IACR Communications in Cryptology
ISSN 3006-5496, Vol. 1, No. 3, 24 pages.

https://doi.org/10.62056/a6qj5w7sf

Optimizing c-sum BKW and Faster Quantum Variant for LWE

Jinzheng Cao[1], Qingfeng Cheng[1] and Jian Weng[2]

[1] Information Engineering University, Zhengzhou, China
[2] Jinan University, Guangzhou, China

Abstract. The Learning with Errors (LWE) problem has become one of the most prominent candidates to base PQC on, offering promising potential to meet the challenge of quantum computing. From a theoretical perspective, optimizing BKW algorithms to solve LWE is a vital task for the analysis of this cryptographic primitive. In this paper, we propose a fine-grained time/memory trade-off method to analyze c-sum BKW variants for LWE in both classical and quantum models, then offer new complexity bounds for multiple BKW variants determined by modulus q, dimension k, error rate α, and stripe size b. Through our analysis, optimal BKW parameters can be found for various LWE instances to minimize time and memory complexities. Furthermore, we enhance the performance of c-sum BKW by adopting the quantum Meet-in-the-Middle c-sum solver, which is exponentially faster than existing c-sum algorithms.

Keywords: LWE · BKW algorithm · Quantum algorithm · Time/memory tradeoff

1 Introduction

The BKW algorithm [BKW03] is an important combinatorial algorithm for solving LWE. The generic BKW algorithm, initially proposed to solve the Learning Parity with Noise (LPN) problem, consists of two main phases, the sample reduction phase and the distinguisher phase. In particular, the reduction phase of BKW reorganizes the samples to produce a new instance with reduced dimension and slightly larger errors. The distinguisher phase of BKW searches for the secret vector of the new instance, which is a partial solution to the original problem. The BKW algorithm was improved by applying the fast Walsh-Hadamard transformation method [LF06]. Guo et al. further modified BKW for LPN with a covering-codes-based procedure [GJL14], and extended it to LWE [GJS15]. The BKW algorithm was then improved by Zhang et al. [ZJW16] and Bogos et al. [BTV16] in a series of works. Although the original BKW algorithm and various improvements perform well in speed, its memory consumption grows dramatically on large LWE instances. To improve BKW's performance in real cryptographic circumstances, the balance of BKW's time/memory costs has become a topic of interest. Esser et al.

The work of Qingfeng Cheng (Corresponding Author) was supported by the National Natural Science Foundation of China under Grant Nos. 62472438 and 62172433, and the Natural Science Foundation of Henan under Grant No. 242300421414. The work of Jian Weng was supported by National Key Research and Development Plan of China under Grant No. 2020YFB1005600, Major Program of Guangdong Basic and Applied Research Project under Grant No. 2019B030302008, National Natural Science Foundation of China under Grant No. 61825203, Guangdong Provincial Science and Technology Project under Grant Nos. 2017B010111005 and 2021A0505030033, National Joint Engineering Research Center of Network Security Detection and Protection Technology, and Guangdong Key Laboratory of Data Security and Privacy Preserving.

E-mail: caojinzheng@126.com (Jinzheng Cao), qingfengc2008@sina.com (Qingfeng Cheng), cryptjweng@gmail.com (Jian Weng)

Received: 2024-06-26 Accepted: 2024-09-02

[EHK+18] discussed utilizing c-sum algorithms in BKW, which combines c samples at a time instead of only two, then further proposed a Dissection technique to iteratively divide the sample reduction task into smaller groups. Recently, Liu et al. [LY22] analyzed the time-space trade-off of BKW without additional heuristics.

The studies of BKW algorithm for LWE follow a similar route of research. Solving LWE with BKW has been suggested by Regev [Reg05] when initiating the problem. A systemic description of the LWE version of BKW was then proposed [ACF+15]. Following the initiation of BKW, a lazy modulus switching technique was discussed as a sample reduction technique [AFFP14]. The naive distinguishing phase is based on exhaustive search, and an FFT-based distinguisher was introduced in [DTV15]. The distinguisher was further improved in [GJMS17, KF15]. Esser et al. [EHK+18] discussed extending the c-sum BKW to solve LWE, and gave a complexity analysis. Liu et al. [LY22] introduced a complexity trade-off method for the initial form of BKW reduction as well as experimental analysis under relaxed conditions. In the quantum model, different versions of BKW for LPN/LWE have been studied in a series of works [KF15, NPS20]. From the perspective of attackers with access to quantum computers, one of the crucial tasks is to find efficient quantum sample reduction subroutines of BKW. A quantum c-sum BKW variant has been proposed by Esser et al. [EHK+18], who used a quantum c-sum algorithm with Grover's search in the BKW's sample reduction phase for solving LPN.

Our contributions. We now introduce our contributions in more detail. First, we propose an optimized time/memory complexity trade-off model of classic and quantum c-sum BKW variants for LWE. While previous results rely on a particular stripe size b, our analysis inspires us to find optimal parameters and reach lower complexities for given c. Second, we further improve the c-sum solver with Meet-in-the-Middle approach, and implement it as a subroutine in BKW. By partitioning and precomputing, this method can accelerate the sample reduction phase of BKW. We will also discuss its quantum complexity through the trade-off formula we develop.

Organization of the paper. The rest of the paper is organized as follows. Section 2 describes the preliminaries about the LWE problem and the BKW algorithm. Section 3 presents the first part of our work, a new complexity analysis of c-sum BKW algorithm for LWE. Section 4 introduces a quantum augmented BKW variant which uses Meet-in-the-Middle techniques for reduction, along with its complexity trade-off. We conclude the paper in Section 5.

2 Preliminaries

We will use the following notations throughout the paper. When $N \in \mathbb{N}$, let $[\pm N]$ denote set $\{\pm 1, \pm 2, \ldots, \pm N\}$. For a set S and integer $c \leq |S|$, let $\binom{S}{c}$ denote the set of all c-size subsets of S, and $\binom{[\pm N]}{c}$ is the set of all c-size subsets of $[\pm N]$. When a list $L \in S^n$ is presented as $L = (l_1, \ldots, l_n)$, we have $l_{-i} = -l_i$.

2.1 LWE

We briefly review basic concepts about the LWE problem. Suppose we have a list of samples $(a_i, b_i), i = 1, \cdots, m$ with $b_i = \langle a_i, s_i \rangle + e_i \in \mathbb{F}_q$. The error e_is are sampled from χ. The search-LWE problem is to compute secret vector $s \in \mathbb{F}_q^k$. The decision-LWE is to decide whether e is sampled from χ or from the uniform distribution. We denote $\alpha = \sigma/q$ as the error rate.

In the study of the BKW algorithm for LWE, we often resort to the **Independence Heuristic**, which assumes that although sums of LWE samples $(a_{i_1} \pm \ldots \pm a_{i_c})$ are stochastically dependent after a sample reduction operation, such dependency does not affect the reduction of the next stripe. Under the Independence Heuristic, these new samples can be treated as independent samples. The correctness of the assumption has been verified in [EHK+18].

2.2 Statistics

We use Ber_p to denote the Bernoulli distribution where $X \sim \text{Ber}_p$ if $Pr[X = 1] = 1 - Pr[X = 0] = p$. Adding up n random variables $X_1, X_2, \dots, X_n$ from Ber_p produces an X following binomial distribution with parameters n, p. We denote the distribution by $\text{Bin}_{n,p}$. Suppose $\mathcal{N}(\mu, \sigma^2)$ be the Gaussian distribution of mean μ and standard deviation σ. In LWE, we need a Gaussian distribution over $\mathbb{Z}_q$. We discuss two kinds of discrete Gaussian variants over $\mathbb{Z}_q$. The first, rounded Gaussian distribution, was suggested by [Reg05]. Consider a wrapped Gaussian distribution $\Psi_{\sigma,q}$ with probability density $p(\theta; \sigma, q)$ given by $p(\theta; \sigma, q) = \sum_{k=-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{-(\theta+kq)^2}{2\sigma^2}\right], \theta \in [-\frac{q}{2}, \frac{q}{2}]$. The distribution is obtained by sampling from $\Psi_{\sigma,q}$ then rounding to the nearest integer in $[-\frac{q}{2}, \frac{q}{2}]$. The distribution is denoted by $\bar{\Psi}_{\sigma,q}$, whose probability mass function is defined as

$$\Pr[x \leftarrow \bar{\Psi}_{\sigma,q}] = \int_{x-\frac{1}{2}}^{x+\frac{1}{2}} p(\theta; \sigma, q)\ \mathrm{d}\theta, x \in [-\frac{q}{2}, \frac{q}{2}].$$

The other Gaussian distribution over $\mathbb{Z}_q$ is the discrete Gaussian distribution, which is the most commonly used in LWE applications. For integer x in $[-\frac{q}{2}, \frac{q}{2}]$, the discrete Gaussian distribution $D_{\sigma,q}$ is defined as

$$\Pr[x \leftarrow D_{\sigma,q}] = \frac{\exp(-x^2/(2\sigma^2))}{\sum_{k\in[-\frac{q}{2},\frac{q}{2}]} \exp(-K^2/(2\sigma^2))}.$$

2.3 Quantum computing

Our quantum BKW algorithm is based on the Grover's quantum search. Given a set of objects indexed by $[1, 2, \dots, N]$, we setup an identifier oracle $\mathcal{O}$ such that in the classical setting, $\mathcal{O}(i) = 1$ ($\mathcal{O}|i\rangle = -|i\rangle$ in the quantum setting) when i is a target, and $\mathcal{O}(i) = 0$ ($\mathcal{O}|i\rangle = |i\rangle$ in the quantum setting) when i is not. The search algorithm aims to find a target $j \in 1, 2, \dots, N$ by making queries to oracle $\mathcal{O}$. While in the classical algorithm we need $O(N)$ queries, the quantum search solves the problem with only $O(\sqrt{N})$ queries. The quantum circuit model evaluates the time complexity as the circuit size, or the total number of elementary quantum gates.

In our quantum algorithm, we still need to access classical data. To this end, we adopt the quantum random-access memory (QRAM), specifically, we use the classical memory with quantum random access [Kup13]. It is possible to implement QRAM using a universal quantum gate set at a considerable cost. Given classical registers $|x_0, \dots, x_{r-1}\rangle$, [KP20] proves that QRAM can be constructed in time $\mathcal{O}(r)$. The query of data from register x_i in the form $|i\rangle|y\rangle \mapsto |i\rangle|y \oplus x_i\rangle$ has polylogarithmic depth.

3 New c-Sum BKW Complexity Trade-off and Optimization

Our work focuses on the improvement and analysis of the BKW algorithm for LWE, especially one of its versions called c-sum BKW. Generally, the BKW algorithm consists of two phases – the reduction phase and the distinguisher phase. The sample reduction phase of original BKW looks for pairs of input LWE samples $(a_{i_1}, b_{i_1}), (a_{i_2}, b_{i_2})$ such that they

add to zero in a block of b entries (called a **stripe**), e.g., $a_{i_1} \pm a_{i_2} = (\underbrace{0, \ldots, 0,}_{b\text{-size stripe}} *, \ldots, *)$.
It thus produces combined samples $(a'_i, b'_i) = (a_{i_1} \pm a_{i_2}, b_{i_1} \pm b_{i_2})$ which in fact form a new LWE instance whose dimension is reduced by b, and secret s_1 is a $(k-b)$-block of original secret vector s. By iteratively running the process for $(a-1)$ times, the sample reduction phase obtains a list of new LWE samples with lower dimension $k' = k - (a-1)b$. In the distinguisher phase, we run majority vote or FFT on the new samples to recover the new LWE secret vector, which forms partial entries of the original secret s. The procedure to recover other entries of s is likewise.

3.1 Review of c-sum algorithm

Esser et al. [EHK+18] proposed the first BKW variant with support for time/space trade-off, called the c-sum BKW. Instead of always combining 2 samples, the c-sum technique combines $c \geq 2$ samples that add to zero on a stipe, i.e. $(a_{i_1}, b_{i_1}), (a_{i_2}, b_{i_2}), \ldots, (a_{i_c}, b_{i_c})$ such that $\underbrace{a_{i_1} \pm a_{i_2} \pm \ldots \pm a_{i_c}}_{c} = (\underbrace{0, \ldots, 0,}_{b\text{-size stripe}} *, \ldots, *)$. Formally, the c-sum problem for LWE is defined as follows.

Definition 1 (The c-Sum Problem, c-SP$_b$)**.** Given $b, c, N \in \mathbb{N}$ with $c \geq 2, q$ is a prime. Let $L = (l_1, \ldots, l_N)$ be a list with l_i sampled from $\mathcal{U}_{\mathbb{F}_q^b}$ for all i, and let $t \in \mathbb{F}_q^b$ be a target. A single-solution of the c-SP$_b$ is a c-size set $\mathcal{L} \in \binom{[\pm N]}{c}$ such that

$$\sum_{j \in \mathcal{L}} l_j = t.$$

A solution of the c-SP$_b$ is a set of N or more distinct single-solutions. The c-sum problem consists of finding a solution given L, t and a fixed c.

The reduction from sample reduction phase in CC-BKW to an instance of c-sum problem (c-SP$_b$) is natural: When we aim to find c LWE samples which add to 0^b on a b-size stripe, we first project the samples to this stripe. Then, we find these collisions on the projected stripe by c-sum solver. Finally, we sum up the corresponding samples. Note that a solution of the c-SP$_b$ consists of N or more distinct single-solutions. Therefore, the sample reduction phase uses the output of previous c-sum subroutines as input for a new c-sum instance, allowing us to move from one stripe to the next while keeping the number of input samples. A basic c-sum solver is described in Algorithm 1 by extending the LPN version of c-sum in [EHK+18].

Algorithm 1 Basic c-SP$_b$ solver

```
Require: sorted list $L = (l_1, \ldots, l_N) \in (\mathbb{F}_q^b)^N$, target vector $t \in \mathbb{F}_q^b$
Ensure: list $S$ or $\emptyset$
  for all $\nu = \{i_1, \ldots, i_{c-1}\} \in \binom{[\pm N]}{c-1}$ do
    for all $i_c \in [\pm N] \setminus \nu$ satisfying $t = l_{i_c} + \left(\sum_{i \in \nu} l_i\right)$ do
      $S \leftarrow S \bigcup \{\{i_1, \ldots, i_c\}\}$
      if $|S| = N$ then
        return $S$
      end if
    end for
  end for
  return $\emptyset$
```

Algorithm 2 c-sum-BKW^{LWE}

```
Require: dimension $k$, sample number $m$, modulus $q$, $N \geq q^{\frac{\log_q(\frac{q}{q-1})+c\cdot\log_q(c/2)+b}{(c-1)}}$, $\epsilon_a > 0$
Ensure: secret $s \in \mathbb{F}_q^k$
  set number of stripe $a > 0$, stripe size $b := \frac{k(1+\epsilon_a)}{a}$, $m := b \cdot e^{\frac{2\pi^2\sigma^2 c^a}{q^2}}$
  for $i \leftarrow 1, \ldots, m$ do
    set list of samples $S_{sum} = \emptyset$
    generate $N$ LWE samples stored in $L$
    for $j \leftarrow 1, \ldots, a-1$ do
      project vectors in $L$ onto their $j$-th stripe to obtain a list $L'_j$
      solve the $c$-sum problem instance $c$-SP$(L'_j, 0^b)$ with Algorithm 1, to obtain a solution $S$
      sum up the vectors in $L$ as specified in $S$, and rewrite list $L$ as the list of these vector sums
    end for
    if $L = \emptyset$ then
      return $\emptyset$
    end if
    $S_{sum} \leftarrow S_{sum} \bigcup L$
  end for
  run FFT distinguisher on $S$ to find $k'$ components of secret vector $s$
  determine other components of $s$ iteratively
  return $s$
```

We will prove that the c-sum solver in Algorithm 1 runs in time $\tilde{\mathcal{O}}\left(N^{c-1}\right)$ and memory $\tilde{\mathcal{O}}(N)$ in Lemma 1. Compared with analogues in LPN, the algorithm considers adding $\pm l_i$ for every $l_i \in L$. Equipped with this c-sum algorithm, we describe the c-sum BKW for k-dimensional LWE in Algorithm 2. By calling this c-sum solver, c-sum BKW combines c independent samples to produce a new LWE instance of reduced dimension $k' = k-(a-1)\cdot b$, while the new secret key is composed of the first k' components of original secret s. Then the secret is recovered by FFT in the distinguishing phase. On the basis of c-sum BKW, [EHK+18] proposes the Dissection BKW. This variant is equipped with a time-optimized Dissection c-sum solver. The complexities of these BKW variants will be discussed in the following subsection.

3.2 New complexity trade-off in classical model

We now provide a novel model to analyze the performance of c-sum BKW. Based on the complexity analysis, we discuss its applications in cryptanalysis of LWE. The analysis begins with the classical model, and will then extend to the quantum complexities. The analysis will follow 3 steps.

- First, we consider the complexities of individual c-sum operations and total amount of operations in BKW, summarized in Lemma 1 and Lemma 2.
- Second, we construct the trade-off model for the whole c-sum BKW algorithm through Lemma 4, 3 and summarize the result in Theorem 1.
- Finally, we choose the optimal parameters for c-sum BKW in Corollary 1, and compare its efficiency with previous works. We also extend our model to other cases such as Dissection BKW.

3.2.1 Analyzing cost of c-sum solver

First, we give an analysis of the c-sum solver which underlies the BKW algorithm. Our analysis depends on estimating the success probability of c-sum subroutines. For simplicity, we adopt the Independence Heuristic: The dependency of c-sums $\sum_{j\in\mathcal{L}} l_j$ only mildly affect the performance of BKW algorithms. So in the following analysis, we can still treat the c-sums of samples as independent vectors. We begin with the condition of success for the classical c-sum solver:

Lemma 1. *Let (L,t) be a c-SP instance with*

$$|L| = N, N = q^{\frac{\log_q(\frac{q}{q-1})+c\log_q(c/2)+b}{c-1}}.$$

Under the Independence Heuristic, (L,t) has at least N single solutions with probability $1-\exp\left(-\frac{N}{2q(q-1)}\right)$.

Proof. We define an indicator $X_{\mathcal{L}}$ such that $X_{\mathcal{L}} = 1 \Leftrightarrow \sum_{j\in\mathcal{L}} l_j = t$ for every $\mathcal{L} \subseteq [\pm N]$ with $|\mathcal{L}| = c$. Under the Independence Heuristic, $X_{\mathcal{L}}$ follows $\mathrm{Bin}_{\binom{N}{c},q^{-b}}$, and $\mathbb{E}[X] = 2^c \cdot \binom{N}{c} \cdot q^{-b} \geq 2^c \cdot \left(\frac{N}{c}\right)^c \cdot q^{-b}$. Taking logarithms, we have

$$\begin{aligned}
\log_q \mathbb{E}[X] &\geq c(\log_q 2 + \log_q N - \log_q c) - b \\
&\geq c\left(\frac{\log_q(\frac{q}{q-1}) + c\log_q(\frac{c}{2}) + b}{c-1} + \log_q 2 - \log_q c\right) - b \\
&= \frac{c\log_q(\frac{q}{q-1}) + cb + c\log_q(c) - c\log_q(2)}{c-1} - b \\
&= \frac{\log_q(\frac{q}{q-1}) + c\log_q(c) - c\log_q(2) + b}{c-1} + \log_q(\frac{q}{q-1}) \\
&= \log_q(N) + \log_q(\frac{q}{q-1}).
\end{aligned}$$

Therefore, our N value ensures that $\mathbb{E}[X] \geq \frac{q}{q-1} \cdot N$. We are able to estimate the probability of having fewer than N single solutions with the Chernoff bound: [Cry06], $\Pr[X < N^{\alpha}] \leq \Pr\left[X < \frac{q}{q-1}\mathbb{E}[X]\right] \leq \exp\left(-\frac{\mathbb{E}[X]}{2q^2}\right) \leq \exp\left(-\frac{N}{2q(q-1)}\right)$. Further, the probability of existing at least N single solutions is $1-\exp\left(-\frac{N}{2q(q-1)}\right)$. □

The reduction phase of c-sum BKW produces new LWE samples, where Algorithm 1 is used as the c-sum solver. To analyze the complexity of BKW, we first study the time/memory cost of the c-sum algorithm. We assume the list of samples L is first sorted, which requires time $\tilde{\mathcal{O}}(N)$. Specifically, the target vector for c-sum is an all-zero block 0^b in the context of BKW, so the combination $\{i_1,\ldots,i_c\}$ and $\{-i_1,\ldots,-i_c\}$ are considered the same. As a result, the algorithm can only search a half of the space, and the time complexity is $\tilde{\mathcal{O}}(2^{c-2}N^{c-1})$. Therefore, the complexity of the c-sum subroutine is estimated as follows.

Lemma 2. *c-sum-q recovers a solution of c-SP$_b$ in time $\tilde{\mathcal{O}}((2N)^{c-1})$ and memory $\tilde{\mathcal{O}}(N)$.*

Proof. The correctness of c-sum-q can be proven by [EHK+18, Lemma 4.1]. For the time cost, the outer **for** loop of Algorithm 1 iterates all $\binom{N}{c-1}$ possible $\{i_1,\ldots,i_{c-1}\}$ sets, and for every index i_j, $\pm i_j$ are considered separately. Therefore, $\binom{N}{c-1}\cdot 2^{c-1} \leq (2N)^{c-1}$ combinations are searched. In a sorted L, each solution is found in $\mathcal{O}(\log N) = \tilde{\mathcal{O}}(1)$. This adds up to a run time of $\binom{N}{c-1}\cdot 2^{c-1} \leq (2N)^{c-1}$. The algorithm returns at most N single solutions, so the memory consumption is $\tilde{\mathcal{O}}(\max(N,|S|)) = \tilde{\mathcal{O}}(N)$. □

3.2.2 Analyzing number of samples based on FFT distinguisher

We now try to estimate the complexity of the c-sum BKW algorithm as a whole. In particular, the analysis of c-sum BKW faces two primary questions. First, the cost of the c-sum solver; Second, how many c-sum subroutines are required in the entire BKW algorithm. The first question is answered by the previous subsection. In this subsection, we estimate the required number m of new k'-dimensional samples to solve LWE.

Before this task, we need to explain the basic qualities of the FFT distinguisher, which determines the number of required input samples. After obtaining a k'-dimensional LWE instance (A', b') by sample reduction, we write the function

$$f(x) = \sum_{j=1}^{m} I(A'_j = x)\theta_q^{b_j}, x \in \mathbb{Z}_q^k,$$

where θ_q is the primitive root. The FFT distinguisher computes the FFT of f , i.e.

$$\hat{f}(s') = \sum_{j=1}^{m} \theta_q^{-(\langle A'_j, s'\rangle - c_j)} = \sum_{j=1}^{m} \theta_q^{-(\nu_{j,1} \pm \cdots \pm \nu_{j,c^{a-1}})},$$

where $\nu_{j,l}$ are independent samples from the original LWE oracle. When there are sufficient samples, then the correct secret vector s' will give the maximum value of $Re(\hat{f}(s'))$. With the notations, we are able to analyze the FFT process.

We define

$$\mathcal{R}_{\sigma,q,\chi} := \begin{cases} \dfrac{q}{\pi}\sin\left(\dfrac{\pi}{q}\right)e^{-2\pi^2\sigma^2/q^2} & X \sim \bar{\Psi}_{\sigma,q} \\ 1 - \dfrac{2\pi^2\sigma^2}{q^2} & X \sim D_{\sigma,q}. \end{cases}$$

The following lemma states its property.

Lemma 3. $\mathbb{E}\left[Re(\hat{f}(s'))\right] \geq m \cdot (\mathcal{R}_{\sigma,q,\chi})^{c^{a-1}}$.

Proof. Through the independence of $\nu_{j,l}$ and $\mathbb{E}[\theta_q^{\pm\nu_{j,l}}] = \mathbb{E}[\cos(2\pi\nu_{j,l}/q)]$, we get

$$\begin{aligned} \mathbb{E}\left[\mathrm{Re}(\hat{f}(s'))\right] &= \mathrm{Re}\left(\sum_{j=1}^{m}\mathbb{E}\left[\theta_q^{-(\nu_{j,1}\pm\cdots\pm\nu_{j,c^{a-1}})}\right]\right) \\ &= \mathrm{Re}\left(\sum_{j=1}^{m}\mathbb{E}\left[\cos\left(\frac{2\pi}{q}\nu_{j,1}\right)\right]^{c^{a-1}}\right). \end{aligned} \tag{1}$$

□

Lemma 4 (Lemma 10, 11 in [DTV15]). *For q an odd integer, if $X \sim \bar{\Psi}_{\sigma,q}$ and $Y \sim 2\pi X/q$, then $\mathbb{E}[\cos(Y)] \geq \frac{q}{\pi}\sin\left(\frac{\pi}{q}\right)e^{-2\pi^2\sigma^2/q^2}, \mathbb{E}[\sin(Y)] = 0$. If $X \sim D_{\sigma,q}$, then $\mathbb{E}[\cos(Y)] \geq 1 - \frac{2\pi^2\sigma^2}{q^2}, \mathbb{E}[\sin(Y)] = 0$.*

We get $\mathbb{E}\left[\cos(2\pi\nu_{j,l}/q)\right] \geq \mathcal{R}_{\sigma,q,\chi}$ based on Lemma 4. Combined with Lemma 3, we have $\mathbb{E}\left[\mathrm{Re}(\hat{f}(s'))\right] \geq \sum_{j=1}^{m}(\mathcal{R}_{\sigma,q,\chi})^{c^{a-1}} = m \cdot (\mathcal{R}_{\sigma,q,\chi})^{c^{a-1}}$. Based on this result, the probability that FFT distinguisher finds the correct s' is $1 - q^{k'} \cdot \exp\left(-\frac{m}{8} \cdot (\mathcal{R}_{\sigma,q,\chi})^{c^a}\right)$. The detailed analysis can be found in [DTV15]. Therefore, we estimate the required m to recover the correct s'. For the c-sum BKW procedure in Algorithm 2 and a given failure probability ϵ that s' is not found, we need

$$\begin{aligned}\epsilon &= \Pr\left[\exists\alpha \neq s' : \mathrm{Re}(\hat{f}(\alpha)) \geq \mathrm{Re}(\hat{f}(s'))\right] \\ &< q^{k'} \cdot \exp\left(-\frac{m}{8}\cdot (\mathcal{R}_{\sigma,q,\chi})^{c^{a-1}}\right).\end{aligned} \tag{2}$$

Solving the inequality for m, we get

$$m^{c^a} \geq \begin{cases} 8k' \log\left(\frac{q}{\epsilon}\right)\frac{q}{\pi}\sin\left(\frac{\pi}{q}\right)e^{\frac{-2\pi^2\sigma^2}{q^2}}, X \sim \bar{\Psi}_{\sigma,q} \\ 8k' \log\left(\frac{q}{\epsilon}\right)\left(1-\frac{2\pi^2\sigma^2}{q^2}\right), \qquad X \sim D_{\sigma,q}. \end{cases} \tag{3}$$

3.2.3 Complexity trade-off of c-sum BKW for LWE

Putting the results of Subsection 3.2.1 and Subsection 3.2.2 together, we achieve the main result about the c-sum BKW's time and memory costs under the Independence Heuristic. We take the c-sum BKW variant in the classic computing model as an example, and present the key result in Theorem 1. We will later extend to more BKW variants in Theorem 2 and Theorem 4.

Theorem 1 (Classical trade-off for c-sum BKW)**.** *For given $c \in \mathbb{N}$, $\epsilon > 0$, c-sum-BKW^{LWE} solves the k-dimensional LWE instance in time $T = 2^{\vartheta(1+\epsilon)}$ and memory $M = 2^{\mu^{(1+\epsilon)}}$, where $\vartheta = \log(k) + \frac{2\pi^2\sigma^2\log(e)}{q^2}\cdot c^{k/b} + \log(q)\cdot b, \mu = \log(b) + \frac{2\pi^2\sigma^2\log(e)}{q^2}\cdot c^{k/b} + \log(q)\cdot\frac{b}{c-1}$.*

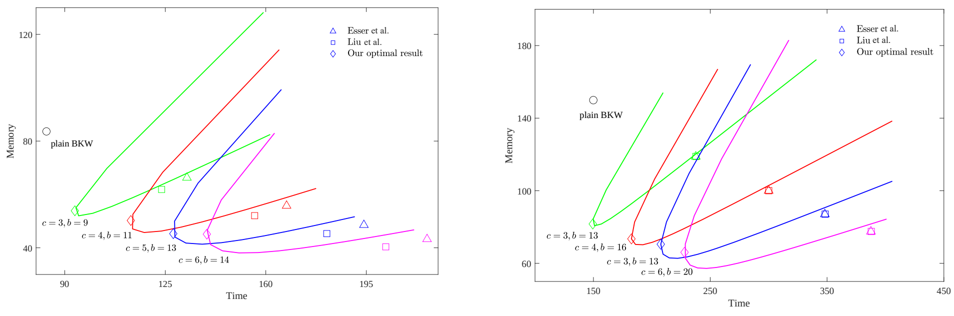
Proof. Let $N = q^{\frac{\log_q(\frac{q}{q-1})+c\log_q(c/2)+b}{c-1}}$. Lemma 1 ensures the success probability of c-sum subroutines. From Lemma 2, we denote the time and memory cost of c-sum-q by $T_{c,N} = \tilde{\mathcal{O}}((2N)^{c-1}), M_{c,N} = \tilde{\mathcal{O}}(N)$. When analyzing the run time of BKW, we neglect the overhead caused by failures of c-sum algorithm. For $T_{c,N} \geq N$, an iteration of m outer **for**-loops has time complexity $\tilde{\mathcal{O}}(\max\{N,(a-1)\cdot T_{c,N}\}) = \tilde{\mathcal{O}}(a\cdot T_{c,N})$. Hence, to recover m samples, we need time $\tilde{\mathcal{O}}(m\cdot a\cdot T_{c,N}) = \tilde{\mathcal{O}}(m\cdot a\cdot(2N)^{c-1}) = \tilde{\mathcal{O}}(m\cdot a\cdot q^b)$ for a constant c. So the c-sum-BKW^{LWE} has time complexity $T = m\cdot a\cdot q^{b(1+o(1))}$. From Lemma 4, Lemma 3, and Equation (3), we summarize that $m = \tilde{\mathcal{O}}\left(b\cdot e^{\frac{2\pi^2\sigma^2c^a}{q^2}}\right)$. Since we set the stripe width $b = \left\lfloor\frac{k}{a}\right\rfloor$, we get the time complexity $2^{\vartheta(1+\epsilon)}$, where$\vartheta = \log(k) + \frac{2\pi^2\sigma^2\log(e)}{q^2}\cdot c^{k/b} + \log(q)\cdot b$. Similarly, we estimate the memory cost. $M = m\cdot M_{c,N} = m\cdot\left(q^{\frac{b}{c-1}}\right)^{1+o(1)} = 2^{\mu}(1+\epsilon)$, where $\mu = \log(b) + \frac{2\pi^2\sigma^2\log(e)}{q^2}\cdot c^{k/b} + \log(q)\cdot\frac{b}{c-1}$.

□

For given q,σ and constant c, this result shows how BKW's time/memory complexity changes with the stripe size b. With error rate $\alpha = \sigma/q$, the result is simplified as

$$\begin{aligned}\vartheta &= \log(k) + 2\pi^2\alpha^2\log(e)\cdot c^{k/b} + \log(q)\cdot b, \\ \mu &= \log(b) + 2\pi^2\alpha^2\log(e)\cdot c^{k/b} + \log(q)\cdot\frac{b}{c-1}.\end{aligned}$$

Figure 1 compares our time/memory trade-off with previous results. By adjusting the values of parameter b, we get different time/memory complexities, corresponding to the curves in the figure. We also mark the results of Esser et al. [EHK+18], Liu et al. [LY22], and the plain BKW, which rely on a given b value which is not necessarily optimal.



(a) $k = 50, \alpha = 0.03$ (b) $k = 100, \alpha = 0.01$

Figure 1: BKW time/memory trade-off and optimized complexity results

3.2.4 Optimal b choice

Theorem 1 has indicated the connection between c-sum BKW's complexity and stripe size b. Through the trade-off formula, we choose proper b to minimize time complexity. By evaluating $\frac{d\vartheta}{db} = 2\pi^2\alpha^2\log(c)\cdot c^{k/b}\cdot\frac{-k}{b^2}+\log(q)$, $\frac{d\mu}{db} = \frac{1}{b\ln(2)}+2\pi^2\alpha^2\log(c)\cdot c^{k/b}\cdot\frac{-k}{b^2}+\frac{\log(q)}{c-1}$, theoretically optimal b values can be found by choosing b that makes $\frac{d\vartheta}{db} = 0$ or $\frac{d\mu}{db} = 0$. The results are shown in Corollary 1.

Corollary 1 (Optimal parameters for minimized complexity)**.** *For given c, when b satisfies $2\pi^2\alpha^2\log(c)\cdot c^{k/b}\cdot\frac{k}{b^2} = \log(q)$, c-sum BKW has minimized time complexity. When b satisfies $2\pi^2\alpha^2\log(c)\cdot c^{k/b}\cdot\frac{k}{b^2} = \frac{1}{b\ln(2)}+\frac{\log(q)}{c-1}$, c-sum BKW has minimized memory complexity.*

Examples of optimal parameters are also illustrated in Figure 1. For each curve which represents the time/memory costs for an LWE instance, the integer point most close to the origin point is marked to illustrate the minimum computation cost. By choosing proper b values, we can achieve the optimized BKW performance. Note that in the BKW algorithm, b should be integers. Therefore, we find valid b values by rounding to the closest integer in practical circumstances. In Figure 1, we mark the parameter settings that lead to minimized time complexities for different instances. It is obvious in the figure that our new complexity results have lower time and memory costs than previous works.

3.2.5 Complexity trade-off of Dissection BKW for LWE

Our analysis can be extended to estimate the complexity of the Dissection approach. The method is proposed by Esser et al. [EHK⁺18] to reduce the running time of the c-sum solver. With existing results about the complexity of Dissection subroutine, we give the complexity trade-off of Dissection BKW in Theorem 2.

Theorem 2 (Classical trade-off for Dissection BKW)**.** *For given $c_i \in \mathbb{N}$, $\epsilon > 0$, $c_{-1} = 1, c_i = c_{i-1}+i+1$, when using Dissect method to solve SP, BKW finds the k-dimensional LWE key in time $T = 2^{\vartheta(1+\epsilon)}$ and memory $M = 2^{\mu^{(1+\epsilon)}}$, where $\vartheta = \log k + 2\pi^2\alpha^2\log e\cdot c^{\frac{k}{b}}+\log(q)\cdot(1-\frac{i}{c_i-1})\cdot b, \mu = \log(b)+\frac{2\pi^2\sigma^2\log(e)}{q^2}\cdot c^{k/b}+\log(q)\cdot\frac{b}{c-1}$.*

Proof. Let $N = q^{\frac{\log_q(\frac{q}{q-1})+c_i\log_q(c_i/2)+b}{c_i-1}}$. According to [EHK⁺18], the Dissection method solves c_i-SP in time $T_{c_i,N} = \tilde{\mathcal{O}}(N^{c_{i-1}})$, memory $M_{c,N} = \tilde{\mathcal{O}}(N)$. Further, $c_i = c_{i-1}+i+1$.

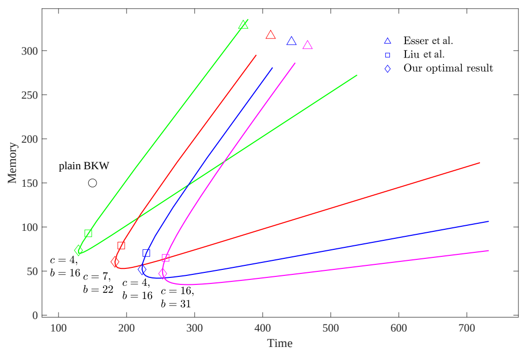


Figure 2: Dissection BKW time/memory tradeoff

To recover m samples, we need time

$$\begin{aligned}\log T =& (\log k + 2\pi^2\alpha^2 \log e \cdot c^{\frac{k}{b}} + \log(q) \cdot \frac{c_{i-1}}{c_i - 1} \cdot b) \cdot (1+\varepsilon)\\ =& (\log k + 2\pi^2\alpha^2 \log e \cdot c^{\frac{k}{b}} + \log(q) \cdot (1 - \frac{i}{c_i - 1}) \cdot b) \cdot (1+\varepsilon)\end{aligned}$$

for given c_i. Similarly, we estimate the memory $M = m \cdot M_{c_i,N} = m \cdot \left(q^{\frac{b}{c_i-1}}\right)^{1+o(1)} = 2^{\mu}(1+\epsilon), \mu = \log(b) + \frac{2\pi^2\sigma^2 \log(e)}{q^2} \cdot c_i^{k/b} + \log(q) \cdot \frac{b}{c_i-1}$. □

The trade-off formula enables us to minimize the complexities of Dissection BKW. A comparison in $k = 100, \alpha = 0.01$ can be found in Figure 2. Our chosen parameter b is close to Liu et al. [LY22] and has an advantage over the original Dissection BKW by Esser et al. [EHK+18].

3.3 Complexity trade-off in quantum model

For the analysis of post-quantum cryptography, we are naturally interested in how Grover's quantum search algorithm and other quantum oracles can contribute to the c-sum subroutine in BKW. We provide a complexity evaluation for BKW in quantum computing model in this subsection. The property of Grover's quantum search can be described with the amplitude amplification.

Theorem 3 (Amplitude amplification [BHMT02])**.** *Suppose we have a set of N objects of which some are targets. Let $\mathcal{O}$ be a quantum oracle that identifies the targets. Let $\mathcal{A}$ be a quantum circuit using no intermediate measurements, i.e. reversible. Let a be the initial success probability of $\mathcal{A}$, that is the probability that a measurement of $\mathcal{A}|0\rangle$ outputs a target. There exists a quantum algorithm that calls $\mathcal{O}(\sqrt{1/a})$ times $\mathcal{A}, \mathcal{A}^{-1}$ and $\mathcal{O}$, uses as many qubits as $\mathcal{A}$ and $\mathcal{O}$, and outputs a target with probability greater than $1-a$.*

Grover's search can be viewed as a special case of the theorem when we define that $\mathcal{A}$ produces the index of a uniformly selected object. When we use Grover's algorithm to find x such that $f(x) = 1$ for function $f : D \to \{0,1\}$, we have $a = \frac{|f^{-1}(1)|}{|D|}$. In c-sum, the f is defined as

$$f_t : \binom{[\pm|L|]}{c-1} \to \{0,1\}, \nu \longmapsto \begin{cases} 1 & \exists i_c \in [\pm|L|] \setminus \nu : \sum_{j=1}^{c-1} l_{i_j} = -l_{i_c} + t \\ 0 & else. \end{cases}$$

Algorithm 3 Quantum-c-sum-q

```
Require: sorted list $L = (l_1, \ldots, l_N) \in (\mathbb{F}_q^b)^N$, target vector $t \in \mathbb{F}_q^b$
Ensure: list $S$ or $\perp$
  create a QRAM $O_W$
  for every sample $l_j$ in $L$ do
    add $l_j$ to $O_W$ at index $j$
  end for

  create oracle $\hat{\mathcal{O}}(i_1, \ldots, i_{c-1})$ :
  get $l_{i_1}, \ldots, l_{i_{c-1}}$ from $O_W$
  if exists $l_{i_c}$ such that $\sum_{j=1}^{c-1} l_{i_j} = -l_{i_c} + t$ then
    return 1
  end if
  return 0
  end oracle $\hat{\mathcal{O}}(i_1, \ldots, i_{c-1})$

  repeat $\tilde{\mathcal{O}}(N)$ times:
  use Grover's search to find $(i_1, \ldots, i_{c-1})$ such that $\hat{\mathcal{O}}(i_1, \ldots, i_{c-1}) = 1$
  recover $i_c \in [\pm N] \setminus \nu$ satisfying $\sum_{j=1}^{c-1} l_{i_j} = -l_{i_c} + t$,
  $S \leftarrow S \cup \{\{i_1, \ldots, i_c\}\}$
  if $|S| = N$ then

    return $S$
  end if
  return $\perp$
```

Now we present the quantum version of the c-sum solver shown in Algorithm 3. In the algorithm, we adopt the classical memory with quantum random access to access classical data based on [Kup13].

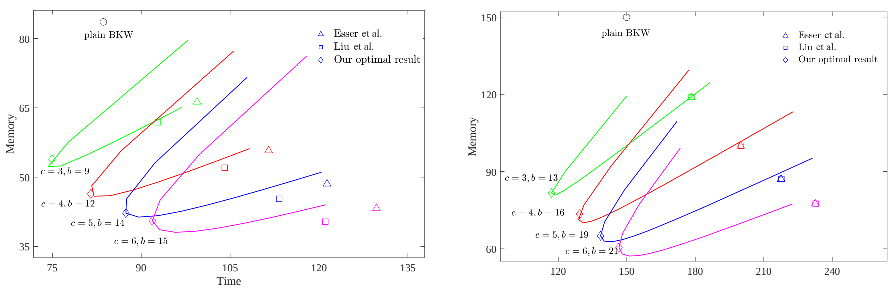
Correctness/sample complexity: In the algorithm, function f_t has domain size $|D| = \left|\binom{[\pm N]}{c-1}\right| \cdot \frac{1}{2} \approx \binom{N}{c-1} 2^{c-2} = \mathcal{O}(N^{c-1}2^{c-2})$ for we only need to search half of the space as discussed in Subsection 3.2. Since c is significantly smaller than N in LWE's setting, we omit the case that l_js are joint for simplicity. Therefore, we have $a = \frac{|f^{-1}(1)|}{|D|} = \sqrt{\frac{N}{N^{c-1}2^{c-2}}}$. Since Grover's search returns a target with probability $1-a$, we estimate that $\tilde{\mathcal{O}}(N \ln N) = \tilde{\mathcal{O}}(N)$ searches are required to find N valid $\{i_1, \ldots, i_c\}$.

Time/memory complexity: When L is sorted, the oracle $\hat{\mathcal{O}}$ has time complexity $\tilde{\mathcal{O}}(\log N) = \tilde{\mathcal{O}}(1)$. Therefore, finding a single-solution will need time complexity $\tilde{\mathcal{O}}(\sqrt{1/a}) = \tilde{\mathcal{O}}\left(\sqrt{\frac{|D|}{|f^{-1}(1)|}}\right) = \tilde{\mathcal{O}}\left(\sqrt{\frac{N^{c-1}2^{c-2}}{N}}\right) = \tilde{\mathcal{O}}\left(N^{\frac{c}{2}-1}2^{\frac{c}{2}-1}\right)$, and the total time complexity is $\tilde{\mathcal{O}}\left(N \cdot N^{\frac{c}{2}-1}2^{\frac{c}{2}-1}\right) = \tilde{\mathcal{O}}\left(N^{\frac{c}{2}}2^{\frac{c}{2}-1}\right)$. Similar to the classical c-sum method, the quantum algorithm consumes memory $\tilde{\mathcal{O}}(N)$.

Summarizing the result above, we describe the complexity of quantum-c-sum algorithm as follows.

Lemma 5. *Quantum-c-sum-q recovers a solution of c-SP$_b$ in time $\tilde{\mathcal{O}}(N^{\frac{c}{2}}2^{\frac{c}{2}-1})$ and memory $\tilde{\mathcal{O}}(N)$.*

In the structure of BKW algorithm, we can utilize the quantum-c-sum-q solver to



(a) $k = 50, \alpha = 0.03$ (b) $k = 100, \alpha = 0.01$

Figure 3: Quantum c-sum BKW time/memory tradeoff

construct a quantum version of c-sum-BKW^{LWE}. Similar to method in Subsection 3.2, we have the following result about the complexity of quantum BKW. The main complexity trade-off formula is presented in Theorem 4.

Theorem 4 (Quantum trade-off for c-sum BKW)**.** *For given $c \in \mathbb{N}$, $\epsilon > 0$, quantum c-sum-BKW^{LWE} solves the k-dimensional LWE instance in time $T = 2^{\vartheta(1+\epsilon)}$ and memory $M = 2^{\mu^{(1+\epsilon)}}$, where*

$$\vartheta = \log(k) + 2\pi^2\alpha^2 \log(e) \cdot c^{k/b} + \frac{c \log q}{2(c-1)} \cdot b,$$

$$\mu = \log(b) + 2\pi^2\alpha^2 \log(e) \cdot c^{k/b} + \log(q) \cdot \frac{b}{c-1}.$$

Proof. Similar to classical BKW in Theorem 1, the total run time for the algorithm is $\tilde{\mathcal{O}}(m \cdot a \cdot T_{c,N})$ for constant c. So the c-sum-BKW^{LWE} has time complexity $T = m \cdot q^{b(1+o(1))}$. Consider that for quantum-c-sum-q,

$$\begin{aligned} T_{c,N} &= \tilde{\mathcal{O}}(N^{\frac{c}{2}} 2^{\frac{c}{2}-1}) \\ &= \tilde{\mathcal{O}}(q^{\frac{\log_q \frac{q}{q-1} + c\log_q(c/2)+b}{c-1} \cdot \frac{c}{2}} \cdot 2^{\frac{c}{2}-1}) \\ &= \tilde{\mathcal{O}}(2^{\frac{c}{2(c-1)} \cdot \left(\log \frac{q}{q-1} + c\log(c/2)+b\right) \log q + \frac{c}{2} - 1}) \\ &= \tilde{\mathcal{O}}(2^{\frac{c \log q}{2(c-1)} \cdot b}), \end{aligned}$$

$$\begin{aligned} M_{c,N} = \tilde{\mathcal{O}}\left(N\right) &= \tilde{\mathcal{O}}(q^{\frac{\log_q \frac{q}{q-1} + c\log_q c + b}{c-1}}) \\ &= \tilde{\mathcal{O}}(2^{\frac{\log q}{c-1} \cdot b}), \end{aligned}$$

and $m = \tilde{\mathcal{O}}\left(b \cdot e^{\frac{2\pi^2\sigma^2 c^a}{q^2}}\right)$. Then, for stripe width $b = \frac{k}{a}$, we have time complexity $2^{\vartheta(1+\epsilon)}$, $\vartheta = \log(k) + \frac{2\pi^2\sigma^2 \log(e)}{q^2} \cdot c^{k/b} + \frac{c \log q}{2(c-1)} \cdot b$. Similarly, we estimate the memory cost. $\square$

The trade-off results and minimized time complexities are illustrated in Figure 3. By choosing proper parameters, the time cost is significantly lower than the classical model. Similar to the classical model, we can choose optimal parameters to ensure minimized complexities. In Corollary 2, we show the integer b values which minimize time complexities.

Corollary 2. *For given c, when b satisfies $2\pi^2\alpha^2 \log(c) \cdot c^{k/b} \cdot \frac{k}{b^2} = \frac{c \log(q)}{2(c-1)}$, c-sum BKW has minimized time complexity. When b satisfies $2\pi^2\alpha^2 \log(c) \cdot c^{k/b} \cdot \frac{k}{b^2} = \frac{1}{b\ln(2)} + \frac{\log(q)}{c-1}$, c-sum BKW has minimized memory complexity.*

4 Quantum Meet-in-the-Middle in BKW Algorithm

In the preceding section, we provide a new complexity analysis for basic c-sum BKW variants. While the time and memory costs of the algorithm has been reduced by choosing proper parameters, we still wish to find faster BKW variants, specifically in the quantum computation model. In this section, we improve on the existing BKW variants to design a quantum augmented BKW algorithm with Meet-in-the-Middle method as c-sum solver. Further, we utilize and adjust the complexity trade-off formula discussed in Section 3 to illustrate the new variant's advantage in time/memory cost. When optimal parameters are chosen, the new algorithm achives reduced time exponents compared with the basic quantum BKW.

4.1 MitM c-sum solver in BKW

We now present a Meet-in-the-Middle (MitM) method as BKW's c-sum solver, then discuss its analogue in the quantum computing model. Meet-in-the-Middle algorithms are discussed and used for attacking ECDSA [HGJ10] and LWE [GvVW17]. By generalizing the method, we derive the MitM c-sum solver in Algorithm 4, where input samples are partitioned into 4 lists. To find target vector $t \in \mathbb{F}_q^b$, an intermediate variable $\tau \in \mathbb{F}_q^\gamma$ is iterated. Then from two of the lists, we run Meet-in-the-Middle (MitM) subroutines to find two samples from each of them such that their last γ entries add up to τ. When we use low_γ to denote the last γ entries of a vector, we search for a sum t of $c/2$ vectors such that $\text{low}(t)_\gamma = \tau$. From the other two lists, $c/2$ samples are searched similarly so that their lower blocks add up to $\text{low}_\gamma(t) - \tau$.

Algorithm 4 Meet-in-the-Middle subroutine (MitM)

```
Require: sorted lists $\{L_i\}_{i=1}^2$ of samples, target $\tau \in \mathbb{F}_q^\gamma$, $\gamma \in [0,b]$, $c$ is a multiple of 4,
    precomputed $\text{low}_\gamma(l_{i_1} + \ldots + l_{i_{c/4}})$ for all $i_1, \ldots, i_{c/4} \in \pm L_1$
Ensure: list $S'$
  $S' \leftarrow \emptyset$
  for all $\nu = \{i_{c/4+1}, \ldots, i_{c/2}\} \in \binom{\pm L_2}{c/4}$ do
    for all precomputed $\text{low}_\gamma(l_{i_1} + \ldots + l_{i_{c/4}})$ satisfying $\tau = \text{low}_\gamma(l_{i_1} + \ldots + l_{i_{c/4}} + l_{i_{c/4+1}}$
    $+ \ldots + l_{i_{c/2}})$ do
      $S' \leftarrow S' \bigcup \{\{i_1, \ldots, i_{c/2}\}\}$
      return $S'$
    end for
  end for
```

Our c-sum solver in this section adopts the MitM algorithm as subroutines to reach better time/memory tradeoff. By running the MitM procedure with a intermediate target, we can select some possible solutions of the original problem, thus reducing the search space. The c-sum algorithm with MitM subroutine is described in Algorithm 4, 5. We now show the complexity of this algorithm in Lemma 6 and Corollary 3, then go on to analyze the cost of BKW algorithm with the new c-sum solver in Theorem 5.

Lemma 6. *For parameter γ, 4-list-c-sum-q algorithm recovers a solution of c-SP$_b$ in time $\tilde{\mathcal{O}}(\max\{q^{\frac{c}{2}\lambda}, q^{\lambda \cdot \frac{c}{4}+\gamma}\})$ and memory $\tilde{\mathcal{O}}(\max\{q^{\frac{c}{2}\lambda-\gamma},\ q^{\lambda\cdot\frac{c}{4}}, |S|\})$.*

Proof. The MitM steps of the algorithm find elements in L_1', L_2', where each MitM process has time/memory complexity $\tilde{\mathcal{O}}(2^{\frac{c}{4}} q^{\lambda\cdot\frac{c}{4}}) = \tilde{\mathcal{O}}(q^{\lambda\cdot\frac{c}{4}})$ for constant c. After the steps, we expect $\tilde{\mathcal{O}}((2^{\frac{c}{4}} q^{\lambda\cdot\frac{c}{4}})^2/q^\gamma) = \tilde{\mathcal{O}}(2^{\frac{c}{2}} q^{\frac{c}{2}\lambda-\gamma})$ elements are stored in L_1', L_2'. The final search for $(l_1', l_2') \in L_1' \times L_2'$ line takes time $\tilde{\mathcal{O}}(2^{\frac{c}{2}} q^{\frac{c}{2}\lambda-\gamma})$ for sorted L_1', L_2'. Since the first line iterates

Algorithm 5 4-list-c-sum-q

```
Require: sorted lists $\{L_i\}_{i=1}^4$ of $q^\lambda/4$ samples each, target vector $t \in \mathbb{F}_q^b$, parameter $\gamma \in [0, b]$, $c$ is a multiple of 4
Ensure: list $S$
  for all $\tau \in \mathbb{F}_q^\gamma$ do
    use MitM, look for $l_{i_1}, \dots, l_{i_{c/4}} \in \pm L_1$ and $l_{i_{c/4+1}}, \dots, l_{i_{c/2}} \in \pm L_2$ such that $\mathrm{low}_\gamma\left(\sum_{j=1}^{c/4} l_{i_j} + \sum_{j=c/4+1}^{c/2} l_{i_j}\right) = \tau$; store $l_{i_1}, \dots, l_{i_{c/2}}$ in a sorted list $L'_1$
    use MitM, look for $l_{i_{c/2}}, \dots, l_{i_{3c/4}} \in \pm L_3$ and $l_{i_{3c/4+1}}, \dots, l_{i_c} \in \pm L_4$ such that $\mathrm{low}_\gamma\left(\sum_{j=1+c/2}^{3c/4} l_{i_j} + \sum_{j=3c/4+1}^{c} l_{i_j}\right) = \mathrm{low}_\gamma(t) - \tau$; store $l_{i_{c/2+1}}, \dots, l_{i_c}$ in a sorted list $L'_2$
    look for $(l'_1, l'_2) \in L'_1 \times L'_2$ such that $l'_1 + l'_2 = t$, recover indices $(i_1, \dots, i_c)$
    $S \leftarrow S \cup \{\{i_1, \dots, i_c\}\}$
    if $|S| = 4q^\lambda$ then
      return $S$
    end if
  end for
  return $\emptyset$
```

τ over $\mathbb{F}_q^\gamma$, the total time complexity is $\tilde{\mathcal{O}}(\max\{q^{\frac{c}{2}\lambda}, q^{\lambda\cdot\frac{c}{4}+\gamma}\})$, and the memory complexity is $\tilde{\mathcal{O}}(\max\{q^{\frac{c}{2}\lambda-\gamma}, q^{\lambda\cdot\frac{c}{4}}, |S|\})$. $\square$

Corollary 3. *When parameter $\gamma = \frac{c}{4}\cdot\lambda$, 4-list-c-sum-q has minimal time cost $\tilde{\mathcal{O}}(q^{\frac{c}{2}\lambda})$ and memory $\tilde{\mathcal{O}}(\max\{q^{\lambda\cdot\frac{c}{4}}, |S|\})$.*

Adopting Algorithm 5 as the c-sum solver, we get the BKW algorithm with MitM techniques, denoted as MitM-BKW^{LWE}. In Theorem 5, we will analyze the cost of the algorithm in the classical model.

Theorem 5 (Classical trade-off for MitM-BKW^{LWE})**.** *For given $c \in \mathbb{N}$, $\epsilon > 0$, and parameter $\gamma = \frac{c}{4}\cdot\lambda$, MitM-BKW^{LWE} solves the k-dimensional LWE instance in time $T = 2^{\vartheta(1+\epsilon)}$ and memory $M = 2^{\mu^{(1+\epsilon)}}$, where $\vartheta = \log(k) + 2\pi^2\alpha^2\log(e)\cdot c^{k/b} + \frac{c\log(q)}{2(c-1)}\cdot b, \mu = \log(b) + 2\pi^2\alpha^2\log(e)\cdot c^{k/b} + \frac{c\log(q)}{4(c-1)}\cdot b$.*

Proof. Similar to basic BKW, sample number $N = q^{\frac{\log_q(\frac{q}{q-1}) + c\log_q(c/2) + b}{c-1}}$ is enough to recover the secret key. The 4-list method partitions the input samples into four lists of size q^λ. In this case,

$$\lambda = \frac{\log_q(\frac{q}{q-1}) + c\log_q(c/2) + b}{c-1} - \log_q(4).$$

According to Corollary 3, we set parameter $\gamma = \frac{c}{4}\cdot\lambda$ to minimize the run time. Since $\frac{c}{4}\cdot\lambda < b$, this setting is feasible. Thus, the time cost $T_{c,N}$ of 4-list-c-sum-q is computed as

$$\begin{aligned}\tilde{\mathcal{O}}(q^{\frac{c}{2}\cdot\lambda}) &= \tilde{\mathcal{O}}\left(q^{\frac{c}{2(c-1)}\cdot\left(\log_q(\frac{q}{q-1}) + c\log_q(c/2) + b\right) - \frac{c}{2}\log_q 2}\right)\\ &= \tilde{\mathcal{O}}\left(q^{\frac{c}{2(c-1)}\cdot b}\right)\end{aligned}$$

for constant c. The run time of MitM-BKW^{LWE} is

$$\begin{aligned}T &= \tilde{\mathcal{O}}(m \cdot a \cdot T_{c,N})\\ &= 2^{\vartheta(1+\epsilon)},\end{aligned}$$

with $\vartheta = \log(k) + \frac{2\pi^2\sigma^2\log(e)}{q^2} \cdot c^{k/b} + \frac{c\log(q)}{2(c-1)} \cdot b$. Lemma 6 states the memory cost of 4-list-c-sum-q as $M_{c,N} = \tilde{\mathcal{O}}(\max\{q^{\lambda\cdot\frac{c}{4}}, |S|\})$. Under the Independence Heuristic, $\mathbb{E}[|S|] = \left(\frac{N}{c}\right)^c \cdot q^{-b} = \tilde{\mathcal{O}}(q^{\frac{cb}{c-1}-b}) = \tilde{\mathcal{O}}(q^{\frac{b}{c-1}})$, and $q^{\frac{c}{4}\lambda} = \tilde{\mathcal{O}}\left(q^{\frac{c}{4(c-1)}\cdot b}\right)$. In the 4-sum algorithms, $c \geq 4$, so the memory $M = m \cdot q^{\frac{c}{4(c-1)}\cdot b} = \tilde{\mathcal{O}}\left(b \cdot e^{\frac{2\pi^2\sigma^2 c^a}{q^2}} \cdot q^{\frac{c}{4(c-1)}\cdot b}\right) = 2^{\mu(1+\epsilon)}$, where $\mu = \log(b) + \frac{2\pi^2\sigma^2\log(e)}{q^2} \cdot c^{k/b} + \frac{c\log(q)}{4(c-1)} \cdot b$. □

4.2 Quantum MitM-BKW algorithm

Based on the classical variant discussed above, we combine quantum speed-up techniques with the 4-list-c-sum-q algorithm. The quantum variant of the MitM-based c-sum solver is presented in Algorithm 6. This quantum MitM construction involves the technique of utilizing Grover search as subroutines in loops, which is first discussed in [NPS20]. In our algorithm, we set parameter $\gamma = b$, so finding one possible $(i_1, \ldots, i_{c/4})$ and $(i_{c/2+1}, \ldots, i_{3c/4})$ is enough to recover a single solution. In our method, we first precomputed the sorted sums $l_{i_{c/4+1}} + \ldots + l_{i_{c/2}}$, $l_{i_{3c/4+1}} + \ldots + l_{i_c}$. To find the indices $(i_1, \ldots, i_{c/4})$, we define

$$f_t : \binom{[\pm q^\lambda]}{c/4-1} \to \{0,1\}, \nu \longmapsto \begin{cases} 1 & \exists i_{c/4} \in [\pm q^\lambda] \backslash \nu : \displaystyle\sum_{j=1}^{c/4-1} l_{i_j} = -l_{i_{c/4}} + t, \\ 0 & else, \end{cases}$$

for some $t = \tau - \sum_{j=c/4+1}^{c/2} l_{i_j}$ with precomputed $\sum_{j=c/4+1}^{c/2} l_{i_j}$, and use Grover's search to find the pre-image of 1. Indices $(i_{c/2+1}, \ldots, i_{3c/4})$ can be found similarly. For f_t, domain size $|D| = \left|\binom{[\pm q^\lambda]}{c/4-1}\right| = \binom{q^\lambda}{c/4-1} 2^{c/4-1} \leq q^{\lambda(c/4-1)} 2^{c/4-1}$. Therefore, finding a collision takes time $\tilde{\mathcal{O}}\left(\sqrt{|D|}\right) = \tilde{\mathcal{O}}\left(\sqrt{q^{\lambda(\frac{c}{4}-1)}2^{\frac{c}{4}-1}}\right) = \tilde{\mathcal{O}}\left(q^{\lambda(\frac{c}{8}-\frac{1}{2})}\right)$. Then finding $\tilde{\mathcal{O}}(q^\lambda)$ single-solutions costs time $\tilde{\mathcal{O}}(q^\lambda) \cdot \tilde{\mathcal{O}}\left(\sqrt{|D|}\right) = \tilde{\mathcal{O}}\left(q^{\lambda(\frac{c}{8}+\frac{1}{2})}\right)$. In the MitM subroutine, $\left|\binom{[\pm q^\lambda]}{c/4}\right| = \tilde{\mathcal{O}}(q^{\lambda\cdot\frac{c}{4}})$ combinations are precomputed. Therefore, the memory cost is $\tilde{\mathcal{O}}(\max\{q^{\lambda\cdot\frac{c}{4}}, |S|\})$.

Summarizing the results, we obtain a theoretical analysis of quantum-4-list-c-sum-q algorithm's cost in Lemma 7.

Lemma 7. *Quantum-4-list-c-sum-q can recover a solution of c-SP$_b$ in expected time* $\tilde{\mathcal{O}}\left(q^{\lambda(\frac{c}{8}+\frac{1}{2})}\right)$ *and memory* $\tilde{\mathcal{O}}(\max\{q^{\lambda\cdot\frac{c}{4}}\}, |S|)$.

We let quantum MitM-BKW^{LWE} denote the BKW variant where Quantum-4-list-c-sum-q is instantiated as the c-sum subroutine. Based on Lemma 7, we estimate the time/memory of the algorithm in Theorem 6.

Theorem 6 (Quantum trade-off for MitM-BKW^{LWE})**.** *For given* $c \in \mathbb{N}$, $\epsilon > 0$, *and a sufficiently large k, quantum MitM-BKW^{LWE} solves the LWE instance in time* $T = 2^{\vartheta(1+\epsilon)}$ *and memory* $M = 2^{\mu(1+\epsilon)}$, *where* $\vartheta = \log(k) + 2\pi^2\alpha^2\log(e) \cdot c^{k/b} + \frac{(c+4)\log(q)}{8(c-1)} \cdot b, \mu = \log(b) + 2\pi^2\alpha^2\log(e) \cdot c^{k/b} + \frac{c\log(q)}{4(c-1)} \cdot b$.

Proof. Following the classical model, we set $N = q^{\frac{\log_q(\frac{q}{q-1})+c\log_q(c/2)+b}{c-1}}$, and $\lambda = \frac{1}{c-1}(\log_q(\frac{q}{q-1}) + c\log_q(c/2) + b) - \log_q(4)$. By Lemma 7, the time cost $T_{c,N}$ of quantum 4-list-c-sum-q is

Algorithm 6 Quantum-4-list-c-sum-q

```
Require: sorted lists $\{L_i\}_{i=1}^4$ of $q^\lambda$ samples in total, target vector $t \in \mathbb{F}_q^b$, c is a multiple of 4
Ensure: list $S$ or $\perp$
  create QRAM $O_{W_1}, \ldots, O_{W_4}$
  for all $l_i$ in $\pm L_1$ do
    add $l_i$ to $O_{W_1}$ at index $i$
  end for
  for all $(l_{i_{c/4+1}}, \ldots, l_{i_{c/2}})$ in $\pm L_2$ do
    add $\sum_{j=c/4+1}^{c/2} l_{i_j}$ to $O_{W_2}$ at index $(i_{c/4+1}, \ldots, i_{c/2})$
  end for
  for all $l_i$ in $\pm L_3$ do
    add $l_i$ to $O_{W_3}$ at index $i$
  end for
  for all $(l_{i_{3c/4+1}}, \ldots, l_{i_c})$ in $\pm L_4$ do
    add $\sum_{j=3c/4+1}^{c} l_{i_j}$ to $O_{W_4}$ at index $(i_{3c/4+1}, \ldots, i_c)$
  end for

  create oracle $\mathcal{O}_1(i_1, \ldots, i_{c/4})$
  get $l_{i_1}, \ldots, l_{i_{c/4}}$ from $O_W$
  if exists $i_{c/4+1}, \ldots, i_{c/2}$ such that $\sum_{j=1}^{c/4} l_{i_j} + \sum_{j=c/4+1}^{c/2} l_{i_j} = \tau$ then
    return 1
  end if
  end $\mathcal{O}_1(i_1, \ldots, i_{c/4})$

  create oracle $\mathcal{O}_2(i_{c/2+1}, \ldots, i_{3c/4})$
  get $l_{i_{c/2+1}}, \ldots, l_{i_{3c/4}}$ from $O_W$
  if exists $i_{3c/4+1}, \ldots, i_c$ such that $\sum_{j=c/2+1}^{3c/4} l_{i_j} + \sum_{j=3c/4+1}^{c} l_{i_j} = t - \tau$ then
    return 1
  end if
  end $\mathcal{O}_2$

  create oracle $\hat{\mathcal{O}}(\tau)$:
  use Grover's search to find $(i_1, \ldots, i_{c/4})$, $(i_{c/2+1}, \ldots, i_{3c/4})$ such that $\mathcal{O}_1(i_{3c/4+1}, \ldots, i_{c/2}) = 1$, $\mathcal{O}_2(i_{c/2+1}, \ldots, i_{3c/4}) = 1$
  if $(i_1, \ldots, i_{c/4}), (i_{c/2+1}, \ldots, i_{3c/4})$ exist then
    return 1
  end if
  end oracle $\hat{\mathcal{O}}(\tau)$

  use Grover's search to find $\tau$ such that $\hat{\mathcal{O}}(\tau) = 1$
  for all $(i_1, \ldots, i_{c/4}), (i_{c/2+1}, \ldots, i_{3c/4})$ satisfying $\mathcal{O}_1(i_1, \ldots, i_{c/4}) = 1$, find $(i_{c/4+1}, \ldots, i_{c/2}), (i_{3c/4+1}, \ldots, i_c)$ from precomputed data, $S \leftarrow S \cup \{\{i_1, \ldots, i_c\}\}$
  if $|S| = 4q^\lambda$ then
    return $S$
  end if
  return $\perp$
```

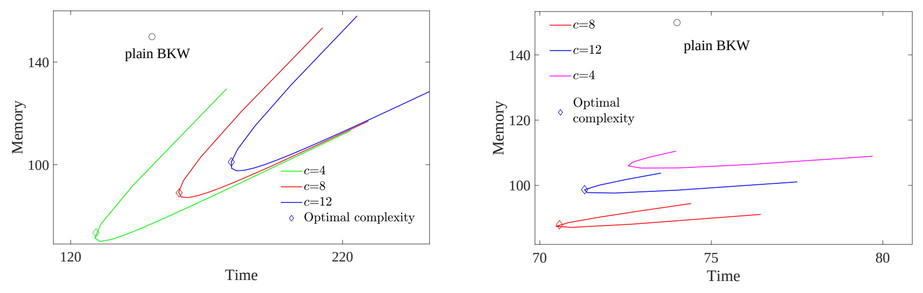
$\tilde{\mathcal{O}}(q^{(\frac{c}{8}+\frac{1}{2})\cdot\lambda}) = \tilde{\mathcal{O}}\left(q^{\frac{c+4}{8(c-1)}\cdot(\log_q(\frac{q}{q-1})+c\log_q(c/2)+b)-\frac{c+4}{4}\log_q 2}\right) = \tilde{\mathcal{O}}\left(q^{\frac{c+4}{8(c-1)}\cdot b}\right)$ for constant c. The run time of MitM-BKW^{LWE} is

$$\tilde{\mathcal{O}}(m \cdot a \cdot T_{c,N}) = \tilde{\mathcal{O}}\left(k \cdot e^{\frac{2\pi^2\sigma^2 c^a}{q^2}} \cdot q^{\frac{c+4}{8(c-1)}\cdot b}\right) = 2^{\vartheta(1+\epsilon)},$$

with $\vartheta = \log(k) + \frac{2\pi^2\sigma^2\log(e)}{q^2} \cdot c^{k/b} + \frac{(c+4)\log(q)}{8(c-1)} \cdot b$. Since $\mathbb{E}[|S|] = \tilde{\mathcal{O}}(q^{\frac{b}{c-1}})$ and $q^{\frac{c}{4}\lambda} = \tilde{\mathcal{O}}\left(q^{\frac{c}{4(c-1)}\cdot b}\right)$, the memory complexity is $M = m \cdot q^{\frac{c}{4(c-1)}\cdot b} = \tilde{\mathcal{O}}\left(b \cdot e^{\frac{2\pi^2\sigma^2 c^a}{q^2}} \cdot q^{\frac{c}{4(c-1)}\cdot b}\right) = 2^{\mu(1+\epsilon)}$, where $\mu = \log(b) + \frac{2\pi^2\sigma^2\log(e)}{q^2} \cdot c^{k/b} + \frac{c\log(q)}{4(c-1)} \cdot b$. □

According to Theorem 6, optimal b can be chosen to minimize time/memory costs similar with the basic c-sum BKW. The values of these parameters are presented in Corollary 4. In practice, the parameter b should be found via integer programming since it must be an integer. As is shown in Corollary 4, the MitM-BKW that we propose can further benefit from selecting optimal parameters, as is illustrated in Figure 4. In the figure, the complexity results with different b values form a list of curves, then we can adjust the value of integer b to obtain an optimistic time/memory balances, i.e. the points nearest to the origin point.

Corollary 4. *For given c, when b satisfies $2\pi^2\alpha^2\log(c)\cdot c^{k/b}\cdot\frac{k}{b^2} = \frac{(c+4)\log(q)}{8(c-1)}$, MitM-BKW has minimized time complexity. When b satisfies $2\pi^2\alpha^2\log(c)\cdot c^{k/b}\cdot\frac{k}{b^2} = \frac{1}{b\ln(2)} + \frac{\log(q)}{4(c-1)}$, MitM-BKW has minimized memory complexity.*



(a) classical (b) quantum

Figure 4: MitM-BKW time/memory tradeoff, $k = 100, \alpha = 0.01$

4.3 Performance evaluation and application

Based on the analysis above, we now evaluate the performance of BKW variants in various LWE instances. We first summarize the time complexities $T = 2^{\vartheta}$ and memory $M = 2^{\mu}$ of our algorithms in Table 1. While previous works provide complexity results by assuming specific b values, our analysis introduces stripe size b into the trade-off formula. Through this model, we reach proper time/memory balances and minimize the computation costs of BKW.

To give an example, we apply the models to some simple LWE instances with parameters $k = 50, \alpha = 0.03$ and $k = 100, \alpha = 0.01$. Based on the optimal parameters discussed earlier, we compare the exponents ϑ, μ under optimal parameters for specific LWE instances in Table 2. Under our trade-off model, the complexity exponents of c-sum BKW and MitM BKW are significantly reduced compared with previous results. On the other hand, the plain BKW is still faster, but its memory cost is dramatically higher than other algorithms. Therefore, the c-sum BKW variants are still better in terms of time/memory trade-off.

Table 1: Complexity exponents of our BKW variants and previous results

	Algorithm	ϑ	μ
Classical BKW	c-sum BKW	$\log(k) + 2\pi^2\alpha^2 \log(e) \cdot c^{k/b} + \log(q) \cdot b$	$\log(b) + 2\pi^2\alpha^2 \log(e) \cdot c^{k/b} + \log(q) \cdot \frac{b}{c-1}$
	MitM BKW	$\log(k) + 2\pi^2\alpha^2 \log(e) \cdot c^{k/b} + \frac{c\log(q)}{2(c-1)} \cdot b$	$\log(b) + 2\pi^2\alpha^2 \log(e) \cdot c^{k/b} + \frac{c\log(q)}{4(c-1)} \cdot b$
	Esser et al. [EHK+18]	$\frac{\log(c)k}{2} \cdot \frac{\log_k(q)}{\log_k(q)-\log_k(\sigma)+1/2}$	$\frac{\log(c)k}{2(c-1)} \cdot \frac{\log_k(q)}{\log_k(q)-\log_k(\sigma)+1/2}$
	Liu et al. [LY22]	$\frac{\log(c)k\log(q)(1+\epsilon)}{\log(k)+2\log(q)-2\log(\sigma)}$	$\frac{1}{c-1} \cdot \frac{\log(c)k\log(q)(1+\epsilon)}{\log(k)+2\log(q)-2\log(\sigma)}$
Quantum BKW	Basic quantum BKW	$\log(k) + 2\pi^2\alpha^2 \log(e) \cdot c^{k/b} + \frac{c\log q}{2(c-1)} \cdot b$	$\log(b) + 2\pi^2\alpha^2 \log(e) \cdot c^{k/b} + \log(q) \cdot \frac{b}{c-1}$
	MitM BKW	$\log(k) + 2\pi^2\alpha^2 \log(e) \cdot c^{k/b} + \frac{(c+4)\log(q)}{8(c-1)} \cdot b$	$\log(b) + 2\pi^2\alpha^2 \log(e) \cdot c^{k/b} + \frac{c\log(q)}{4(c-1)} \cdot b$
	Esser et al. [EHK+18]	$\frac{c\log(c)}{4(c-1)} \cdot \frac{k\log_k(q)}{\log_k(q)-\log_k(\sigma)+1/2}$	$\frac{\log(c)k}{2(c-1)} \cdot \frac{\log_k(q)}{\log_k(q)-\log_k(\sigma)+1/2}$
	Liu et al. [LY22]	$\frac{c\log(c)}{2(c-1)} \cdot \frac{k\log(q)}{\log(k)+2\log(q)-2\log(\sigma)}$	$\frac{1}{c-1} \cdot \frac{\log(c)k\log(q)}{\log(k)+2\log(q)-2\log(\sigma)}$

Table 2: Optimal parameters by new trade-off formula

	Algorithm	ϑ	μ		Algorithm	ϑ	μ
$k = 50, \alpha = 0.03, c = 4$							
Classical BKW	c-sum BKW	119.3	45.8	Quantum BKW	c-sum BKW	83.5	45.8
	MitM BKW	83.5	45.7		MitM BKW	48.5	46.6
	Esser et al. [EHK+18]	156.1	52.0		Esser et al. [EHK+18]	104.1	52.0
	Liu et al. [LY22]	167.2	55.7		Liu et al. [LY22]	111.5	55.7
	plain BKW	83.6	83.6		plain BKW	41.8	83.6
$k = 100, \alpha = 0.01, c = 4$							
Classical BKW	c-sum BKW	186.0	70.4	Quantum BKW	c-sum BKW	132.6	70.2
	MitM BKW	132.6	70.2		MitM BKW	74.0	71.6
	Esser et al. [EHK+18]	300.1	100.0		Esser et al. [EHK+18]	200.1	100.0
	Liu et al. [LY22]	299.8	99.9		Liu et al. [LY22]	199.9	100.0
	plain BKW	175.8	175.8		plain BKW	87.8	175.8

As a further application of the c-sum BKW complexity model, we use it to evaluate the concrete security of TU Darmstadt LWE challenge instances and cryptographic LWE instances, then compare our results with BKW variants (Coded BKW [GJS15], Coded BKW with sieving [GJMS17], FWHT BKW [BGJ+21]) and lattice-based attacks. The complexities of previous algorithms are obtained by the lattice estimator [NPS20] and sage [Sag]. Table 3 shows the complexity exponents, where each entry in the table represents the complexity exponent computed following our analysis. From the results, the MitM-BKW has lower time complexity compared with most BKW variants. In most cases, the

MitM-BKW is faster than the dual lattice attack, although still slightly slower than the primal lattice attack. Especially, the advantage of MitM-BKW is more significant for larger α, with time exponent only 25% of c-sum BKW on average. Compared with the fastest previous variant, FWHT BKW, our algorithm is faster for smaller α. When $\alpha = 0.005$, MitM-BKW's time exponent is about 80% of FWHT BKW.

Table 3: Security estimations of the TU Darmstadt LWE challenge

Parameters			BKW algorithms ($c = 16$)							lattice algorithms	
k	q	α	BKW	coded BKW	sieve-coded BKW	FWHT BKW	c-sum BKW	dissect BKW	MitM-BKW	primal	dual
40	1601	0.005	40.6	42.6	41.5	34.4	45.8	33.8	28.6	31.4	37.1
		0.01	43.5	43.7	42.7	39.3	65.8	52.9	35.0	31.7	37.6
		0.015	46.5	52.6	44.1	42.4	100.2	86.4	40.5	32.0	41.4
		0.02	49.0	52.6	49.1	46.2	149.6	135.1	45.5	32.4	43.8
		0.025	52.6	52.7	49.2	48.3	221.4	205.7	51.6	34.3	48.1
		0.03	52.6	52.7	50.4	50.0	221.4	205.7	54.0	37.9	53.4
45	2027	0.005	44.1	55.2	45.0	37.7	46.7	33.4	30.9	31.6	37.4
		0.01	48.8	55.2	45.3	43.5	60.2	45.5	39.2	32.0	39.9
		0.015	51.8	55.2	54.7	48.3	78.9	63.4	45.0	32.5	43.9
		0.02	55.4	55.2	54.8	51.2	107.0	90.4	51.3	35.7	50.7
		0.025	55.7	55.3	54.8	54.1	136.8	120.1	54.2	39.9	57.1
		0.03	59.7	64.1	63.3	56.3	184.9	166.9	61.2	44.2	64.1
50	2503	0.005	48.9	46.4	45.5	41.8	51.7	37.1	33.4	31.9	37.6
		0.01	54.4	56.0	53.3	48.7	67.1	50.7	43.1	32.5	42.5
		0.015	57.8	56.8	53.4	52.5	88.1	70.8	50.2	35.5	50.5
		0.02	61.9	61.9	60.4	56.4	119.6	101.1	57.2	41.0	59.0
		0.025	61.9	66.1	61.7	59.3	152.1	133.5	60.2	46.4	65.4
		0.03	66.9	66.3	65.6	63.3	207.2	187.1	68.5	51.4	75.5
70	4903	0.005	66.2	62.3	62.2	62.3	83.1	63.1	51.3	34.6	49.5
		0.01	72.9	67.1	70.6	73.7	147.2	125.2	63.7	47.1	66.3
		0.015	77.5	73.3	72.7	75.6	255.0	231.6	73.6	57.4	81.2
120	14401	0.005	113.4	110.5	108.8	100.1	140.6	105.9	85.2	70.3	98.4
		0.01	123.1	124.0	115.8	115.1	240.9	203.4	105.6	86.5	106.1
		0.015	130.9	136.8	135.3	127.0	412.9	372.9	121.6	98.3	133.3

We next estimate the security of instances from real cryptosystems to present the efficiency of different BKW variants. In particular, we compare the performance of our BKW with other algorithms on LWE instances proposed by Regev [Reg05] and Lindner & Peikert [LP11]. Generally, the time cost of our algorithm is close to the best previous results, significantly faster than c-sum BKW and dissect BKW. Considering that MitM-BKW requires less memory consumption compared with original BKW and coded BKW, this comparison is able to demonstrate the improvement of MitM-BKW. The detailed comparison is shown in Table 4. On average, the time exponent of MitM-BKW is 98% of the fastest previous algorithm, FWHT BKW.

Table 4: Security estimation of LWE instances from Regev's and Lindner-Peikert's cryptosystems

Parameters			BKW algorithms ($c = 16$)							lattice-based algorithms	
k	q	σ	BKW	coded BKW	sieve-coded BKW	FWHT BKW	c-sum BKW	dissect BKW	MitM-BKW	primal	dual
Regev's parameters											
128	16411	11.81	107.5	84.5	84.2	59.2	111.6	78.9	70.8	57.3	69.2
256	65537	25.53	200.8	145.1	130	107	204.1	142.4	126.9	103.6	121
512	262147	57.06	384.6	287.6	247.6	243.3	390.2	271.3	242.6	201.6	231.2
Lindner-Peikert's parameters											
128	2503	2.7	95.7	69.7	69.2	48.8	97.7	68.6	60.2	53.4	67.5
256	4099	3.34	167.9	123.8	112.9	98.5	171.7	120.2	108.4	95.2	112.3
512	4099	2.9	308	209.2	197.3	243.3	323.5	228.7	211.7	179	207.8

4.4 Discussion: Solving c-sum via collision search

As an extension of the quantum c-sum BKW, we are also interested in another possible algorithmic technique – using quantum collision search methods as c-sum solvers, and implementing them as subroutines in BKW-type algorithms. The reduction from c-sum BKW's sample reduction phase to an instance of collision search is natural: When we aim to find c LWE samples which add to 0^b on a b-size stripe, we first define a $\frac{c}{2}$-sum function $f(i_1, i_2, \ldots, i_{c/2}) = l_{i_1} + l_{i_2} + \ldots + l_{i_{c/2}} \mod q$ with domain $X = \binom{[\pm N]}{c/2}$. Then, we try to find a collision $f(x) = f(y)$, where $x = (i_1, i_2, \ldots, i_{c/2}), y = (i_{c/2+1}, i_{c/2+2}, \ldots, i_c)$. It follows that $f(x) + f(-y) = 0^b$, i.e. $l_{i_1} + \ldots + l_{i_{c/2}} + (-l_{i_{c/2+1}}) + \ldots + (-l_{i_c}) = l_{i_1} + \ldots + l_{i_{c/2}} + l_{-i_{c/2}-1} + \ldots + l_{-i_c} = 0^b \mod q$. In this way, c samples are found to form a new LWE sample with zero stripe.

Obviously, the essential task for implementing such an algorithm lies in efficiently solving the multiple collision search instance $f(x) = f(y)$, with f's domain $X = \binom{[\pm N]}{c/2}$, range size $\mathbb{Z}_q^b$, number of required collisions N. By the theoretical results of Liu et al. [LZ19], we estimate its query complexity as $\mathcal{O}(q^{\frac{b}{3}} \cdot N^{\frac{2}{3}})$. Some possible quantum collision search methods include the Grover-based BHT collision search algorithm [BHT98], and the quantum random walk based method [BCSS23]. While the collision search method can accelerate the sample reduction, a possible challenge is that by defining $\frac{c}{2}$-sum function f, the search space is separated. Therefore, the algorithm may require a larger N to generate enough collisions, so the practical performance of collision-BKW is a more complicated matter. In future work, we will further discuss the implementation and complexity of BKW variants with collision search.

5 Conclusion

While the existing c-sum BKW variants have already benefited from previous works, there is still a need for detailed complexity analysis and optimal parameter selection to solve LWE instances. In this paper, we develop a generic complexity trade-off formula for c-sum BKW for both classical and quantum variants. This trade-off is built on analyzing the costs of c-sum solver subroutines, and can guide us in choosing the best parameters to optimize the performance of popular BKW algorithms. Through the analysis, we offer proper parameters to reduce the time and memory complexities of BKW compared to previous results. Compared with works of Esser et al. and Liu et al., our analysis achieves lower complexity bounds thanks to optimal parameters. To further improve its efficiency, we propose a quantum augmented variant with lower complexity than the original quantum c-sum BKW. This algorithm extends the structure of the basic c-sum BKW by utilizing a quantum MitM subroutine as the c-sum solver. By exploiting our trade-off analysis method, we estimate the complexity of this algorithm. Under the trade-off formula proposed in this paper, our method reaches a lower cost than previous works, and allows achieving new time/memory balances by adjusting the parameters.

References

[ACF+15] Martin Albrecht, Carlos Cid, Jean-Charles Faugere, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the BKW algorithm on LWE. *Designs, Codes and Cryptography*, 74, 02 2015. doi:10.1007/s10623-013-9864-x.

[AFFP14] Martin R. Albrecht, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. Lazy modulus switching for the BKW algorithm on LWE. In Hugo Krawczyk, editor, *Public-Key Cryptography – PKC 2014*, pages 429–445, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. doi:10.1007/978-3-642-54631-0_25.

[BCSS23] Xavier Bonnetain, André Chailloux, André Schrottenloher, and Yixin Shen. Finding many collisions via reusable quantum walks. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 221–251, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-30589-4_8.

[BGJ$^+$21] Alessandro Budroni, Qian Guo, Thomas Johansson, Erik Mårtensson, and Paul Stankovski Wagner. Improvements on making BKW practical for solving LWE. *Cryptography*, 5(4), 2021. doi:10.3390/cryptography5040031.

[BHMT02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplication and estimation. *Contemporary Mathematics*, 305:53–74, 2002. doi:10.1090/conm/305.

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In Cláudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN'98: Theoretical Informatics*, pages 163–169, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. doi:10.1007/BFb0054319.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of ACM*, 50(4):506–519, jul 2003. doi:10.1145/792538.792543.

[BTV16] Sonia Bogos, Florian Tramér, and Serge Vaudenay. On solving LPN using BKW and variants. *Cryptography and Communications*, 8:331–369, 2016. doi:10.1007/s12095-015-0149-2.

[Cry06] Mary Cryan. Probability and computing: Randomized algorithms and probabilistic analysis. *Bulletin of Symbolic Logic*, 12(2):304–308, 2006. doi:10.1017/S107989860000278X.

[DTV15] Alexandre Duc, Florian Tramèr, and Serge Vaudenay. Better algorithms for LWE and LWR. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 173–202, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. doi:10.1007/978-3-662-46800-5_8.

[EHK$^+$18] Andre Esser, Felix Heuer, Robert Kübler, Alexander May, and Christian Sohler. Dissection-BKW. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 638–666, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-96881-0_22.

[GJL14] Qian Guo, Thomas Johansson, and Carl Löndahl. Solving LPN using covering codes. In Tetsu Iwata and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2014*, volume 8873, pages 1–20, Germany, 2014. Springer. doi:10.1007/s00145-019-09338-8.

[GJMS17] Qian Guo, Thomas Johansson, Erik Mårtensson, and Paul Stankovski. Coded-BKW with sieving. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 323–346, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-70694-8_12.

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 23–42, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. doi:10.1007/978-3-662-47989-6_2.

[GvVW17] Florian Göpfert, Christine van Vredendaal, and Thomas Wunderer. A hybrid lattice basis reduction and quantum search attack on LWE. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography*, pages 184–202, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-59879-6_11.

[HGJ10] Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 235–256, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:10.1007/978-3-642-13190-5_12.

[KF15] Paul Kirchner and Pierre-Alain Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 43–62, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. doi:10.1007/978-3-662-47989-6_3.

[KP20] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Phys. Rev. A*, 101:022316, Feb 2020. doi:10.1103/PhysRevA.101.022316.

[Kup13] Greg Kuperberg. Another subexponential-time quantum algorithm for the dihedral Hidden Subgroup Problem. In Simone Severini and Fernando Brandao, editors, *8th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2013)*, volume 22 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20–34, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TQC.2013.20.

[LF06] Éric Levieil and Pierre-Alain Fouque. An improved LPN algorithm. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks*, pages 348–359, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. doi:10.1007/11832072_24.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *Topics in Cryptology – CT-RSA 2011*, pages 319–339, Berlin, Heidelberg, 2011. Springer. doi:10.1007/978-3-642-19074-2_21.

[LY22] Hanlin Liu and Yu Yu. A non-heuristic approach to time-space tradeoffs and optimizations for BKW. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022*, pages 741–770, Cham, 2022. Springer Nature Switzerland. doi:10.1007/978-3-031-22969-5_25.

[LZ19] Qipeng Liu and Mark Zhandry. On finding quantum multi-collisions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 189–218, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-17659-4_7.

[NPS20] María Naya-Plasencia and André Schrottenloher. Optimal merging in quantum k-xor and k-sum algorithms. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 311–340, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-45724-2_11.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC 2005, pages 84–93, New York, NY, USA, 2005. Association for Computing Machinery. doi:10.1145/1060590.1060603.

[Sag] SageMath. Accessed on June 10, 2024. URL: https://www.sagemath.org/index.html.

[ZJW16] Bin Zhang, Lin Jiao, and Mingsheng Wang. Faster algorithms for solving LPN. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 168–195, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. doi:10.1007/978-3-662-49890-3_7.

A Evaluation of c-sum solver

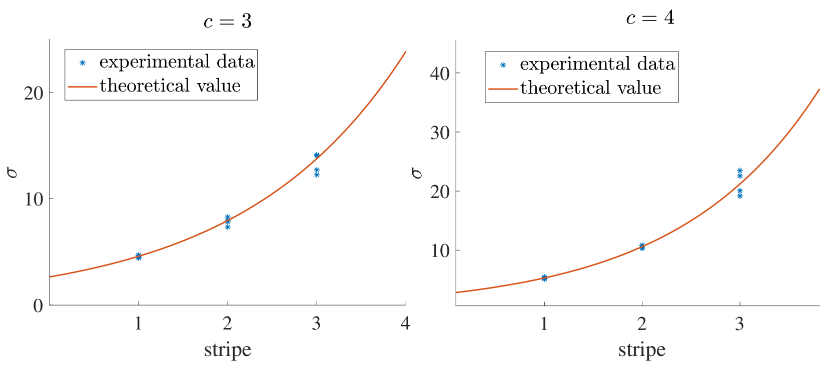


Figure 5: Distribution of errors

We present experimental results to verify the basic assumptions to support our analysis. While we have discussed using c-sum solver for LWE by combining multiple samples and proposed a time-memory trade-off to evaluate its efficiency, an important basis of the algorithms is the Independence Heuristic. In studies of LPN, the heuristic is often recognized, and the dependent samples after previous c-sum stripes do not explicitly affect the behavior of elements of samples in the next stripe. In the LWE problem, we also need this assumption to ensure the success probability and input sample size are correctly estimated. Furthermore, we need to verify the overall success probability after a few stripes, and expect that different stripes present similar success rates as a result of the Independence Heuristic.

Table 5: Success rate of BKW reduction

	$q = 23, a = 4, b = 4, c = 3$				$q = 31, a = 4, b = 4, c = 3$		
	stripe	success rate	prediction		stripe	success rate	prediction
$N = 660$	1	0.85	0.8	$N = 1200$	1	0.92	0.9
	2	0.74			2	0.85	
	3	0.79			3	0.93	
	4	0.81			4	0.86	
$N = 650$	1	0.54	0.5	$N = 1180$	1	0.45	0.5
	2	0.55			2	0.61	
	3	0.58			3	0.55	
	4	0.56			4	0.46	
$N = 640$	1	0.22	0.2	$N = 1170$	1	0.29	0.3
	2	0.23			2	0.38	
	3	0.20			3	0.21	
	4	0.21			4	0.32	

We first implement the c-sum algorithm in sage [Sag], and test the probability of recovering solutions in multiple stripes. A solution consists of a list of N single solutions. We run a BKW sample reduction process to test the assumption. For each parameter setting, we repeat 200 tests. The frequencies of finding enough single solutions in different stripes are listed in Table 5. The experimental results match with the predicted success probability in most cases. We also find that success rates in different stripes are all similar to the theoretical value, indicating that the generated samples by c-sum solver can be treated as independent. This agrees with the Independence Heuristic.

We also verify the behavior of error terms in the process of BKW sample reduction. In theory, the variant of errors σ grows to $\sqrt{c}\sigma$ in every c-sum stripe. Our experiment set the initial error rate $\alpha = 0.05$, and monitor the new error after every stripe. The input sample size is 2000. In our experiments, the error terms still follow the claim for more than one stripes. We can assume that the dependency in the reduced samples has a relatively small impact on the error terms. The results are shown in Figure 5.