The Power of Few Qubits and Collisions – Subset Sum below Grover's Bound

Alexander Helm* and Alexander May**

Ruhr-University Bochum, Germany
alexander.helm@rub.de, alex.may@rub.de

Abstract. Let $a_1, \dots a_n, t$ be a solvable subset sum instance, i.e. there exists a subset of the a_i that sums to t. Such a subset can be found with Grover search in time $2^{\frac{n}{2}}$, the square root of the search space, using only $\mathcal{O}(n)$ qubits. The only quantum algorithms that beat Grover's square root bound – such as the Left-Right-Split algorithm of Brassard, Hoyer, Tapp – either use an exponential amount of qubits or an exponential amount of expensive classical memory with quantum random access (QRAM). We propose the first subset sum quantum algorithms that breaks the square root Grover bound with linear many qubits and without QRAM. Building on the representation technique and the quantum collision finding algorithm from Chailloux, Naya-Plasencia and Schrottenloher (CNS), we obtain a quantum algorithm with time $2^{0.48n}$.
Using the Schroeppel-Shamir list construction technique, we further improve downto run time $2^{0.43n}$. The price that we have to pay for beating the square root bound is that as opposed to Grover search our algorithms require classical memory, but no QRAM, i.e. we get a time/memory/qubit tradeoff. Thus, our algorithms have to be compared to purely classical time/memory subset sum trade-offs such as those of Howgrave-Graham and Joux. Our quantum algorithms improve on these purely classical algorithms for all memory complexities $M < 2^{0.2n}$. As an example, for memory $2^{0.1n}$ we obtain run time $2^{0.47n}$ as opposed to $2^{0.63n}$ for the best classical algorithm.

Keywords: Quantum Algorithms · Amplitude Amplification · Representation Technique · Subset Sum · Collision Finding

1 Introduction

Although there is remarkable progress in the development of quantum computing devices, in the medium-term we will implement our quantum algorithms with a very limited number of qubits. Thus, it is of great importance both from a theoretical and practical perspective to develop algorithms that run with small quantum memory consumption, say polynomial or even linear.

A prominent candidate for sharpening our algorithmic tools is the random subset sum problem, which lies at the heart of many post-quantum hardness

* Founded by NRW Research Training Group SecHuman.
** Funded by DFG under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

assumptions such as SIS [Reg09]. Random subset sum instances consist of randomly chosen $a_1, \ldots, a_n \in \mathbb{Z}_{2^n}$, and a t that is the sum of a subset of a_i's modulo 2^n.

Classically, random subset sum instances can be solved with *polynomial memory* using collision finding and the representation technique [HGJ10] in time $2^{0.65n}$ [BCJ11, EM19], and without memory restrictions in time and space $2^{0.29n}$ [BCJ11]. There exist various time/memory tradeoffs in between [HGJ10, BCJ11, DDKS12, DEM19].

Quantumly, random subset sum can be solved with $\mathcal{O}(n)$ qubits in time $2^{n/2}$ using Grover search [Gro96]. The Left-Right-Split algorithm of Brassard, Hoyer and Tapp [BHT98, BJLM13] solves subset sum in time $2^{n/3}$ using $\mathcal{O}(n)$ many qubits, but also using $2^{n/3}$ classical memory with quantum random access (QRAM). However, QRAM is believed to be expensive to realize in practice [GR04]. The currently best time bound of $2^{0.23n}$ for subset sum is achieved by using a quantum random walk technique on the Becker-Coron-Joux algorithm [HM18]. However, this quantum walk algorithm also requires $2^{0.23n}$ many qubits, and therefore is practically completely unattainable.

In this paper, we want to focus on subset sum algorithms with a linear amount of qubits and without using QRAM. Our central research question is whether we can beat the Grover square root bound $2^{n/2}$ in our setting. Notice that our setting is motivated by the research direction initiated by Chailloux, Naya-Plasencia and Schrottenloher [CNPS17]. The authors of [CNPS17] developed a hash collision algorithm (called CNS) for hash functions $\{0,1\}^* \to \{0,1\}^n$ with run time $2^{2/5n}$ using $2^{1/5n}$ classical memory (without QRAM) and $\mathcal{O}(n)$ qubits.

Our contribution. Since the best classical polynomial memory algorithms for subset sum also use collision finding, we take CNS quantum collision finding as our starting point. Combining CNS with the representation technique, we achieve a first quantum subset sum algorithm with run time $2^{0.48n}$ and classical memory $2^{0.24n}$. While this already breaks Grover's bound using only $\mathcal{O}(n)$ many qubits, our first algorithm is still a bit unsatisfactory. Namely, there exist purely classical subset sum algorithms that with the same memory $2^{0.24n}$ achieve run time only $2^{0.39n}$, without using any qubits. Thus, our first algorithm does not improve on the known classical subset sum landscape.

Based on our first CNS adaption, we develop a second subset sum quantum algorithm using Schroeppel-Shamir list construction [SS81]. With $\mathcal{O}(n)$ qubits, our new construction achieves a time/memory tradeoff with

$$\text{time } T = \frac{2^{0.5n}}{M^{0.25}} \quad \text{for any classical memory } M \leq 2^{0.28n} \ .$$

The resulting tradeoff line is depicted in Figure 1 as 2^{nd} algorithm. Notice that for maximal memory $2^{0.28n}$ we go downto $2^{0.43n}$. As desired, our algorithm achieves Grover complexity $2^{n/2}$ when we use no memory. Moreover for any additional memory, we go below the Grover bound, see the red area in Figure 1. In comparison to purely classical time-memory tradeoffs, we improve for any

memory $M \leq 2^{n/5}$, see the shaded area in Figure 1. Thus, our relatively moderate $\mathcal{O}(n)$ qubit memory provides speedups in a relatively large parameter space.

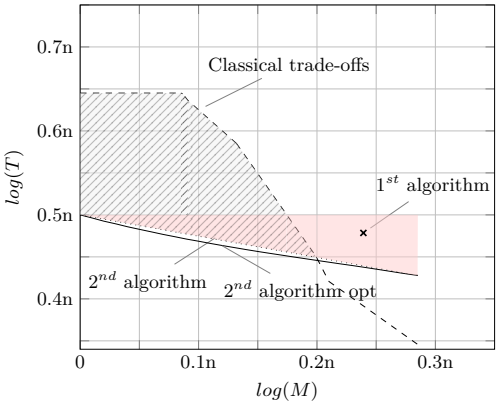

Fig. 1. Comparison of our results and the previous best classical trade-offs for subset sum.

We further optimize our second algorithm, resulting in the slightly improved convex curve denoted in Figure 1 by 2^{nd} algorithm opt.

Our paper is organized as follows. In Section 2 we recall the CNS quantum collision finding algorithm [CNPS17]. We develop our first subset sum algorithm based on collision finding in Section 3. The second subset sum algorithm, that achieves a linear time-memory tradeoff, is described in Section 4. In Section 4.1 we further optimize our second quantum subset sum algorithm.

2 Quantum Collision Finding

Let us briefly define some preliminaries and recall the CNS collision finding algorithm [CNPS17].

We consider random subset sum instances defined as follows.

Definition 1 (Random Subset Sum). *Let* $\mathbf{a}$ *be chosen uniformly at random from* $(\mathbb{Z}_{2^n})^n$*. For a random* $\mathbf{e} \in \{0,1\}^n$ *with Hamming weight* $\mathrm{wt}(\mathbf{e}) = \frac{n}{2}$ *we define* $t = \langle \mathbf{a}, \mathbf{e} \rangle = \sum_{i=1}^{n} a_i e_i \bmod 2^n$*. Then* $(\mathbf{a}, t) \in (\mathbb{Z}_{2^n})^{n+1}$ *is called a* random subset sum instance *and any* $\mathbf{e}' \in \{0,1\}^n$ *with* $\langle \mathbf{a}, \mathbf{e}' \rangle = t$ *is called a* solution.

By Definition 1, every random subset sum instance has at least one solution and with high probability at most $\mathrm{poly}(n)$ many solutions. For ease of notation we assume, that we have a unique solution $\mathbf{e}$, which is the worst-case for our algorithms. Any $(\mathbf{x}_1, \ldots, \mathbf{x}_k) \in \{-1, 0, 1\}^n$ with $\mathbf{e} = \sum_{i=1}^{k} \mathbf{x}_i$ is called a *representation* of $\mathbf{e}$.

By $H(\cdot)$ we denote the binary entropy function $H(\alpha) := -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$ for $\alpha \in [0,1]$, where $0 \cdot \log 0 := 0$. We use Stirling's formula to approximate binomial coefficients by the entropy function, as

$$\binom{n}{m} = \tilde{\Theta}\left(2^{H(\frac{m}{n})n}\right) \ ,$$

where the soft-Oh notion suppresses polynomial factors. We also round upwards for ease of notation, e.g. we have $n2^{n/3} = \tilde{\mathcal{O}}(2^{n/3}) \leq 2^{0.34n}$ for sufficiently large n.

Let $\mathcal{A}$ be an algorithm that implements $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ within runtime t_f. Then the quantum unitary O_f on $n+m$ and additional ancilla qubits defined as

$$O_f(|\mathbf{x}\rangle |\mathbf{y}\rangle) \coloneqq |\mathbf{x}\rangle |\mathbf{y} \oplus f(\mathbf{x})\rangle$$

can be implemented in runtime $T_f = \mathcal{O}(t_f)$.

Set-Membership Oracle *[CNPS17]:* Let $|\phi\rangle = \sum_i |\mathbf{x}_i\rangle$ be a quantum superposition, where each $\mathbf{x}_i \in \mathbb{F}_2^n$ has non-zero amplitude. Let $L \subseteq \mathbb{F}_2^n$ define a list. Notice that by definition, L does not contain an element twice.

We want to know which of the vectors $\mathbf{x}_i$ of $|\phi\rangle$ are in L. To this end, define the characteristic function

$$f_L : \mathbb{F}_2^n \to \mathbb{F}_2, \quad \mathbf{x} \mapsto \begin{cases} 1 & \text{if } \mathbf{x} \in L \\ 0 & \text{else} \end{cases} \ .$$

Then the quantum set-membership oracle for L with operator

$$O_{f_L}(|\phi\rangle |0\rangle) \coloneqq \sum_i |\mathbf{x}_i\rangle |f_L(\mathbf{x}_i)\rangle \tag{1}$$

can be computed in time $T_{f_L} = \mathcal{O}(n \cdot |L|)$ with $2n+1$ qubits.

Amplitude Amplification *[BHMT02]:* Let $\mathcal{A}$ be a quantum algorithm that works with no measurements and produces a uniformly distributed superposition $|\phi\rangle = \sum_{\mathbf{x}\in\mathcal{X}} |\mathbf{x}\rangle$ for some set $\mathcal{X} \subseteq \mathbb{F}_2^n$ in runtime $T_\mathcal{A}$. Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function with quantum unitary O_f on $n+1$ and additional ancilla qubits that has runtime T_f.

Let us define the set $\mathcal{X}_f \coloneqq \{\mathbf{x} \in \mathcal{X} \mid f(\mathbf{x}) = 1\}$. Thus a random $x \in \mathcal{X}$ evaluates to $f(\mathbf{x}) = 1$ with probability $p = \frac{|\mathcal{X}_f|}{|\mathcal{X}|}$. Then there exists a quantum algorithm, called *amplitude amplification*, that outputs an element $\mathbf{x} \in \mathcal{X}$ with $f(\mathbf{x}) = 1$ by sending $\mathcal{O}(\sqrt{p^{-1}})$ many queries to $\mathcal{A}, \mathcal{A}^{-1}, O_f$ and O_f^{-1} and finally measures.
We call $\mathcal{A}$ the setup and f the oracle function of amplitude amplification and set $T_{Setup} = T_\mathcal{A}$. The total runtime of amplitude amplification is given by

$$T = \tilde{\mathcal{O}}\left((T_{Setup} + T_f) \cdot \sqrt{p^{-1}}\right) \ . \tag{2}$$

Amplitude amplification is a generalization of Grover search [Gro96]. Notice that in the Grover setting we have $\mathcal{X} = \mathbb{F}_2^n$ and $\mathcal{A}$ consists of a Hadamard operation on each qubit, which can be done efficiently in time $T_{Setup} = \mathcal{O}(1)$.

Remark 1. For the setup $\mathcal{A}$ we may use Grover search without final measurement. If we use as Grover oracle the characteristic function $g(\mathbf{x}) = 1 \Leftrightarrow \mathbf{x} \in \mathcal{X} \subseteq \mathbb{F}_2^n$ of $\mathcal{X}$, we achieve the desired uniform superposition $|\phi\rangle = \sum_{\mathbf{x}\in\mathcal{X}} |\mathbf{x}\rangle$. A random $\mathbf{x} \in \mathbb{F}_2^n$ evaluates to $g(\mathbf{x}) = 1$ with probability $p = \frac{|\mathcal{X}|}{|\mathbb{F}_2^n|}$. This implies $T_{Setup} = \tilde{\mathcal{O}}\left(T_g \cdot \sqrt{p^{-1}}\right)$.

CNS Quantum Collision Finding *[CNPS17].* Let us slightly adapt CNS quantum collision finding to our needs. Instead of finding a collision for a random function $h\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ we use two random functions $h_i\colon S_i \to \mathbb{F}_2^n$ for $i = 1, 2$ with arbitrary domains S_1, S_2 satisfying $|S_1| \leq 2^n$. We denote the set of collisions between h_1 and h_2 by

$$C = \{(\mathbf{x}_c, \mathbf{x}_q) \in S_1 \times S_2 \mid h_1(\mathbf{x}_c) = h_2(\mathbf{x}_q)\} \text{ with } |C| = R \ .$$

Let $(\mathbf{x}_c, \mathbf{x}_q) \in C$. We call $\mathbf{x}_c$ the *classical half* and $\mathbf{x}_q$ the *quantum half* of a collision, since we store $\mathbf{x}_c$ classically and compute $\mathbf{x}_q$ in quantum superposition.

Correctness. The CNS algorithm (see Algorithm 0) finds a collision $(\mathbf{x}_c, \mathbf{x}_q) \in C$ with only $\mathcal{O}(n)$ qubits in a two step process, see also Figure 2. First one constructs a classically stored list L that contains candidates for the classical half $\mathbf{x}_c$ of a collision. The second step is an amplitude amplification that quantumly enumerates in superposition potential quantum halves $\mathbf{x}_q$ of a collision. We find matching halves by using the quantum set-membership oracle for L.

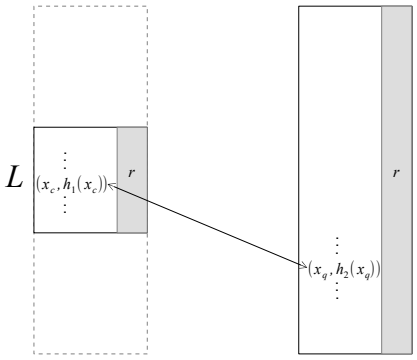

Fig. 2. Main idea of CNS quantum collision finding (Algorithm 0).

Algorithm 0: Quantum Collision Finding

Input : $h_i \colon S_i \to \mathbb{F}_2^n$ for $i = 1, 2$
Output : $(\mathbf{x}_c, \mathbf{x}_q) \in S_1 \times S_2$ with $h_1(\mathbf{x}_c) = h_2(\mathbf{x}_q)$
Parameters: Optimize r, ℓ.

1. Let $S_r^{h_i} \coloneqq \{\mathbf{x} \in S_i \mid h_i(\mathbf{x}) = 0 \bmod 2^r\}$ for $i = 1, 2$.
   Construct, element-wise via Grover search, a sorted (by second entry) list

$$L = \{(\mathbf{x}_c, h_1(\mathbf{x}_c)) \in S_r^{h_1} \times \mathbb{F}_2^n\} \text{ with } |L| = 2^\ell.$$

2. Perform amplitude amplification with the following Setup and Oracle.

(i) Setup: Construct

$$|\phi_r\rangle \coloneqq \frac{1}{\sqrt{|S_r^{h_2}|}} \sum_{\mathbf{x}_q \in S_r^{h_2}} |\mathbf{x}_q, h_2(\mathbf{x}_q)\rangle\, |0\rangle\,.$$

(ii) Oracle: Set-Membership-Oracle $O_{f_L^h}$

$$O_{f_L^h}(|\phi_r\rangle) = \frac{1}{\sqrt{|S_r^{h_2}|}} \sum_{\mathbf{x}_q \in S_r^{h_2}} |\mathbf{x}_q, h_2(\mathbf{x}_q)\rangle\, |f_L^h(\mathbf{x}_q)\rangle\,.$$

Amplitude amplification eventually outputs some $|(\mathbf{x}_q, h_2(\mathbf{x}_q))\rangle\, |1\rangle$.

3. For the quantum half $\mathbf{x}_q$ search for the classic half $\mathbf{x}_c \in L$ with $h_1(\mathbf{x}_c) = h_2(\mathbf{x}_q)$.

In more detail, we construct a sorted (by second entry) list

$$L = \{(\mathbf{x}_c, h_1(\mathbf{x}_c)) \mid h_1(\mathbf{x}_c) = 0 \bmod 2^r\} \subseteq S_1 \times \mathbb{F}_2^n \ ,$$

where each element of L is constructed via Grover search. Since we fix r bits in L, on expectation $\frac{|S_1|}{2^r}$ elements of S_1 fulfill restriction $h_1(\mathbf{x}_c) = 0 \bmod 2^r$, and $h_1(\mathbf{x}_c)$ can take at most 2^{n-r} different values. Since elements in L are different and $|S_1| \leq 2^n$, we obtain the restriction

$$\log L = \ell \leq \min\{\log |S_1| - r, n - r\} = \log |S_1| - r \ .$$

Furthermore we want to guarantee that on expectation L contains at least one element $\mathbf{x}_c$ that can be completed to a collision $(\mathbf{x}_c, \mathbf{x}_q)$. In other words, we need an $\mathbf{x}_c \in L$ such that there exists an $\mathbf{x}_q$ with $(\mathbf{x}_c, \mathbf{x}_q) \in C$, $|C| = R$. A random $\mathbf{x}_c \in S_1$ can be completed to a collision with probability $\frac{R}{|S_1|}$. Thus, L should contain at least $\frac{|S_1|}{R}$ many elements, leading to the condition

$$\log |S_1| - \log R \leq \ell \leq \log |S_1| - r \ . \tag{3}$$

From (3) we obtain

$$0 \leq r \leq \log R \ . \tag{4}$$

For amplitude amplification we define the oracle function

$$f_L^h(\mathbf{x}_q) \coloneqq \begin{cases} 1 & \text{if } \exists(\mathbf{x}_c, h_1(\mathbf{x}_c)) \in L \text{ with } h_1(\mathbf{x}_c) = h_2(\mathbf{x}_q) \\ 0 & \text{else} \end{cases} \quad .$$

Then the set-membership oracle for L, defined for a single $\mathbf{x}_q$, becomes

$$O_{f_L^h}(|\mathbf{x}_q, h_2(\mathbf{x}_q)\rangle\,|0\rangle) \coloneqq |\mathbf{x}_q, h_2(\mathbf{x}_q)\rangle\,|f_L^h(\mathbf{x}_q)\rangle \quad .$$

Runtime. Let $|L| = 2^\ell$. By the randomness of h_1, every $\mathbf{x}_c \in S_1$ satisfies the restriction $h_1(\mathbf{x}_c) = 0 \bmod 2^r$ with probability $p = 2^{-r}$. Thus the runtime of the first step is

$$T_1 = \tilde{\Theta}\left(|L| \cdot \sqrt{p^{-1}}\right) = \tilde{\Theta}\left(2^\ell \cdot 2^{\frac{r}{2}}\right) \quad . \tag{5}$$

In the second step of Algorithm 0 we create a superposition over the set $S_r^{h_2} \coloneqq \{\mathbf{x}_q \in S_2 \mid h_2(\mathbf{x}_q) = 0 \bmod 2^r\}$, as described in Remark 1. By the randomness of h_2, every $\mathbf{x}_q \in S$ satisfies the restriction $h_2(\mathbf{x}_q) = 0 \bmod 2^r$ with probability $p = 2^{-r}$. Hence the setup runtime of amplitude amplification is

$$T_{Setup} = \tilde{\mathcal{O}}\left(\sqrt{p^{-1}}\right) = \tilde{\mathcal{O}}\left(2^{\frac{r}{2}}\right) \quad .$$

As described before, the quantum set-membership oracle for L requires time

$$T_{f_L^h} = \tilde{\mathcal{O}}(|L|) = \tilde{\mathcal{O}}(2^\ell) \quad .$$

Recall that a random element $\mathbf{x}_c \in S_1$ can be completed to a collision $(\mathbf{x}_c, \mathbf{x}_q)$ with probability $\frac{R}{|S_1|}$. Notice that this probability is unchanged if we choose a random $\mathbf{x}_c \in S_r^{h_1}$ as in Algorithm 0. Therefore, we expect in total $|L| \cdot \frac{R}{|S_1|}$ many collisions between L and the set $S_r^{h_2}$ constructed in superposition. Thus, every $\mathbf{x}_q \in S_r^{h_2}$ evaluates to $f_L^h(\mathbf{x}_q) = 1$ with probability

$$p = \frac{|L|R}{|S_r^{h_2}||S_1|} = \frac{|L|R2^r}{|S_1||S_2|} \quad .$$

Using (2), we obtain for the second step (amplitude amplification) runtime

$$\begin{aligned} T_2 &= \tilde{\mathcal{O}}\left(\left(T_{Setup} + T_{f_L^h}\right) \cdot \sqrt{p^{-1}}\right) = \tilde{\mathcal{O}}\left(\left(2^{\frac{r}{2}} + 2^\ell\right) \cdot \sqrt{\frac{|S_1||S_2|}{|L|R2^r}}\right) \\ &= \tilde{\mathcal{O}}\left(\left(2^{-\frac{\ell}{2}} + 2^{\frac{\ell - r}{2}}\right) \frac{|S_1|^{\frac{1}{2}}|S_2|^{\frac{1}{2}}}{|R|^{\frac{1}{2}}}\right) \quad . \end{aligned} \tag{6}$$

Since L is sorted, the third step of Algorithm 0 runs in time $\mathcal{O}(\ell)$. In total, we obtain runtime

$$T = \tilde{\mathcal{O}}\left(\max\{T_1, T_2\}\right) \quad . \tag{7}$$

3 Subset Sum via Quantum Collision Finding

Let $\mathbf{e}$ be a unique solution for a random subset sum instance from Definition 1. Let $(\mathbf{x}_c, \mathbf{x}_q) \in \{-1, 0, 1\}^n$ be a representation of $\mathbf{e}$, i.e. $\mathbf{e} = \mathbf{x}_c + \mathbf{x}_q$. Then $\langle \mathbf{a}, \mathbf{e} \rangle = t$ which implies

$$\langle \mathbf{a}, \mathbf{x}_c \rangle = t - \langle \mathbf{a}, \mathbf{x}_q \rangle \ . \tag{8}$$

Let $S_i \subseteq \{-1, 0, 1\}^n$ for $i = 1, 2$. We define two functions $\Sigma_i : S_i \to \mathbb{Z}_{2^n}$, $i = 1, 2$, with

$$\Sigma_1 : \mathbf{x} \mapsto \langle \mathbf{a}, \mathbf{x} \rangle \text{ and } \Sigma_2 : \mathbf{x} \mapsto t - \langle \mathbf{a}, \mathbf{x} \rangle \ . \tag{9}$$

Then every representation $(\mathbf{x}_c, \mathbf{x}_q)$ of $\mathbf{e}$ is a collision of Σ_1, Σ_2, i.e. $\Sigma_1(\mathbf{x}_c) = \Sigma_2(\mathbf{x}_q)$. However, the converse is not true.

Let $(\mathbf{x}_c, \mathbf{x}_q) \in \{-1, 0, 1\}^{2n}$ be a collision of Σ_1, Σ_2. Then by construction $(\mathbf{x}_c, \mathbf{x}_q)$ fulfills Equation (8) and therefore satisfies the subset sum identity $\langle \mathbf{a}, \mathbf{x}_c + \mathbf{x}_q \rangle = t$. However, in general we have $\mathbf{x}_c + \mathbf{x}_q \in \{-2, \ldots, 2\}^n$. Therefore, $(\mathbf{x}_c, \mathbf{x}_q)$ is a representation of the unique solution $\mathbf{e}$ iff $\mathbf{x}_c + \mathbf{x}_q \in \{0, 1\}^n$.

Definition 2. *Let $(\mathbf{x}_c, \mathbf{x}_q) \in \{0, 1, -1\}^{2n}$ be a collision of Σ_1, Σ_2. We call $(\mathbf{x}_c, \mathbf{x}_q)$* consistent *iff $\mathbf{x}_c + \mathbf{x}_q \in \{0, 1\}^n$.*

Hence solving subset sum is equivalent to finding a consistent collision $(\mathbf{x}_c, \mathbf{x}_q)$. Moreover, the representations of $\mathbf{e}$ are exactly the consistent collisions of Σ_1, Σ_2. Remark 2 follows.

Remark 2. The number R of representations of the solution $\mathbf{e}$ is equal to the number of consistent collisions of Σ_1, Σ_2.

Remark 3. Notice that our hash function Σ_1 is linear, i.e.

$$\Sigma_1(\mathbf{x} + \mathbf{y}) = \langle \mathbf{a}, \mathbf{x} + \mathbf{y} \rangle = \langle \mathbf{a}, \mathbf{x} \rangle + \langle \mathbf{a}, \mathbf{y} \rangle = \Sigma_1(\mathbf{x}) + \Sigma_1(\mathbf{y}) \ .$$

We use this linearity for our improved algorithm in Section 4.

It remains to define good representations $(\mathbf{x}_c, \mathbf{x}_q)$ of $\mathbf{e}$. Let us start for didactical reasons with a natural, unique representation of $\mathbf{e}$ that fails to beat Grover's square root bound.

Unique Representation. Let us define

$$S_1 = \{0, 1\}^{n/2} \times 0^{n/2} \text{ and } S_2 = 0^{n/2} \times \{0, 1\}^{n/2} \ .$$

Then every $\mathbf{e}$ has a unique representation in $S_1 \times S_2$. This implies that Σ_1, Σ_2 have a single collision, i.e. $R = 1$. Condition (3) implies

$$\ell \geq \log |S_1| - \log R = \frac{n}{2} \ .$$

From equation (5) we have $T_1 = \Omega(2^{\frac{n}{2}})$, which implies that we cannot beat Grover's bound.

More Representations. Let $0 \leq \alpha \leq 1/4$. For $i = 1, 2$ we define

$$S_i = \{\mathbf{x} \in \{-1, 0, 1\}^n \mid \mathbf{x} \text{ contains } \left(\frac{1}{4} + \alpha\right) n \text{ many } 1\text{'s and } \alpha n \text{ many } (-1)\text{'s}\} \tag{10}$$

with size

$$|S_1| = |S_2| = \binom{n}{(\frac{1}{4} + \alpha)n, \alpha n} .$$

By the choice of S_1, S_2 every 1-entry of the solution $\mathbf{e}$ can be represented as $1+0$ and $0+1$ and every 0-entry can be represented as $0+0$, $1+(-1)$ and $(-1)+1$. Thus the number of representations is

$$R = \binom{\frac{n}{2}}{\frac{n}{4}} \binom{\frac{n}{2}}{\alpha n, \alpha n} .$$

Quantum Collision Finding for Subset Sum. Let us adapt the CNS quantum collision finding (Algorithm 0) to our subset sum setting, resulting in Algorithm 1.

Algorithm 1: Quantum Subset Sum Collision Finding

Input : $(\mathbf{a}, t) \in (\mathbb{Z}_{2^n})^{n+1}$
Output : $\mathbf{e} \in \{0, 1\}^n$
Parameters: Optimize r, ℓ, α as $r = 0.4784n$, $\ell = 0.2392n$, $\alpha = 0.0175$.

1. Let $S_r^{\Sigma_i} \coloneqq \{\mathbf{x} \in S_i \mid \Sigma_i(\mathbf{x}) = 0 \bmod 2^r\}$ for $i = 1, 2$.
   Construct, element-wise via Grover search, a sorted (by second entry) list

$$L = \{(\mathbf{x}_c, \Sigma_1(\mathbf{x}_c)) \in S_r^{\Sigma_1} \times \mathbb{Z}_{2^n}\} \text{ with } |L| = 2^\ell.$$

2. Perform amplitude amplification with the following Setup and Oracle.

(i) Setup: Construct

$$|\phi_r\rangle \coloneqq \frac{1}{\sqrt{|S_r^{\Sigma_2}|}} \sum_{\mathbf{x}_q \in S_r^{\Sigma_2}} |\mathbf{x}_q, \Sigma_2(\mathbf{x}_q)\rangle\, |0\rangle .$$

(ii) Oracle: Set-Membership-Oracle $O_{f_L^\Sigma}$

$$O_{f_L^\Sigma}(|\phi_r\rangle) = \frac{1}{\sqrt{|S_r^{\Sigma_2}|}} \sum_{\mathbf{x}_q \in S_r^{\Sigma_2}} |\mathbf{x}_q, \Sigma_2(\mathbf{x}_q)\rangle\, |f_L^\Sigma(\mathbf{x}_q)\rangle .$$

Amplitude amplification eventually outputs some $|(\mathbf{x}_q, \Sigma_2(\mathbf{x}_q))\rangle\, |1\rangle$.

3. For the quantum half $\mathbf{x}_q$ search for the classic half $\mathbf{x}_c \in L$ with $\Sigma_1(\mathbf{x}_c) = \Sigma_2(\mathbf{x}_q)$ and $\mathbf{x}_c + \mathbf{x}_q \in \{0, 1\}^n$.

We instantiate the hash functions by Σ_1, Σ_2 from equation (9), where S_1, S_2 are defined via (10).

In addition, we have to slightly modify the quantum set-membership oracle, because we have to check for consistency of collisions (Definition 2). Let us define the oracle function

$$f_L^{\Sigma}(\mathbf{x}_q) \coloneqq \begin{cases} 1 & \text{if } \exists(\mathbf{x}_c, \Sigma_1(\mathbf{x}_c)) \in L \text{ with } \Sigma_1(\mathbf{x}_c) = \Sigma_2(\mathbf{x}_q) \wedge \mathbf{x}_c + \mathbf{x}_q \in \{0,1\}^n \\ 0 & \text{else} \end{cases} .$$

Then our quantum set-membership oracle for L, defined for a single $\mathbf{x}_q$, becomes

$$O_{f_L^{\Sigma}}(|\mathbf{x}_q, \Sigma_2(\mathbf{x}_q)\rangle\,|0\rangle) \coloneqq |\mathbf{x}_q, \Sigma_2(\mathbf{x}_q)\rangle\,|f_L^{\Sigma}(\mathbf{x}_q)\rangle \quad . \tag{11}$$

The set-membership oracle can be realized in time $T_{f_L^{\Sigma}} = \tilde{\mathcal{O}}(|L|)$.

Theorem 1. *Algorithm 1 solves random subset sum instances* $(\mathbf{a}, t) \in (\mathbb{Z}_{2^n})^{n+1}$ *in expected time* $T = 2^{0.4785n}$ *using* $\mathcal{O}(n)$ *qubits and memory* $M = 2^{0.2392n}$.

Proof. Our parameter choice $\alpha = 0.0175 \leq \frac{1}{4}$ determines the values of $|S_1|, |S_2|$ and R as

$$|S_1| = |S_2| = \binom{n}{(\frac{1}{4}+\alpha)n, \alpha n} = \tilde{\mathcal{O}}(2^{0.9569n}) \ ,$$
$$R = \binom{\frac{n}{2}}{\frac{n}{4}}\binom{\frac{n}{2}}{\alpha n, \alpha n} = \tilde{\mathcal{O}}(2^{0.7177n}) \ .$$

Moreover, we easily check that our optimized parameter choice $\alpha = 0.0175$, $r = 0.4784n$ and $\ell = 0.2392n$ fulfills restrictions (3) and (4):

$$\log|S_1| - \log R = 0.2392n \leq \ell \leq 0.4785n = \log|S_1| - r \ ,$$
$$0 \leq r \leq 0.7177n = \log R \ .$$

Using equations (5) and (6), we obtain runtimes

$$T_1 = \mathcal{O}\left(2^{\ell} \cdot 2^{\frac{r}{2}}\right) = \tilde{\mathcal{O}}(2^{0.4784n}) \ ,$$
$$T_2 = \tilde{\mathcal{O}}\left(\frac{|S_1|^{\frac{1}{2}}|S_2|^{\frac{1}{2}}}{|R|^{\frac{1}{2}}}2^{-\frac{\ell}{2}} + \frac{|S_1|^{\frac{1}{2}}|S_2|^{\frac{1}{2}}}{|R|^{\frac{1}{2}}}2^{\frac{\ell-r}{2}}\right) = \tilde{\mathcal{O}}(2^{0.47845n} + 2^{0.47845n}) \ .$$

Thus, by equation (7) Algorithm 1 has total runtime

$$T = \tilde{\mathcal{O}}\left(\max\{T_1, T_2\}\right) = \tilde{\mathcal{O}}(2^{0.47845n}) \leq 2^{0.4785n} \quad .$$

The memory complexity is determined by the size of L as

$$M = \tilde{\mathcal{O}}(|L|) = \tilde{\mathcal{O}}(2^{\ell}) = \tilde{\mathcal{O}}(2^{0.2392n}) \leq 2^{0.2392n} \quad .$$

The application of Grover search and amplitude amplification both require only $\mathcal{O}(n)$ many qubits. □

While Theorem 1 beats Grover's bound using only $\mathcal{O}(n)$ qubits and no QRAM, it does not improve over purely classical time/memory tradeoffs, see Figure 1.

4 Using a Classical Algorithm for List Construction

While Algorithm 1 is a direct adaptation of CNS quantum collision finding, it ignores special properties of the subset sum setting. E.g. in step 1 of Algorithm 1 we are building a classical list L, where r bits of the hash function evaluation $\Sigma_1(x)$ are fixed to zero. Each list element is constructed one by one using Grover search, resulting in total runtime $|L| \cdot 2^{r/2}$ for step 1.

However, quantum algorithms like Grover Search are not optimal in finding many solutions to a problem, and Grover search does not take advantage of the linearity of hash function Σ_1 (see Remark 3). We improve the step 1 list construction by using the classical Schroeppel-Shamir algorithm [SS81]. We also tried other more advanced classical list constructions for L such as BCJ [BCJ11], but could not further improve over Schroeppel-Shamir.

To make optimal use of the representation technique, we also have to redefine our search spaces.

Tunable Representations. Let $0 \leq c \leq 1$. We define the following sets

$$T(c) \coloneqq \left\{ \mathbf{x} \in \{-1,0,1\}^{cn} \,\middle|\, \begin{array}{c} \mathbf{x} \text{ contains } (1/4+\alpha)\,cn \text{ many } 1\text{'s} \\ \text{and } \alpha cn \text{ many } (-1)\text{'s} \end{array} \right\},$$
$$B(c) \coloneqq \left\{ \mathbf{x} \in \{0,1\}^{cn} \,\middle|\, \mathbf{x} \text{ contains } \frac{1}{2}cn \text{ many } 1\text{'s} \right\}.$$

Let $0 \leq c_1 \leq 1$. We set our search spaces as

$$S_1 = T(c_1) \times 0^{(1-c_1)n} \ ,$$
$$S_2 = T(c_1) \times B(1-c_1) \ .$$

Therefore, we obtain in S_1, S_2 an overlapping part of length $c_1 n$, and an additional length $(1-c_1)n$ search space for the quantum part, see also Figure 3. In the additional search space $\mathbf{x}_q$ has relative weight $1/2$. In the overlapping part both $\mathbf{x}_c, \mathbf{x}_q$ have relatively (to the length $c_1 n$) $(1/4+\alpha)$ many 1-entries and α many (-1)-entries.

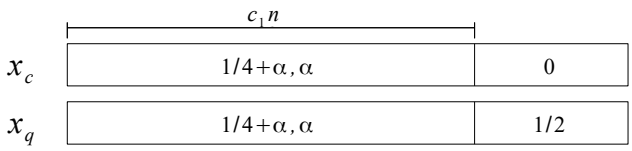

Fig. 3. Visualization of search spaces S_1, S_2.

Thus S_1 and S_2 have sizes

$$|S_1| = |T(c_1)| = \binom{c_1 n}{(\frac{1}{4}+\alpha)c_1 n, \alpha c_1 n} \ ,$$

$$|S_2| = |T(c_1)| \cdot |B(1-c_1)| = \binom{c_1 n}{(\frac{1}{4}+\alpha)c_1 n, \alpha c_1 n} \binom{(1-c_1)n}{\frac{1}{2}(1-c_1)n} \ ,$$

and the number of representations is

$$R = \binom{\frac{1}{2}c_1 n}{\frac{1}{4}c_1 n} \binom{\frac{1}{2}c_1 n}{\alpha c_1 n, \alpha c_1 n} \ .$$

Constructing L via Schroeppel-Shamir. Our hash functions Σ_1, Σ_2 from (9) remain unchanged. We have to compute

$$L = \{(\mathbf{x}_c, \Sigma_1(\mathbf{x}_c)) \in S_1 \times \mathbb{Z}_{2^n} \mid \Sigma_1(\mathbf{x}_c) = 0 \bmod 2^r\} \ .$$

We expect that L has size $\frac{|S_1|}{2^r}$. By Definition 1 of our random subset sum instances, it is not hard to show that by a Chernoff bound with overwhelming probability $|L|$ deviates from its expectation by at most a logarithmic factor. Hence, in the following we set $|L| = \tilde{\Theta}(\frac{|S_1|}{2^r})$.
L can be computed with the Schroeppel-Shamir algorithm in time

$$T_1 = \tilde{\mathcal{O}}\left(\max\left\{|S_1|^{\frac{1}{2}}, |L|\right\}\right) = \tilde{\mathcal{O}}\left(\max\left\{|S_1|^{\frac{1}{2}}, \frac{|S_1|}{2^r}\right\}\right) \tag{12}$$

using classical memory $M = \tilde{\mathcal{O}}(\max\left\{|S_1|^{\frac{1}{4}}, \frac{|S_1|}{2^r}\right\})$.
Our modifications result in Algorithm 2.

Theorem 2. *Algorithm 2 solves random subset sum instances $(\mathbf{a}, t) \in (\mathbb{Z}_{2^n})^{n+1}$ by using only $\mathcal{O}(n)$ qubits in expected runtime*

$$T = \tilde{\mathcal{O}}\left(\frac{2^{0.5n}}{M^{0.2532}}\right) \quad \textit{for any classical memory } M \leq 2^{0.2852n}.$$

Proof. Our parameter choice $\alpha = 0.0042 \leq \frac{1}{4}$ determines the values of $|S_1|, |S_2|$ and R as a function of $0 \leq c_1 \leq 1$ as

$$\begin{aligned}
|S_1| &= \binom{c_1 n}{(\frac{1}{4}+\alpha)c_1 n, \alpha c_1 n} = \tilde{\mathcal{O}}(2^{0.8556 c_1 n}) \ , \\
|S_2| &= \binom{(1-c_1)n}{\frac{1}{2}(1-c_1)n} \binom{c_1 n}{(\frac{1}{4}+\alpha)c_1 n, \alpha c_1 n} = \tilde{\mathcal{O}}(2^{(1-0.1444 c_1)n}) \ , \\
R &= \binom{\frac{1}{2}c_1 n}{\frac{1}{4}c_1 n} \binom{\frac{1}{2}c_1 n}{\alpha c_1 n, \alpha c_1 n} = \tilde{\mathcal{O}}(2^{0.5704 c_1 n}) \ .
\end{aligned}$$

From equation (12) and $r = \log R$, Schroeppel-Shamir runs in time

$$T_1 = \tilde{\mathcal{O}}\left(\max\left\{|S_1|^{\frac{1}{2}}, \frac{|S_1|}{2^r}\right\}\right) \leq \max\left\{2^{0.4278 c_1 n}, 2^{0.2852 c_1 n}\right\} = 2^{0.4278 c_1 n} \ ,$$

Algorithm 2: Quantum Subset Sum Collision Finding II

Input : $(\mathbf{a}, t) \in (\mathbb{Z}_{2^n})^{n+1}$
Output : $\mathbf{e} \in \{0,1\}^n$
Parameters: Optimize r, α as $r = \log R$, $\alpha = 0.0042$ and $0 \leq c_1 \leq 1$.

1. Let $S_r^{\Sigma_i} \coloneqq \{\mathbf{x} \in S_i \mid \Sigma_i(\mathbf{x}) = 0 \bmod 2^r\}$ for $i = 1, 2$.
   Construct, via Schroeppel-Shamir algorithm, a sorted (by second entry) list

$$L = \{(\mathbf{x}_c, \Sigma_1(\mathbf{x}_c)) \in S_r^{\Sigma_1} \times \mathbb{Z}_{2^n}\} \text{ with } |L| = 2^\ell.$$

2. Perform amplitude amplification with the following Setup and Oracle.

(i) Setup: Construct

$$|\phi_r\rangle \coloneqq \frac{1}{\sqrt{|S_r^{\Sigma_2}|}} \sum_{\mathbf{x}_q \in S_r^{\Sigma_2}} |\mathbf{x}_q, \Sigma_2(\mathbf{x}_q)\rangle\, |0\rangle\,.$$

(ii) Oracle: Set-Membership-Oracle $O_{f_L^\Sigma}$

$$O_{f_L^\Sigma}(|\phi_r\rangle) = \frac{1}{\sqrt{|S_r^{\Sigma_2}|}} \sum_{\mathbf{x}_q \in S_r^{\Sigma_2}} |\mathbf{x}_q, \Sigma_2(\mathbf{x}_q)\rangle\, |f_L^\Sigma(\mathbf{x}_q)\rangle\,.$$

Amplitude amplification eventually outputs some $|(\mathbf{x}_q, \Sigma_2(\mathbf{x}_q))\rangle\, |1\rangle$.

3. For the quantum half $\mathbf{x}_q$ search for the classic half $\mathbf{x}_c \in L$ with $\Sigma_1(\mathbf{x}_c) = \Sigma_2(\mathbf{x}_q)$ and $\mathbf{x}_c + \mathbf{x}_q \in \{0,1\}^n$.

using classical memory

$$M = \tilde{\mathcal{O}}\left(\max\left\{|S_1|^{\frac{1}{4}}, \frac{|S_1|}{2^r}\right\}\right) \leq \max\left\{2^{0.2139 c_1 n}, 2^{0.2852 c_1 n}\right\} = 2^{0.2852 c_1 n}. \quad (13)$$

Algorithm 2's amplitude amplification operates on $\mathcal{O}(n)$ qubits without classical memory.

Using equation (6) and $\ell = \log|S_1| - r$, amplitude amplification runs in expected time

$$\begin{aligned} T_2 &= \tilde{\mathcal{O}}\left(\left(2^{-\frac{\ell}{2}} + 2^{\frac{\ell - r}{2}}\right) \frac{|S_1|^{\frac{1}{2}} |S_2|^{\frac{1}{2}}}{|R|^{\frac{1}{2}}}\right) = \tilde{\mathcal{O}}\left(|S_2|^{\frac{1}{2}} + \frac{|S_1||S_2|^{\frac{1}{2}}}{R^{\frac{3}{2}}}\right) \\ &= \tilde{\mathcal{O}}\left(2^{(0.5 - 0.0722 c_1) n} + 2^{(0.5 - 0.0722 c_1) n}\right) = \tilde{\mathcal{O}}\left(2^{(0.5 - 0.0722 c_1) n}\right)\,. \end{aligned}$$

Thus by (7) the total expected runtime is

$$\begin{aligned} T = \tilde{\mathcal{O}}\left(\max\{T_1, T_2\}\right) &= \tilde{\mathcal{O}}\left(\max\left\{2^{0.4278 c_1 n}, 2^{(0.5 - 0.0722 c_1) n}\right\}\right) \\ &= \tilde{\mathcal{O}}\left(2^{(0.5 - 0.0722 c_1) n}\right)\,. \end{aligned}$$

Using $2^{-c_1 n} \leq M^{-\frac{1}{0.2852}}$ from Equation (13), we achieve the desired trade-off

$$T = \tilde{\mathcal{O}}\left(\frac{2^{0.5n}}{M^{0.2532}}\right) \quad \text{for } M \leq 2^{0.2852n} \ .$$

□

Theorem 2 provides a time-memory-tradeoff between runtime and classical memory while using only $\mathcal{O}(n)$ qubits. Note that any classical memory consumption helps us to beat Grover's square root bound. We compare to purely classical time/memory tradeoffs in Figure 4. Notice that our quantum algorithm beats any classical algorithm in the memory regime $M \leq 2^{n/5}$.

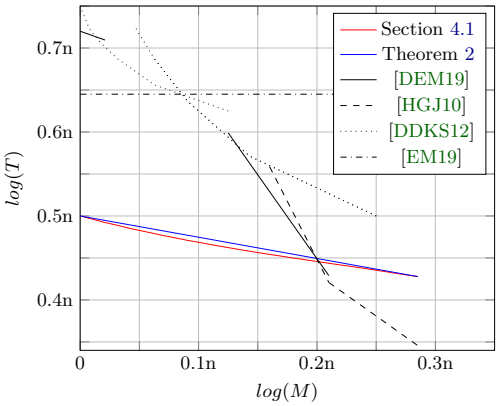

Fig. 4. Comparison of our small qubit algorithm with purely classical time/memory tradeoffs.

4.1 Optimization of Algorithm 2

We further improve on the analysis of Algorithm 2 by elaborating on the representations, the algorithm itself remains unchanged.

More Tunable Representations. Let $0 \leq z \leq 1/2$ and $c, d \in [0,1]$ with $c + d \geq 1$. We define the following sets

$$T(c,d,z) := \left\{ \mathbf{x} \in \{-1,0,1\}^{(c+d-1)n} \middle| \begin{array}{c} \mathbf{x} \text{ contains } (z+\alpha)\,(c+d-1)n \text{ many } 1\text{'s} \\ \text{and } \alpha(c+d-1)n \text{ many } (-1)\text{'s} \end{array} \right\},$$

$$B(c) = \left\{ \mathbf{x} \in \{0,1\}^{cn} \middle| \mathbf{x} \text{ contains } \frac{1}{2}cn \text{ many } 1\text{'s} \right\}.$$

We set our search spaces as

$$S_1 = B(1-c_2) \times T(c_1, c_2, z) \times 0^{(1-c_1)n} \ ,$$
$$S_2 = 0^{(1-c_2)n} \times T(c_1, c_2, \frac{1}{2} - z) \times B(1-c_1) \ .$$

In Algorithm 2 the parameter $0 \leq z \leq 1/2$ controls the relative weight in the

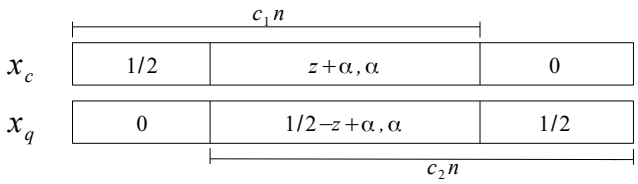

Fig. 5. Visualization of search spaces S_1, S_2.

overlapping part, see Figure 5. The sizes of the search spaces are

$$\begin{aligned} |S_1| &= |B(1-c_2)| \cdot |T(c_1, c_2, z)| \\ &= \binom{(1-c_2)n}{\frac{1}{2}(1-c_2)n} \binom{(c_1+c_2-1)n}{(z+\alpha)(c_1+c_2-1)n, \alpha(c_1+c_2-1)n} \ , \\ |S_2| &= |B(1-c_1)| \cdot |T(c_1, c_2, \frac{1}{2} - z)| \\ &= \binom{(1-c_1)n}{\frac{1}{2}(1-c_1)n} \binom{(c_1+c_2-1)n}{(\frac{1}{2}-z+\alpha)(c_1+c_2-1)n, \alpha(c_1+c_2-1)n} \ , \end{aligned}$$

with the number of representations

$$R = \binom{\frac{1}{2}(c_1+c_2-1)n}{z(c_1+c_2-1)n} \binom{\frac{1}{2}(c_1+c_2-1)n}{\alpha(c_1+c_2-1)n, \alpha(c_1+c_2-1)n} \ .$$

Optimization of c_1, c_2, α and z yields a slight improvement over Theorem 2, as illustrated in Table 1 and Figure 4.

	$\log_2(M)/n$	0.00	0.05	0.10	0.15	0.20	0.25	0.285
Optimization	$\log_2(T)/n$	0.500	0.483	0.469	0.456	0.446	0.436	0.428
Theorem 2		0.500	0.487	0.475	0.462	0.449	0.437	0.428

Table 1. Results of optimization.

References

BCJ11. Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 364–385. Springer, 2011.

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

BHT98. Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *Latin American Symposium on Theoretical Informatics*, pages 163–169. Springer, 1998.

BJLM13. Daniel J Bernstein, Stacey Jeffery, Tanja Lange, and Alexander Meurer. Quantum algorithms for the subset-sum problem. In *International Workshop on Post-Quantum Cryptography*, pages 16–33. Springer, 2013.

CNPS17. André Chailloux, María Naya-Plasencia, and André Schrottenloher. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 211–240. Springer, 2017.

DDKS12. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 719–740, 2012.

DEM19. Claire Delaplace, Andre Esser, and Alexander May. Improved low-memory subset sum and LPN algorithms via multiple collisions. *IACR Cryptology ePrint Archive*, 2019:804, 2019.

EM19. Andre Esser and Alexander May. Low weight discrete logarithms and subset sum in $2^{0.65n}$ with polynomial memory. Cryptology ePrint Archive, Report 2019/931, 2019. https://eprint.iacr.org/2019/931.

GR04. Lov Grover and Terry Rudolph. How significant are the known collision and element distinctness quantum algorithms. *Quantum Info. Comput.*, 4(3):201–206, May 2004.

Gro96. Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

HGJ10. Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 235–256. Springer, 2010.

HM18. Alexander Helm and Alexander May. Subset sum quantumly in 1.17ˆn. In *13th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

Reg09. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

SS81. Richard Schroeppel and Adi Shamir. A t=o(2ˆn/2), s=o(2ˆn/4) algorithm for certain np-complete problems. *SIAM journal on Computing*, 10(3):456–464, 1981.