	2.7						
48	47.2	44.6	2.6	54.3	51.6	2.7						
49	48.1	45.6	2.6	55.4	52.8	2.7						
50	49.1	46.5	2.6	56.6	53.9	2.7						

Fig. 9. Bit complexities $b_{\mathbb{E}[T_{\mathsf{QKG}}]} := \log(\mathbb{E}[T_{\mathsf{QKG}}])$ and $b_{\mathbb{E}[\mathsf{Mon}]} := \log(\mathbb{E}[T_{\mathsf{Mon}}])$ of QuantumKeyGuess and Montanaro's algorithm, respectively.

logarithmic scale all differences in Fig. 10 tend to $\frac{1}{2}\log n$. This implies that the ratio $\mathbb{E}[T_{\mathsf{QKG}}]/\mathbb{E}[T_{\mathsf{Mon}}]$ is approximately $\sqrt{n}$.

Fig. 11 demonstrates that for large n the ratio $\mathbb{E}[T_{\mathsf{QKG}}]/\mathbb{E}[T_{\mathsf{Mon}}]$ becomes even a bit smaller than $\sqrt{n}$. As a conclusion, taking Montanaro's more involved algorithm instead of QuantumKeyGuess results only in a rather minor polynomial speedup of approximately $\sqrt{n}$ for our distributions of cryptographic interest.

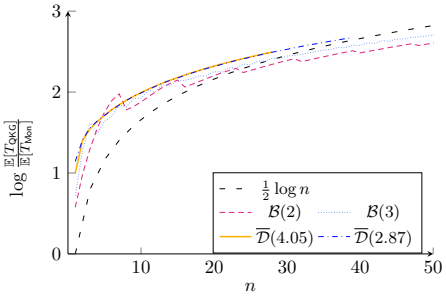

Fig. 10. Difference of bit complexities between QuantumKeyGuess and Montanaro's algorithm.

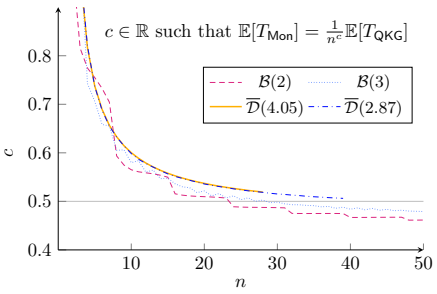

Fig. 11. Polynomial exponent $c \in \mathbb{R}$ such that $\mathbb{E}[T_{\mathsf{Mon}}] = \frac{1}{n^c}\mathbb{E}[T_{\mathsf{QKG}}]$.

References

AS22. Martin R Albrecht and Yixin Shen. Quantum augmented dual attack. *arXiv preprint arXiv:2205.13983*, 2022.

BDK+18. Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber: a cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

Ber41. Andrew C Berry. The accuracy of the gaussian approximation to the sum of independent variates. *Transactions of the american mathematical society*, 49(1):122–136, 1941.

BHMT02. Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

BM23. Alessandro Budroni and Erik Mårtensson. Improved estimation of key enumeration with applications to solving LWE. Cryptology ePrint Archive, Report 2023/139, 2023. `https://eprint.iacr.org/2023/139`.

CDH+21. C Chen, O Danba, J Hoffstein, A Hülsing, J Rijneveld, T Saito, JM Schanck, P Schwabe, W Whyte, K Xagawa, et al. Ntru: A submission to the nist post-quantum standardization effort, 2021.

DKL+18. Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018.

DP23. Léo Ducas and Ludo Pulles. Does the dual-sieve attack on learning with errors even work? *Cryptology ePrint Archive*, 2023.

Ess45. Carl-Gustav Esseen. Fourier analysis of distribution functions. a mathematical study of the laplace-gaussian law. 1945.

FHK+18. Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. In *Submission to NIST's post-quantum cryptography standardization process*, 2018.

Fou98. Electronic Frontier Foundation. Cracking des: Secrets of encryption research, wiretap politics and chip design, 1998.

GJ21. Qian Guo and Thomas Johansson. Faster dual lattice attacks for solving LWE with applications to CRYSTALS. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 33–62. Springer, Heidelberg, December 2021.

Gro96a. Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th Annual ACM Symposium on Theory of Computing*, pages 212–219. ACM Press, May 1996.

Gro96b. Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.

How07. Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169. Springer, Heidelberg, August 2007.

Høy00. Peter Høyer. Arbitrary phases in quantum amplitude amplification. *Physical Review A*, 62(5):052304, 2000.

HPS06. Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*, pages 267–288. Springer, 2006.

IDF22. MATZOV IDF. Report on the security of lwe:improved dual lattice attack, 2022. `https://zenodo.org/record/6412487#.ZCrT7-xBxqs`.

Mas94. James L. Massey. Guessing and entropy. In *Proceedings of 1994 IEEE International Symposium on Information Theory*, page 204. IEEE, 1994.

Mon11. Ashley Montanaro. Quantum search with advice. In *Theory of Quantum Computation, Communication, and Cryptography: 5th Conference, TQC 2010, Leeds, UK, April 13-15, 2010, Revised Selected Papers 5*, pages 77–93. Springer, 2011.

MU17. Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

Ngu21. Phong Q Nguyen. Boosting the hybrid attack on ntru: torus lsh, permuted hnf and boxed sphere. In *NIST Third PQC Standardization Conference*, 2021.

Sch22. Lars Schlieper. *Quantum cryptanalysis with minimal amount of qubits*. PhD thesis, Ruhr-Universität Bochum, 2022.

A Approximation of Falcon distribution

In Figure 12, we see our approximate distributions $\overline{\mathcal{D}}(4.05)$ and $\overline{\mathcal{D}}(2.87)$ for FALCON512 and FALCON1024. These approximations were taken by sampling 2^{19} times from $\mathcal{D}(4.05)$ and $\mathcal{D}(2.87)$. In order to make this approximate distribution symmetric, we set p_i as the average amount of samples of i and $-i$ and normalized it such that $\sum p_i = 1$.

i	$p_i \cdot 2^{20}$	
	$\overline{\mathcal{D}}(4.05)$	$\overline{\mathcal{D}}(2.87)$
0	104,098	145,946
1	100,876	137,664
2	91,062	113,981
3	77,952	84,393
4	63,141	55,367
5	47,872	31,864
6	34,803	16,198
7	23,297	7,422
8	14,642	2,930
9	8,718	1,055
10	4,887	307
11	2,613	109
12	1,323	22
13	608	3
14	260	
15	117	
16	43	
17	19	
18	3	
19	2	
20	1	

Fig. 12. Distribution approximations $\overline{\mathcal{D}}(4.05)$ and $\overline{\mathcal{D}}(2.87)$ for FALCON512 and FALCON1024, respectively. These approximate distributions were taken by sampling 2^{19} times from $\mathcal{D}(\sigma)$ for $\sigma = 1.17 \cdot \sqrt{\frac{12289}{2n}}$.