Entropy Suffices for Guessing Most Keys

Timo Glaser, Alexander May, and Julian Nowakowski

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

Abstract. Historically, most cryptosystems chose their keys uniformly at random. This is in contrast to modern (lattice-based) schemes, which typically sample their keys from more complex distributions $\mathcal{D}$, such as the discrete Gaussian or centered binomial distribution.

It is well-known that any key drawn from the uniform distribution $\mathcal{U}$ can be guessed using *at most* $2^{\mathrm{H}(\mathcal{U})}$ key guesses, where $\mathrm{H}(\mathcal{U})$ denotes the entropy of the uniform distribution. However, for keys drawn from general distributions $\mathcal{D}$ only a *lower bound* of $\Omega(2^{\mathrm{H}(\mathcal{D})})$ key guesses is known. In fact, Massey (1994) even ruled out that the number of key guesses can be upper bounded by a function of the entropy alone.

When analyzing the complexity of so-called hybrid lattice-attacks (which combine lattice reduction with key guessing) one therefore usually conservatively underestimates the complexity of key guessing by $2^{\mathrm{H}(\mathcal{D})}$. However, a *tight* complexity analysis is missing, and due to Massey's result considered impossible.

In this work, we bypass Massey's impossibility result by focusing on the typical cryptographic setting, where keys are drawn from n-fold product distributions $\mathcal{D} = \chi^n$.

It is well known that the optimal key guessing algorithm enumerates keys in χ^n in descending order of probability. In order to provide a refined analysis, we allow to abort enumeration after a certain amount of key guesses. As our main result, we prove that for *any discrete probability distribution* χ the key guessing algorithm that we abort after $2^{\mathrm{H}(\chi)n}$ keys has asymptotically success probability $\frac{1}{2}$, taken over the random key choice. The aborted algorithm allows for a quantum version with success probability $\frac{1}{2}$ within $2^{\mathrm{H}(\chi)n/2}$ key guesses. In other words, for any distribution χ, we achieve a Grover-type square root speedup.

Furthermore, we show that for the distributions used in Kyber and Falcon, the aborted algorithm outperforms the non-aborted algorithm by an exponential factor in the runtime. Hence, for a typical *multi-key scenario*, where a (large scale) adversary wants to attack as many keys with as few as possible resources, our results show that it greatly pays off to tackle only the likely keys.

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 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 a secret key is sufficiently hard to guess.

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.

(Sub-)Key Guessing in Hybrid Attacks. In the special case of lattice-based schemes there exist more sophisticated so-called hybrid attacks 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, also critically mentioned[1] in [Ber23]. 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[2] that this estimation lacks theoretical justification.

In this work, we provide the missing theoretical justification for the $2^{\mathrm{H}(\chi)n}$ estimate. [3] 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

[1] "One can easily argue that the analysis [of BLISS] is too optimistic for the attacker- there are, e.g., silent assumptions that searching cost is as small as an entropy-based lower bound."

[2] "In the analysis [of MATZOV] it seems to be assumed that enumerating possibilities in decreasing order of probability leads to guessing the correct [key] after an average of $2^{\mathrm{H}(X)}$ attempts [...]. There is no justification for this claim, and it appears to be false."

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

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

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 lossless 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.

Our result, classically. 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. In contrast to previous work, we drop the limiting restriction that a key guessing algorithm has to succeed with probability 1. Instead, we simply abort after $2^{\mathrm{H}(\chi)n}$ trials.

Using the Central Limit Theorem, we show that the success probability ε (taken over all keys) of aborted key guessing converges to $\frac{1}{2}$. Thus, with aborted key guessing and our complexity analysis, we achieve for the first time 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.

Multi-Key Scenario. In a nutshell, our result shows that for *all n-fold key distributions* χ^n, it does not pay off to guess unlikely keys[4]. Namely, in order to go significantly above success probability $\frac{1}{2}$, one has to pay an excessive, disproportionally large exponential overhead.

Of course, our result does not help an attacker that tries to enumerate a single unlikely key. However, in a typical *multi-key scenario* a (large scale) adversary acts in an economically optimal manner by enumerating in parallel as many keys with as few as possible resources. In such a scenario, the result of our analysis prevents to spend too much time on unlikely keys, and in turn still guarantees large success probability per attacked key.

Our result, quantumly. We also provide a quantum Grover-type version of the 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 $\frac{1}{2}$. Furthermore, we prove that also Montanaro's algorithm makes at least $\frac{1}{\mathrm{poly}(n)}2^{\mathrm{H}(\chi)n/2}$ queries in the setting of product distributions χ^n – thereby ruling out that with Montanaro's algorithm one can achieve more than polynomial speedups.

Our result, practically. 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) the aborted key guessing outperforms the usual key guessing (without aborts) by a run time factor exponential in n, i.e., we observe an exponential speedup with our abort strategy,
(3) 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 (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 the 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.

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

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 and $\mathbf{X} \leftarrow \chi^n$ to denote that the vector $\mathbf{X}$ has its n coordinates i.i.d. sampled from 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 *problity mass function* of χ is defined as

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

For ease of notation, we write

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

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 that*

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

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 two classical aborted key guessing algorithms, which have worst case runtime $\mathcal{O}(2^{\mathrm{H}(\chi)n})$ and asymptotic success probability at least $\frac{1}{2}$. We start by defining the *key guessing problem* in Section 3.1, and revisit the well-known key guessing algorithm, that enumerates keys in decreasing order of probability. Our first aborted key guessing algorithm builds on top of this algorithm, but aborts after trying the $2^{\mathrm{H}(\chi)n}$-th key. Our second algorithm instead simply samples $\mathcal{O}(2^{\mathrm{H}(\chi)n})$ random keys.

To analyze our algorithms' runtimes and success probabilities, we introduce in Section 3.2 the notion of a *core set*. Using such a core set, we then analyze our algorithms in Sections 3.3 and 3.4.

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 Appendix A, we demonstrate this for 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 efficient algorithm GetKey, that on input (n, P, A, j) outputs the j-th most likely key in A^n. Budroni and Mårtenson [BM23] give an efficient instantiation of such an algorithm, that (after one initial pre-computation phase running in time $\tilde{\mathcal{O}}(n^{|A|-1})$) outputs the j-th key in time $\mathcal{O}(|A| \cdot n)$.

For completeness, we give a description of their algorithm in Appendix B. We also implemented their GetKey as the basis for our experimental validations in Section 5.

For the remainder of this paper, we treat GetKey in a black-box fashion, and simply measure the complexity of all key guess algorithms by the number of oracle queries to τ.

Algorithm 1: KeyGuess

```
Input: Key guessing instance $(n, P, A, \tau)$,
       access to algorithm GetKey$(n, P, A, \cdot)$
Output: Key $\mathbf{X} \in A^n$ satisfying $\tau(\mathbf{X}) = 1$
$j \leftarrow 1$;
$\mathbf{a} \leftarrow$ GetKey$(n, P, A, j)$;
while $\tau(\mathbf{a}) \neq 1$ do
    $j \leftarrow j + 1$;
    $\mathbf{a} \leftarrow$ GetKey$(n, P, A, j)$
end
return $\mathbf{a}$
```

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

Let us denote by p_j the probability of the j-th most likely key in A^n. Then KeyGuess has expected number of key trials

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

Massey [Mas94] showed that for any distribution χ, Equation (1) is lower bounded by

$$\sum_{j=1}^{|A^n|} p_j \cdot j \geq \frac{1}{4} 2^{\mathrm{H}(\chi)n} + 1.$$

However, as we experimentally show in Section 5, for distributions of cryptographic interest, Massey's lower bound is rather loose: Our experiments indicate that KeyGuess's expected number of key trials grows exponentially faster than $2^{\mathrm{H}(\chi)n}$.

Aborted Key Guess Algorithm. Our novel algorithm AbortedKeyGuess (Algorithm 2) is a slight modification of Algorithm 1. The algorithm still enumerates keys in decreasing order of probability, but aborts after trying the $2^{\mathrm{H}(\chi)n}$-th key. Somewhat surprisingly, aborting only slightly lowers the success probability asymptotically to $\varepsilon \geq \frac{1}{2}$. In turn, it allows us to trivially bound the number of key guesses for any distribution χ^n by $2^{\mathrm{H}(\chi)n}$.

In comparison to Algorithm 1, we thereby save an exponential factor of key guesses by sacrificing only a factor of at most 2 for success probability.

Aborted Key Sampling Algorithm. A potential disadvantage of AbortedKeyGuess over KeyGuess is that AbortedKeyGuess succeeds to recover only the first $2^{\mathrm{H}(\chi)n}$ most likely keys. For all other key guess, AbortedKeyGuess clearly has success probability 0.

Algorithm 2: AbortedKeyGuess

```
Input: Key guessing instance $(n, P, A, \tau)$, entropy $\mathrm{H}(\chi)$,
       access to algorithm $\textsc{GetKey}(n, P, A, \cdot)$
Output: Key $\mathbf{a} \in A^n$ satisfying $\tau(\mathbf{a}) = 1$ or $\perp$ (abort).
$j \leftarrow 1$;
$\mathbf{a} \leftarrow \textsc{GetKey}(n, P, A, j)$;
while $\tau(\mathbf{a}) \neq 1$ and $j < 2^{\mathrm{H}(\chi)n}$ do
    $j \leftarrow j + 1$;
    $\mathbf{a} \leftarrow \textsc{GetKey}(n, P, A, j)$
end
if $\tau(\mathbf{a}) = 1$ then return $\mathbf{a}$;
else return $\perp$;
```

As discussed in the introduction, this is not much of an issue in a typical multi-key scenario, since AbortedKeyGuess still recovers half of the keys with high probability, while saving an exponential factor in the runtime over KeyGuess.

Still, in certain scenarios, it might be preferable to have an algorithm that has non-zero success probability for *any* given key. In that case, one may use our second algorithm AbortedKeySampling (Algorithm 3). Instead of enumerating keys by their likeliness, AbortedKeySampling simply iterates over roughly $2^{\mathrm{H}(\chi)n}$ many random keys. (This has the additional advantage of not requiring access to GetKey.)

Naturally, AbortedKeySampling has a slightly lower success probability than AbortedKeyGuess (since there is a non-negligible chance of the algorithm sampling too many unlikely keys). However, as we show in Section 3.4, by increasing number of sampled keys $2^{\mathrm{H}(\chi)n}$ by only a small constant factor $\delta > 1$, we can we can get arbitrarily close to a success probability of (at least) $\frac{1}{2}$.

Algorithm 3: AbortedKeySampling

```
Input: Key guessing instance $(n, P, A, \tau)$, entropy $\mathrm{H}(\chi)$, parameter $\delta \geq 1$.
Output: Key $\mathbf{X} \in A^n$ satisfying $\tau(\mathbf{X}) = 1$ or $\perp$ (abort).
$j \leftarrow 1$;
while $j < \delta \cdot 2^{\mathrm{H}(\chi)n}$ do
    $\mathbf{a} \leftarrow \chi^n$;
    if $\tau(\mathbf{a}) = 1$ then return $\mathbf{a}$ ;
    $j \leftarrow j + 1$;
end
return $\perp$;
```

3.2 The Core Set

Before we can prove our results for AbortedKeyGuess's and AbortedKeySampling's runtime and success probability, we need to introduce the following technical definition.

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*

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

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

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

The main novel observation that allows us to prove all our results is the following theorem, which shows that the core set $\mathcal{C}_\chi^n$ contains at most $2^{\operatorname{H}(\chi)n}$ of the keys, but (asymptotically) makes up half of the probability mass of χ^n.

Theorem 3.3. *Let χ be any (but the uniform) distribution with finite support. Then it holds that*

$$|\mathcal{C}_\chi^n| \leq 2^{\operatorname{H}(\chi)n}.$$

Furthermore, for $\mathbf{X} \leftarrow \chi^n$, we have

$$\Pr[\mathbf{X} \in \mathcal{C}_\chi^n] = \frac{1}{2} \pm \mathcal{O}\left(\frac{1}{\sqrt{n}}\right).$$

Proof. Let us first show that $|\mathcal{C}_\chi^n| \leq 2^{\operatorname{H}(\chi)n}$. We denote by A the support of χ, and by $P : A \to (0,1]$, $P_n : A^n \to (0,1]$ the probability mass functions of χ and χ^n, respectively. By definition of $\mathcal{C}_\chi^n$, it holds that

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

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

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

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

Let $Y_i := -\log P(X_i)$. (Note that Y_i is well-defined, since $P > 0$.) We set $\overline{Y_n} := \frac{1}{n} \sum_{i=1}^n Y_i$, and rewrite the above probability as

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

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,

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

which proves our theorem. □

Assumptions in Theorem 3.3. For ease of notation, we require in Theorem 3.3 that the distribution χ has finite support. However, we like to point out that the theorem applies more generally to all distributions, for which the conditions (1) to (3) from the proof above hold. In particular, Theorem 3.3 also applies to discrete Gaussian distributions.

The assumption that χ is not the uniform distribution, on the other hand, is crucial. (If χ was the uniform distribution, then the Y_i's in the proof would be constant and consequently $\overline{Y_n}$ would not converge to a Gaussian distribution.) Nevertheless, for the uniform distribution, we can prove an even stronger statement.

Theorem 3.4. *Let χ be the uniform distribution on some finite set A. Then for every $n \in \mathbb{N}$ it holds that*

$$\mathcal{C}_\chi^n = A^n, \quad \textit{and} \quad |\mathcal{C}_\chi^n| = |A|^n = 2^{\mathrm{H}(\chi)n}.$$

In particular, for $\mathbf{X} \leftarrow \chi^n$, we have

$$\Pr[\mathbf{X} \in \mathcal{C}_\chi^n] = 1.$$

Proof. The theorem immediately follows from the fact that uniform distribution has entropy $\mathrm{H}(\chi) = \log(|A|)$: We have $|A|^n = 2^{\mathrm{H}(\chi)n}$, and therefore every $\mathbf{a} \in A^n$ has probability $|A|^{-n} = 2^{-\mathrm{H}(\chi)n}$ (showing that $\mathbf{a} \in \mathcal{C}_\chi^n$). □

3.3 Analysis of AbortedKeyGuess

Using the results of Section 3.2, we now can analyse AbortedKeyGuess's runtime and success probability.

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

Theorem 3.5. *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 success probability 1 making at most $2^{\mathrm{H}(\chi)n}$ key trials.*

Proof. Theorem 3.5 follows immediately from Theorem 3.4. □

For all other distributions χ (different from the uniform distribution), we have the following theorem.

Theorem 3.6 (Main Theorem). *Let χ be any (but the uniform) distribution with probability mass function $P : A \to (0, 1]$. Then* AbortedKeyGuess *solves a random key guessing instance (n, P, A, τ), with success probability (taken over the random key choice)*

$$p_{\mathsf{AKG}} \geq \frac{1}{2} \pm \mathcal{O}(\frac{1}{\sqrt{n}}),$$

making at most $2^{\mathrm{H}(\chi)n}$ key trials.

Proof. By definition, AbortedKeyGuess's number of key trials is at most $2^{\mathrm{H}(\chi)n}$.

Let $\mathbf{X} \leftarrow \chi^n$, and let $\mathcal{S} \subseteq A^n$ denote the set of the $2^{\mathrm{H}(\chi)n}$ most likely keys in A^n, i.e., the set of keys over which AbortedKeyGuess itereates.

From the definition of $\mathcal{C}_\chi^n$ (Definition 3.2) and Theorem 3.3 it follows that $\mathcal{S} \supseteq \mathcal{C}_\chi^n$, and thus

$$p_{\mathsf{AKG}} = \Pr[\mathbf{X} \in \mathcal{S}] \geq \Pr[\mathbf{X} \in \mathcal{C}_\chi^n].$$

By Theorem 3.3, a random key $\mathbf{X}$ lies in the core set $\mathcal{C}_\chi^n$ with probability $\frac{1}{2} \pm \mathcal{O}(\frac{1}{\sqrt{n}})$, proving our main theorem. □

3.4 Analysis of AbortedKeySampling

It remains to prove the bound on AbortedKeySampling's success probability.

Theorem 3.7. *Let χ be any distribution with finite support A and probability mass function $P : A \to (0, 1]$. For every constant $\delta \geq 1$,* AbortedKeySampling *solves any key guessing instance (n, P, A, τ), with success probability (taken over the random key choice, and the internal randomness of* AbortedKeySampling*)*

$$p_{\mathsf{AKS}} \geq \frac{1}{2} - \frac{1}{2\delta} \pm \mathcal{O}\left(\frac{1}{\sqrt{n}}\right),$$

making at most $\delta \cdot 2^{\mathrm{H}(\chi)n}$ key trials.

Proof. By definition, AbortedKeySampling's number of key trials is at most $\delta \cdot 2^{\mathrm{H}(\chi)n}$.

Let $\mathbf{X} \in A^n$ be the unique key with $\tau(\mathbf{X}) = 1$, and let $P_n : A^n \to (0, 1]$ be the probability mass function of χ^n. Suppose we draw random variables

$\mathbf{Y}_1, \mathbf{Y}_2, \ldots \leftarrow \chi^n$, until we obtain $\mathbf{Y}_N$ with $\mathbf{Y}_N = \mathbf{X}$. Then N is a geometric random random variable with expected value $\mathbb{E}[N] = \frac{1}{P_n(\mathbf{X})}$, and it holds that

$$p_{\mathsf{AKS}} = \Pr\left[N < \delta \cdot 2^{\mathrm{H}(\chi)n}\right].$$

Conditioning on the event $\{\mathbf{X} \in \mathcal{C}_\chi^n\}$ and using Theorem 3.3, we obtain the following lower bound

$$\begin{aligned} p_{\mathsf{AKS}} &\geq \Pr\left[N < \delta \cdot 2^{\mathrm{H}(\chi)n} \mid \mathbf{X} \in \mathcal{C}_\chi^n\right] \cdot \Pr\left[\mathbf{X} \in \mathcal{C}_\chi^n\right] & (2)\\ &= \Pr\left[N < \delta \cdot 2^{\mathrm{H}(\chi)n} \mid \mathbf{X} \in \mathcal{C}_\chi^n\right] \cdot \left(\frac{1}{2} \pm \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)\right). & (3) \end{aligned}$$

By definition of $\mathcal{C}_\chi^n$, we have for $\mathbf{X} \in \mathcal{C}_\chi^n$ that $2^{\mathrm{H}(\chi)n} \geq \frac{1}{P_n(\mathbf{X})}$, and thus

$$\Pr\left[N < \delta \cdot 2^{\mathrm{H}(\chi)n} \mid \mathbf{X} \in \mathcal{C}_\chi^n\right] \geq \Pr\left[N < \delta \cdot \frac{1}{P(a)} \mid \mathbf{X} \in \mathcal{C}_\chi^n\right].$$

Together with $\frac{1}{P_n(\mathbf{X})} = \mathbb{E}[N]$ and Markov's inequality, this yields

$$\Pr\left[N < \delta \cdot 2^{\mathrm{H}(\chi)n} \mid \mathbf{X} \in \mathcal{C}_\chi^n\right] \geq 1 - \frac{1}{\delta}. \qquad (4)$$

Plugging Equation (4) into Equation (3), we obtain

$$p_{\mathsf{AKS}} \geq \left(1 - \frac{1}{\delta}\right) \cdot \left(\frac{1}{2} \pm \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)\right) = \frac{1}{2} - \frac{1}{2\delta} \pm \mathcal{O}\left(\frac{1}{\sqrt{n}}\right),$$

as required. □

The value of $\boldsymbol{\delta}$. Since we use rather crude lower bounds in the proof of Theorem 3.7 (in particular in Equations (2) and (4)), we expect the actual success probability of AbortedKeySampling to be significantly higher than $\frac{1}{2} - \frac{1}{2\delta}$. Indeed, as we experimentally show in Section 5, even for $\delta = 1$, we already have a non-zero success probability.

However, the point of Theorem 3.7 is not to prove a *tight* lower bound on p_{AKS}. Instead, we show that enumerating only $\mathcal{O}(2^{\mathrm{H}(\chi)n})$ many keys already yields a *constant* success probability.

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 at least $\frac{1}{2}$.

After that we compare our quantum key guessing algorithm with Montanaro's *optimal* algorithm [Mon11], and show that Montanaro's does not substantially outperform ours: The improvement of Montanaro's algorithm over ours is *at most* a small polynomial factor.

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

Recall that our classical algorithm AbortedKeyGuess (Algorithm 2) from Section 3 simply tries the $2^{\mathrm{H}(\chi)n}$ most likely keys. A natural quantum version of our algorithm is to run Grover's algorithm on a superposition of the $2^{\mathrm{H}(\chi)n}$ most likely keys, as depicted in Algorithm 4.

Algorithm 4: QuantumKeyGuess

```
Input: Key guessing instance $(n, P, A, \tau)$, entropy $\mathrm{H}(\chi)$,
       access to quantum algorithm QuantumGetKey$(n, P, A, \cdot)$
Output: Key $\mathbf{X} \in A^n$ satisfying $\tau(\mathbf{X}) = 1$ or $\perp$.
1 $t \leftarrow 2^{\lceil \mathrm{H}(\chi)n \rceil}$
2 $|\Psi\rangle \leftarrow 1/\sqrt{t} \cdot \sum_{i=1}^{t} |i\rangle$;
3 $|\Psi\rangle \leftarrow$ QuantumGetKey$(n, P, A, |\Psi\rangle)$;
4 Run Grover's algorithm on $|\Psi\rangle$ with oracle access to $\tau$ and return the result.
```

Access to QuantumGetKey**.** As discussed in Section 3, our classical algorithm AbortedKeyGuess requires access to an algorithm GetKey, that on input (n, P, A, i), with $1 \leq i \leq |A^n|$, outputs the i-th most likely key. Similarly, QuantumKeyGuess requires access to a quantum algorithm QuantumGetKey, that on input $(n, P, A, |\Psi\rangle)$, where $|\Psi\rangle$ is a superposition over some subset $\mathcal{S} \subseteq \{1, 2, \ldots, |A^n|\}$, outputs a superposition over the keys indexed by $\mathcal{S}$.

Such an algorithm can be instantiated efficiently by simply turning the original GetKey into a quantum algorithm.

Analysis. Since the superposition in Step 2 of QuantumKeyGuess can efficiently be instantiated using $\lceil \mathrm{H}(\chi)n \rceil$ Hadamard gates, the complexity of QuantumKeyGuess is dominated by Step 4 of the algorithm. By Lemma 2.3, Step 4 requires

$$\left\lceil \frac{\pi}{4} 2^{\lceil \mathrm{H}(\chi)n \rceil / 2} \right\rceil + 1 = \Theta(2^{\mathrm{H}(\chi)n/2})$$

oracle queries to τ. Together with Theorem 3.3, 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 our main theorem (Theorem 3.6).

Theorem 4.1. *Let χ be any (but the uniform) distribution with probability mass function $P : A \to (0, 1]$. Then* AbortedKeyGuess *solves a random key guessing instance (n, P, A, τ), with success probability (taken over the random key choice)*

$$p_{\mathsf{QKG}} \geq \frac{1}{2} \pm \mathcal{O}(\frac{1}{\sqrt{n}}),$$

making $\Theta(2^{\mathrm{H}(\chi)n/2})$ key trials.

4.2 Comparison with Montanaro

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 4.2 (Proposition 2.4 in [Mon11]). *Let (n, P, A, τ) be a key guessing instance 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 4.2 (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 4.3 shows, when n is not fixed to 1, then the lower bound from Theorem 4.2 is at most a polynomial factor better than the Grover bound of $2^{\mathrm{H}(\chi)n/2}$. Hence, for the typical cryptographic setting of product distributions χ^n, Montanaros algorithm does not substantially outperform our simple algorithm QuantumKeyGuess from Section 4.1.

We point out that one may view our Theorem 4.3 as a quantum variant of Massey's [Mas94] lower bound for the classical complexity.

Theorem 4.3. *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 finite 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

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

permutations of $\mathbf{a}$.

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

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

Hence, the sum is lower bounded by

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

proving the theorem. □

5 AbortedKeyGuess and AbortedKeySampling Applied to Kyber and Falcon

In this Section, we apply both algorithms from Section 3 to the distributions χ chosen in Kyber and Falcon. Kyber [BDK+18] utilizes 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*

$$P(a) = \frac{\binom{2\eta}{\eta+a}}{2^{2\eta}}.$$

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

Kyber512 samples its keys from $\mathcal{B}(3)^{512}$, whereas Kyber768 and Kyber1024 sample 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*

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

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 sampled from $\mathcal{D}(4.05)^{512}$ and Falcon1024 keys being sampled from $\mathcal{D}(2.87)^{1024}$.

For our algorithms to work, we require the underlying distribution to be of finite support - a property that $\mathcal{D}(\sigma)$ clearly lacks. To circumvent this problem, we use a distribution $\overline{\mathcal{D}}(\sigma)$ instead where we cut off the tails of $\mathcal{D}(\sigma)$, i.e. we condition $\mathcal{D}(\sigma)$ on the integer interval $\{-z, \ldots, z\}$ for some $z \in \mathbb{N}$. For either choice of $\sigma \in \{4.05, 2.87\}$ we use a value for z such that the probability distribution function of $\overline{\mathcal{D}}(\sigma)$, denoted with $\overline{P}(a)$, is approximately equal to the distribution function $P(a)$ of $\mathcal{D}(\sigma)$. Concretely, we choose z such that $\Pr[\mathbf{X} \in \{-z, \ldots, z\}^n] \geq (1 - 2^{-295})^n$ for any $\mathbf{X} \leftarrow \mathcal{D}(\sigma)^n$ and

$$P(a) \leq \overline{P}(a) \leq \frac{P(a)}{1 - 2^{-295}} \qquad \text{for } a \in \{-z, \ldots, z\}.$$

For an in-depth analysis, we refer to Appendix A.

5.1 Success probabilities of AbortedKeyGuess and AbortedKeySampling

We first study the success probabilities of our AbortedKeyGuess and AbortedKeySampling for distributions $\mathcal{B}(2), \mathcal{B}(3), \overline{\mathcal{D}}(2.87)$, and $\overline{\mathcal{D}}(4.05)$. We provide our numbers in Figure 1 and Figure 4 and their visualizations in the graphs in Figure 2 and Figure 3.

For AbortedKeyGuess, with the binomial distributions and the more shallow $\overline{\mathcal{D}}(2.87)$, we went up to key length $n = 50$, whereas we stopped at $n = 35$ for the computationally heavy $\overline{\mathcal{D}}(4.05)$. We provide the numbers in Figure 2, where it becomes easily apparent that the success probability is $\geq \frac{1}{2}$ for our cryptographic distributions, while Theorem 3.6 only guarantees that the success probability of AbortedKeyGuess converges towards $\geq \frac{1}{2}$. Based on Figure 2,

	$\mathcal{B}(2)$	$\mathcal{B}(3)$	$\overline{\mathcal{D}}(4.05)$	$\overline{\mathcal{D}}(2.87)$
n	ε	ε	ε	ε
1	0.69	0.87	0.63	0.70
2	0.68	0.59	0.67	0.67
3	0.67	0.65	0.65	0.66
4	0.63	0.66	0.65	0.65
5	0.62	0.65	0.65	0.65
6	0.63	0.63	0.64	0.64
7	0.63	0.63		0.64
8	0.62	0.63		0.63
9	0.62	0.63		
10	0.62	0.62		
11	0.61	0.62		
12	0.61	0.62		
13	0.61			
14	0.61			

Fig. 1. Relative amount of successfully recovered keys with AbortedKeySampling for Kyber and Falcon distributions. Key sample size is $100,000$ per n.

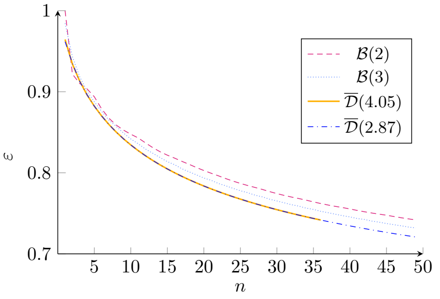

Fig. 2. Convergence of Success Probabilities ε of AbortedKeyGuess for Kyber and Falcon distributions.

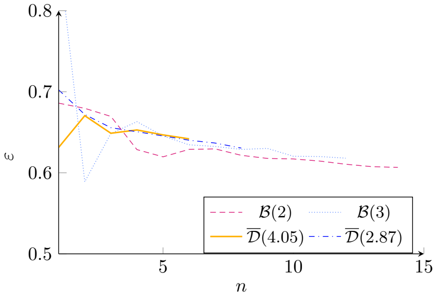

Fig. 3. Relative amount of successfully recovered Kyber and Falcon-keys using AbortedKeySampling.

we believe that the success probability is always larger than $\frac{1}{2}$ for all values of n.

The success probability of AbortedKeySampling is hard to determine as it requires the repeated calculation of values of the form $(1-P(\mathbf{a}))^{2^{\mathrm{H}(\chi)n}}$. Instead, we estimate the actual success probability by running the algorithm for 100,000 randomly sampled keys and returning the relative amount of keys recovered with AbortedKeySampling. These estimated success probabilities for $\delta = 1$

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

Fig. 4. Success probabilities ε of AbortedKeyGuess for Kyber and Falcon distributions.

in small dimensions n are provided in Figure 3. The graph clearly indicates that the presented lower bound from Theorem 3.7 is not tight, but only provides a constant lower bound for the success probability.

	$\mathcal{B}(2)$			$\mathcal{B}(3)$		
n	b_{KG}	b_{AKG}	Δ	b_{KG}	b_{AKG}	Δ
1	1.1	1.1	0.0	1.3	1.3	0.0
2	2.8	2.7	0.1	3.3	3.3	0.1
3	4.7	4.6	0.1	5.6	5.5	0.1
4	6.8	6.6	0.1	8.0	7.7	0.2
5	8.8	8.6	0.2	10.4	10.1	0.3
6	10.9	10.6	0.3	12.8	12.4	0.4
7	13.0	12.6	0.3	15.2	14.7	0.5
8	15.1	14.7	0.4	17.7	17.0	0.6
9	17.2	16.7	0.5	20.1	19.4	0.7
10	19.3	18.7	0.6	22.6	21.7	0.9
11	21.4	20.8	0.6	25.0	24.0	1.0
12	23.5	22.8	0.7	27.5	26.4	1.1
13	25.6	24.8	0.8	30.0	28.7	1.3
14	27.8	26.9	0.9	32.4	31.1	1.4
15	29.9	28.9	1.0	34.9	33.4	1.5
16	32.0	30.9	1.1	37.4	35.7	1.7
17	34.1	32.9	1.2	39.9	38.1	1.8
18	36.3	35.0	1.3	42.3	40.4	1.9
19	38.4	37.0	1.4	44.8	42.7	2.1
20	40.5	39.0	1.5	47.3	45.1	2.2
21	42.6	41.1	1.6	49.8	47.4	2.4
22	44.8	43.1	1.7	52.3	49.8	2.5
23	46.9	45.1	1.8	54.8	52.1	2.7
24	49.0	47.2	1.9	57.3	54.4	2.8
25	51.2	49.2	2.0	59.7	56.8	3.0
26	53.3	51.2	2.1	62.2	59.1	3.1
27	55.4	53.3	2.2	64.7	61.4	3.3
28	57.6	55.3	2.3	67.2	63.8	3.4
29	59.7	57.3	2.4	69.7	66.1	3.6
30	61.9	59.4	2.5	72.2	68.4	3.7
31	64.0	61.4	2.6	74.7	70.8	3.9
32	66.1	63.4	2.7	77.2	73.1	4.1
33	68.3	65.5	2.8	79.7	75.5	4.2
34	70.4	67.5	2.9	82.2	77.8	4.4
35	72.6	69.5	3.0	84.7	80.1	4.5
36	74.7	71.6	3.1	87.2	82.5	4.7
37	76.8	73.6	3.2	89.7	84.8	4.9
38	79.0	75.6	3.3	92.1	87.1	5.0
39	81.1	77.7	3.4	94.6	89.5	5.2
40	83.3	79.7	3.5	97.1	91.8	5.3
41	85.4	81.7	3.7	99.6	94.1	5.5
42	87.5	83.8	3.8	102.1	96.5	5.7
43	89.7	85.8	3.9	104.6	98.8	5.8
44	91.8	87.8	4.0	107.1	101.2	6.0
45	94.0	89.9	4.1	109.6	103.5	6.1
46	96.1	91.9	4.2	112.1	105.8	6.3
47	98.3	93.9	4.3	114.6	108.2	6.5
48	100.4	96.0	4.4	117.1	110.5	6.6
49	102.6	98.0	4.5	119.6	112.8	6.8
50	104.7	100.0	4.7	122.1	115.2	7.0

Fig. 5. Bit complexities $b_{\mathsf{KG}} := \log(\mathbb{E}[T_{\mathsf{KG}}])$ and $b_{\mathsf{AKG}} := \log(\mathbb{E}[T_{\mathsf{AKG}}])$ of expected amount of key trials of KeyGuess and AbortedKeyGuess, respectively. Δ denotes the difference in bit complexities.

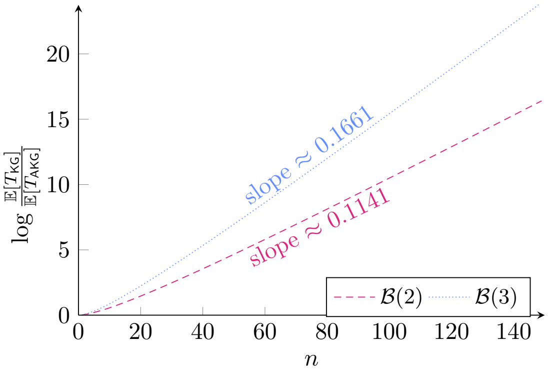

Fig. 6. Logarithm of our gain $\mathbb{E}[T_{\mathsf{KG}}]/\mathbb{E}[T_{\mathsf{AKG}}]$ as a function of n.

5.2 The Benefit of AbortedKeyGuess over KeyGuess

Let $\mathbf{a}_i$ be the i-th most likely key with sampling probability $p_i := P(\mathbf{a}_i)$. The expected number of trials in KeyGuess then is

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

In AbortedKeyGuess, we only enumerate the first $2^{\mathrm{H}(\chi)n}$ most likely keys. If the key that we are looking for is not among the first $2^{\mathrm{H}(\chi)n}$ most likely keys, AbortedKeyGuess aborts after $2^{\mathrm{H}(\chi)n}$ key trials. Consequently, the expected number of trials of AbortedKeyGuess is

$$\mathbb{E}[T_{\mathsf{AKG}}] = \sum_{i=1}^{2^{\mathrm{H}(\chi)n}} p_i \cdot i + \left(1 - \sum_{i=1}^{2^{\mathrm{H}(\chi)n}} p_i\right) \cdot 2^{\mathrm{H}(\chi)n}.$$

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 us to calculate $P(\mathbf{a})$ for every $\mathbf{a} \in A^n$ (up to permutation), 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, 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 5.

In Fig. 6, 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 $\approx 0.11n$, whereas for $\mathcal{B}(3)$ the logarithmic gain is $\approx 0.16n$. The logarithmic gain in turn implies that we save exponential factors of $2^{0.11n}$, respectively $2^{0.16n}$, for the expected amount of key trials when using AbortedKeyGuess rather than KeyGuess.

Our experiments are in line with Ducas and Pulles [DP23], who 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 $2^{\Theta(n)} \cdot 2^{\mathrm{H}(\chi)n}$.

	$\mathcal{B}(2)$			$\mathcal{B}(3)$			$\overline{\mathcal{D}}(4.05)$			$\overline{\mathcal{D}}(2.87)$		
n	b_{QKG}	b_{Mon}	Δ	b_{QKG}	b_{Mon}	Δ	b_{QKG}	b_{Mon}	Δ	b_{QKG}	b_{Mon}	Δ
1	1.6	0.5	1.1	1.6	0.6	1.0	2.3	1.2	1.1	2.0	1.0	1.0
2	2.3	1.1	1.3	2.3	1.4	1.0	3.9	3.0	0.9	3.5	2.5	1.0
3	3.0	1.9	1.1	3.3	2.3	1.0	5.8	4.8	0.9	5.0	4.1	1.0
4	3.9	2.8	1.1	4.4	3.4	1.0	7.8	6.8	1.0	6.8	5.8	1.1
5	4.8	3.8	1.1	5.5	4.4	1.1	9.8	8.7	1.1	8.6	7.4	1.1
6	5.8	4.7	1.1	6.7	5.5	1.2	11.9	10.6	1.2	10.4	9.1	1.2
7	6.8	5.6	1.2	7.8	6.6	1.2	13.9	12.6	1.3	12.1	10.8	1.3
8	7.8	6.6	1.2	9.0	7.7	1.3	15.9	14.6	1.4	13.9	12.6	1.4
9	8.8	7.6	1.2	10.2	8.8	1.3	17.9	16.5	1.4	15.7	14.3	1.4
10	9.8	8.5	1.3	11.3	10.0	1.4	20.0	18.5	1.5	17.5	16.0	1.5
11	10.8	9.5	1.3	12.5	11.1	1.4	22.0	20.5	1.5	19.3	17.7	1.5
12	11.8	10.5	1.3	13.7	12.2	1.5	24.0	22.5	1.6	21.0	19.5	1.6
13	12.9	11.5	1.4	14.8	13.3	1.5	26.1	24.5	1.6	22.8	21.2	1.6
14	13.9	12.4	1.4	16.0	14.5	1.5	28.1	26.5	1.6	24.6	23.0	1.6
15	14.9	13.4	1.5	17.2	15.6	1.6	30.1	28.5	1.7	26.4	24.7	1.7
16	15.9	14.4	1.5	18.3	16.7	1.6	32.2	30.5	1.7	28.2	26.5	1.7
17	16.9	15.4	1.5	19.5	17.9	1.6	34.2	32.5	1.8	30.0	28.2	1.8
18	17.9	16.4	1.5	20.7	19.0	1.7	36.2	34.5	1.8	31.7	30.0	1.8
19	18.9	17.4	1.6	21.8	20.1	1.7	38.3	36.5	1.8	33.5	31.7	1.8
20	20.0	18.4	1.6	23.0	21.3	1.7	40.3	38.5	1.8	35.3	33.5	1.8
21	21.0	19.4	1.6	24.2	22.4	1.7	42.3	40.5	1.9	37.1	35.2	1.9
22	22.0	20.3	1.6	25.3	23.5	1.8	44.4			38.9	37.0	1.9
23	23.0	21.3	1.7	26.5	24.7	1.8	46.4			40.7	38.7	1.9
24	24.0	22.3	1.7	27.7	25.8	1.8	48.4			42.4	40.5	2.0
25	25.0	23.3	1.7	28.8	27.0	1.8	50.5			44.2	42.2	2.0
26	26.1	24.3	1.7	30.0	28.1	1.9	52.5			46.0	44.0	2.0
27	27.1	25.3	1.8	31.2	29.3	1.9	54.5			47.8	45.8	2.0
28	28.1	26.3	1.8	32.3	30.4	1.9	56.6			49.6	47.5	2.0
29	29.1	27.3	1.8	33.5	31.6	1.9	58.6			51.4	49.3	2.1
30	30.1	28.3	1.8	34.7	32.7	1.9	60.6			53.1	51.1	2.1
31	31.1	29.3	1.8	35.8	33.9	2.0	62.7			54.9	52.8	2.1
32	32.1	30.3	1.8	37.0	35.0	2.0	64.7			56.7		
33	33.2	31.3	1.9	38.2	36.2	2.0	66.7			58.5		
34	34.2	32.3	1.9	39.3	37.3	2.0	68.8			60.3		
35	35.2	33.3	1.9	40.5	38.5	2.0	70.8			62.1		
36	36.2	34.3	1.9	41.7	39.6	2.1	72.8			63.8		
37	37.2	35.3	1.9	42.8	40.8	2.1	74.9			65.6		
38	38.2	36.3	1.9	44.0	41.9	2.1	76.9			67.4		
39	39.2	37.3	2.0	45.2	43.1	2.1	78.9			69.2		
40	40.3	38.3	2.0	46.3	44.2	2.1	81.0			71.0		
41	41.3	39.3	2.0	47.5	45.4	2.1	83.0			72.8		
42	42.3	40.3	2.0	48.7	46.5	2.1	85.0			74.5		
43	43.3	41.3	2.0	49.8	47.7	2.2	87.1			76.3		
44	44.3	42.3	2.0	51.0	48.8	2.2	89.1			78.1		
45	45.3	43.3	2.0	52.2	50.0	2.2	91.1			79.9		
46	46.4	44.3	2.1	53.3	51.1	2.2	93.2			81.7		
47	47.4	45.3	2.1	54.5	52.3	2.2	95.2			83.5		
48	48.4	46.3	2.1	55.7	53.4	2.2	97.2			85.2		
49	49.4	47.3	2.1	56.8	54.6	2.2	99.3			87.0		
50	50.4	48.3	2.1	58.0	55.7	2.3	101.3			88.8		

Fig. 7. Bit complexities $b_{\mathsf{QKG}} := \log(\mathbb{E}[T_{\mathsf{QKG}}])$ and $b_{\mathsf{Mon}} := \log(\mathbb{E}[T_{\mathsf{Mon}}])$ of QuantumKeyGuess and Montanaro's algorithm, respectively.

5.3 QuantumKeyGuess compares well to Montanaro's algorithm

Our QuantumKeyGuess from Section 4 requires $\lceil \frac{\pi}{4} 2^{\lceil \mathrm{H}(\chi)n \rceil / 2} \rceil + 1$ many key guesses. Although our algorithm is a comparatively simple Grover-type applica-

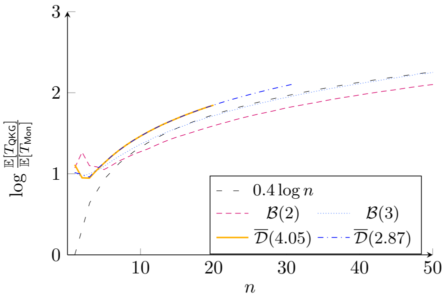

Fig. 8. Difference of bit complexities between QuantumKeyGuess and Montanaro's algorithm.

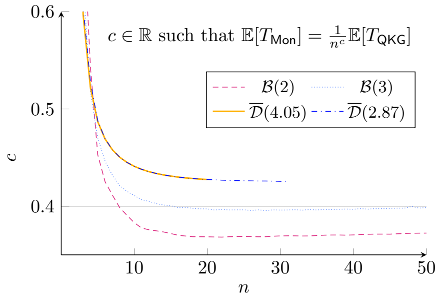

Fig. 9. Polynomial exponent $c \in \mathbb{R}$ such that $\mathbb{E}[T_{\mathsf{Mon}}] = \frac{1}{n^c}\mathbb{E}[T_{\mathsf{QKG}}]$.

tion, we already showed in Theorem 4.3 for Montanaro's more involved algorithm a lower bound that matches our number of key trials by a polynomial factor.

Since the calculation of $\mathbb{E}[T_{\mathsf{Mon}}]$ is hard for distributions with large support, we provide our results for $n \leq 50$ for $\mathcal{B}(2), \mathcal{B}(3)$, for $n \leq 21$ for $\overline{\mathcal{D}}(4.05)$ and for $n \leq 31$ for $\overline{\mathcal{D}}(2.87)$.

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 —modified such that it aborts once it tests more than $2^{\mathrm{H}(\chi)n}$ many vectors $\mathbf{X}$ to enable fair comparison— instead achieves an amount of key trials of

$$\mathbb{E}[T_{\mathsf{Mon}}] = \sum_{i=1}^{2^{\mathrm{H}(\chi)n}} p_i \sqrt{i}.$$

on expectation.

The bit complexities of $\mathbb{E}[T_{\mathsf{QKG}}]$ and $\mathbb{E}[T_{\mathsf{Mon}}]$ are provided in Fig. 7. Their differences are visualized in Fig. 8. Independent of the distribution, on this logarithmic scale all differences in Fig. 8 tend to $\leq \frac{1}{2}\log n$, implying that the ratio $\mathbb{E}[T_{\mathsf{QKG}}]/\mathbb{E}[T_{\mathsf{Mon}}]$ is bounded by $\sqrt{n}$.

Fig. 9 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.

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.

Ber23. Daniel J Bernstein. Asymptotics of hybrid primal lattice attacks. *Cryptology ePrint Archive*, 2023.

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`.

Leh60. D. H. Lehmer. Teaching combinatorial tricks to a computer. 1960.

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.

A Approximating the Falcon distribution

In this section, we have a detailed look at the proximity of the real Falcon distribution $\mathcal{D}(\sigma)$ and our approximate distribution $\overline{\mathcal{D}}(\sigma)$ as utilized in Section 5.

A.1 Approximating the Denominator in $\mathcal{D}(\sigma)$

By Definition 5.2, the distribution function of $\mathcal{D}(\sigma)$ is denoted with

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

As far as we are aware, there is no easy method to calculate the denominator for arbitrary σ. In order to be able to work with pretty accurate values of $P(a)$ which do not overestimate the final success probability of AbortedKeyGuess, we are required to find an approximation $D \in \mathbb{R}$ of the denominator that satisfies

$$D \geq \sum_{j\in\mathbb{Z}} \exp\left(\frac{-j^2}{2\sigma^2}\right) \qquad \text{and} \qquad \frac{\sum_{j\in\mathbb{Z}} \exp(\frac{-j^2}{2\sigma^2})}{D} \approx 1$$

which enables us to utilize

$$\frac{\exp(\frac{-a^2}{2\sigma^2})}{D} \lessapprox P(a)$$

instead.

Let $z \gg 0, \lambda \geq 2$ be such that $\frac{1}{\lambda}\exp(\frac{-z^2}{2\sigma^2}) > \exp(\frac{-(z+1)^2}{2\sigma^2})$. Due to $\exp(\frac{-a^2}{2\sigma^2})$ converging towards 0 exponentially fast, we know that this z exists for any $\lambda \geq 2$ and that

$$\frac{1}{2}\exp\left(\frac{-a^2}{2\sigma^2}\right) \geq \frac{1}{\lambda}\exp\left(\frac{-a^2}{2\sigma^2}\right) > \exp\left(\frac{-(a+1)^2}{2\sigma^2}\right)$$

for every $a \geq z$. By induction, it follows that

$$\exp\left(\frac{-z^2}{2\sigma^2}\right) = \sum_{k=1}^{\infty}\frac{1}{2^k}\exp\left(\frac{-z^2}{2\sigma^2}\right) \geq \sum_{j=z+1}^{\infty}\exp\left(\frac{-j^2}{2\sigma^2}\right),$$

and thus, due to the symmetry of $\exp(\frac{-j^2}{2\sigma^2})$ around 0, we conclude

$$D := \sum_{j=-z}^{z}\exp\left(\frac{-j^2}{2\sigma^2}\right) + 2\exp\left(\frac{-z^2}{2\sigma^2}\right) \geq \sum_{j\in\mathbb{Z}}\exp\left(\frac{-j^2}{2\sigma^2}\right).$$

Choosing $z \gg 0$ allows for $\frac{\sum_{j\in\mathbb{Z}}\exp(\frac{-j^2}{2\sigma^2})}{D}$ to become arbitrarily close to 1.

In our experiments, we chose $\lambda = 2^{10}$ and set z as

$$z = \min\left\{a \in \mathbb{N} \,\middle|\, \frac{1}{\lambda}\exp\left(\frac{-a^2}{2\sigma^2}\right) > \exp\left(\frac{-(a+1)^2}{2\sigma^2}\right)\right\}.$$

For $\sigma \in \{4.05, 2.87\}$, this yields $z = 115$ and $z = 58$, respectively, giving us the inequality

$$\sum_{j\in\mathbb{Z}}\exp\left(\frac{-j^2}{2\sigma^2}\right) \geq D - 2\exp\left(\frac{-z^2}{2\sigma^2}\right) \geq D\left(1 - 2\exp\left(\frac{-z^2}{2\sigma^2}\right)\right) \geq D(1-2^{-295}). \tag{5}$$

In our examples from Section 5, we analyze cases where $n \leq 50$. Let

$$P_n(\mathbf{a}) = \frac{N_1}{\sum_{j\in\mathbb{Z}}\exp(\frac{-j^2}{2\sigma^2})} \cdot \ldots \cdot \frac{N_n}{\sum_{j\in\mathbb{Z}}\exp(\frac{-j^2}{2\sigma^2})} = \frac{N}{(\sum_{j\in\mathbb{Z}}\exp(\frac{-j^2}{2\sigma^2}))^n}$$

denote the probability of sampling a specific vector $\mathbf{a} = (a_1, \ldots, a_n)$ (where N_ℓ denotes the numerator in $P(a_\ell)$ and $N := N_1 \cdot \ldots \cdot N_n$). According to Equation (5) and Bernoulli's inequality, our approximation $\frac{N}{D^n}$ of $P_n(\mathbf{a})$ then lies in the range

$$P_n(\mathbf{a}) \geq \frac{N}{D^n} \geq \frac{N(1-2^{-295})^n}{(\sum_{j\in\mathbb{Z}}\exp(\frac{-j^2}{2\sigma^2}))^n} = P_n(\mathbf{a})(1-2^{-295})^n \geq P_n(\mathbf{a})(1-50\cdot 2^{-295}).$$

We consider this to be a sufficient enough approximation for our purposes.

A.2 The truncated distribution closely resembles the real distribution

The distribution we utilized in Section 5 to present our results for Falcon does not describe the actual Falcon distribution. By definition of our algorithm, we are required to build the compact dictionary $\mathcal{D}_\chi^n$ first, which is impossible if our distribution is defined over an infinite support.

Instead, we opted to use a truncated version of $\overline{\mathcal{D}}(\sigma)$ instead, where any vector of length $n \leq 50$, sampled from said distribution, would lie in the truncated support with overwhelming probability. More formally, for some fixed $z' \in \mathbb{N}$ we denote with $\overline{\mathcal{D}}(\sigma)$ the distribution with probability distribution function

$$\overline{P}(a) = \Pr_{X \leftarrow \mathcal{D}(\sigma)}[X = a \mid |X| \leq z'] = \begin{cases} \frac{\exp(\frac{-a^2}{2\sigma^2})}{\sum_{j=-z'}^{z'} \exp(\frac{-j^2}{2\sigma^2})} & |a| \leq z' \\ 0 & |a| > z' \end{cases}.$$

Consider $z \gg 0$ from our results in Appendix A.1. We have shown that, for $n \leq 50$ and $\sigma \in \{4.05, 2.87\}$, the probability of sampling some element from $\{-z, \ldots, z\}$ is lower bounded by

$$\Pr[|X| \leq z] = 1 - 2 \sum_{j=z+1}^{\infty} P(j) \geq 1 - 2\exp\left(\frac{-z^2}{2\sigma^2}\right) \geq 1 - 2^{-295}.$$

Consequently, the probability to sample a vector of length n that contains only elements in the range $\{-z, \ldots, z\}$ happens with probability

$$\Pr[|X_i| \leq z \quad \forall\, 1 \leq i \leq n] \geq (1 - 2^{-295})^n \geq 1 - 50 \cdot 2^{-295},$$

which we consider to be likely enough to justify using $\overline{\mathcal{D}}(\sigma)$ with $z' = z$ instead of $\mathcal{D}(\sigma)$.

Note that

$$P(a) \leq \overline{P}(a) \leq \frac{P(a)}{1 - 2^{-295}} \qquad \text{for } a \in \{-z, \ldots, z\}.$$

Accordingly, we can assume that our estimated size of the core set of AbortedKeyGuess with Falcon using $\overline{\mathcal{D}}(\sigma)$ are only slightly higher than their counterparts for $\mathcal{D}(\sigma)$.

On a similar note, we have the chain of inequalities

$$(1 - 2^{-295})P(a) \leq \frac{\exp(-\frac{-a^2}{2\sigma^2})}{D} \leq P(a) \qquad \text{for } a \in \{-z, \ldots, z\},$$

so by using $\frac{\exp(-\frac{-a^2}{2\sigma^2})}{D}$ instead of $P(a)$ for calculating the success probability, we can ensure that the calculated success probability is only marginally less than the actual success probability for AbortedKeyGuess for $\mathcal{D}(\sigma)$.

We assume for either of these differences to be too small to make a noticeable impact on our results. In particular, within our bounds of computational accuracy, the fraction

$$\frac{\sum_{j=-z}^{z} \exp(\frac{-j^2}{2\sigma^2})}{D}$$

appears to be indistinguishable from 1.

B Implementing GetKey

In this section, we will give an overview on how to construct an efficient algorithm GetKey that, on input (n, P, A, j), returns $\mathbf{a}_j$ such that $\mathbf{a}_1$ has the highest sampling probability, $\mathbf{a}_2$ has the second highest and so on (where vectors $\mathbf{a}, \mathbf{a}'$ with $P(\mathbf{a}) = P(\mathbf{a}')$ are ordered in some fixed way, i.e. lexicographically). This procedure is required for both algorithms KeyGuess and AbortedKeyGuess.

Our main goal for this subsection is to prove the following Theorem

Theorem B.1. *There exists an algorithm* GetKey *that calculates the j-th most likely element with input (n, P, A, j) in time and space $\Theta(n^{|A|-1})$.*

We build on an approach suggested by Budroni and Mårtenson [BM23]. An implementation of GetKey can be found under `https://anonymous.4open.science/r/Entropy`.

B.1 Constructing the Compact Dictionary

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 immediately 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 B.2 (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. A compact dictionary can be constructed in time polynomial in n, as the following theorem shows:

Theorem B.3 (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*

$$|\mathcal{D}_\chi^n| = \binom{n + |A| - 1}{n} = \mathcal{O}(n^{|A|-1}). \tag{6}$$

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

Proof. Let $\widetilde{A^n}$ be defined as in Definition B.2, 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

$$\begin{aligned} \varphi : \widetilde{A^n} &\to \{(\alpha_1, \ldots, \alpha_m) \in \mathbb{N}_0^m \mid \textstyle\sum_{i=1}^m \alpha_i = n\}, \\ \mathbf{a} &\mapsto \qquad (\omega_1(\mathbf{a}), \ldots, \omega_m(\mathbf{a})). \end{aligned} \tag{7}$$

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 (6).

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 (7) allows us to efficiently construct $\widetilde{A^n}$, from which we then easily construct $\mathcal{D}_\chi^n$. □

B.2 From $\mathcal{D}_\chi^n$ to GetKey

With access to $\mathcal{D}_\chi^n$, we are able to efficiently implement GetKey as follows: Given A and P, we

- construct a compact dictionary $\mathcal{D}_\chi^n$,
- sort it by its second component in decreasing order and
- add a third component to every entry of the dictionary that represents the amount of keys with higher probability.

Let $S(\mathbf{a}) = \binom{n}{\omega_1(\mathbf{a}),\ldots,\omega_m(\mathbf{a})}$ be the amounts of distinct permutations of $\mathbf{a}$. Then, the k-th element of $\mathcal{D}_\chi^n$ is of the form

$$\left(\ \tilde{\mathbf{a}}_k\ ,\ \tilde{p}_k\ ,\ \varSigma_k\ \right) := \left(\ \tilde{\mathbf{a}}_k\ ,\ P(\tilde{\mathbf{a}}_k)\ ,\ \sum_{j=1}^{k-1} S(\tilde{\mathbf{a}}_j)\ \right)$$

and $\mathcal{D}_\chi^n$ is sorted w.r.t. the second component.

It is easy to see that any $\mathbf{a} \in A^n$ can be the j-th most likely element if and only if[5] it is a permutation of $\tilde{\mathbf{a}}_{k_j}$ where $k_j := \max\{k \in \mathbb{N} \mid \varSigma_k < j\}$. As the list is ordered by the second component, it is also sorted by the third component, and finding k_j can be done in $\mathcal{O}(\log |\mathcal{D}_\chi^n|) = \mathcal{O}(|A| \log n)$ via BinarySearch.

By definition, we have $\varSigma_{k_j} < j \leq \varSigma_{k_j+1} = \varSigma_{k_j} + S(\tilde{\mathbf{a}}_j)$, so the inequality

$$1 \leq j - \varSigma_{k_j} \leq S(\tilde{\mathbf{a}}_j)$$

holds. Consequently, all we need to do got find the j-th element is to apply an algorithm that returns the $(j - \varSigma_{k_j})$-th permutation of $\tilde{\mathbf{a}}_{k_j}$. This can be achieved with $\mathcal{O}(|A|n)$ many comparisons, e.g. by implementing a multiset variant of the Lehmer code [Leh60], denoted MultisetLehmer.

When combining these ideas, we end up with the following algorithm for GetKey (Algorithm 5):

Algorithm 5: GetKey

```
Input: $n, P, A, j$
Output: Key $\mathbf{a}_j$ with $P(\mathbf{a}_j) \geq P(\mathbf{a}_{j+1})$ for all $j < |A^n|$
1 Construct a compact dictionary $\mathcal{D}_\chi^n = \{(\mathbf{a}, \prod_{i=1}^n P(a_i) \mid \mathbf{a} \in A^n\}$ .
2 Sort $\mathcal{D}_\chi^n$ by 2nd component in decreasing order of probabilities $\prod_{i=1}^n P(a_i)$.
3 Append $\varSigma_k$ to each tuple $(\tilde{\mathbf{a}}_k, P(\tilde{\mathbf{a}}_k))$.
4 Find $k_j$ via BinarySearch in 3rd component.
5 Find $j - \varSigma_{k_j}$-th permutation $\sigma(\tilde{\mathbf{a}}_{k_j})$ via MultisetLehmer.
6 return $\sigma(\tilde{\mathbf{a}}_{k_j})$
```

Note that steps $1 - 3$, i.e. the construction and sorting of $\mathcal{D}_\chi^n$, do not depend on j, so it suffices to do these steps once. Consequently, after the initial call of GetKey, every consecutive call can be done in time $\mathcal{O}(|A|n)$.

This proves Theorem B.1 and shows that the Budroni-Mårtenson approach for representing χ^n compactly via $\mathcal{D}_\chi^n$ 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.

[5] We ignore cases with two distinct $\tilde{\mathbf{a}}, \tilde{\mathbf{a}}' \in \widetilde{A^n}$ where $P(\tilde{\mathbf{a}}) = P(\tilde{\mathbf{a}}')$.