

Two remarks on the vectorization problem

Wouter Castryck and Natan Vander Meeren

imec-COSIC, KU Leuven

Abstract. We share two small but general observations on the vectorization problem for group actions, which appear to have been missed by the existing literature. The first observation is pre-quantum: explicit examples show that, for classical adversaries, the vectorization problem cannot in general be reduced to the parallelization problem. The second observation is post-quantum: by combining a method for solving systems of linear disequations due to Ivanyos with a Kuperberg-style sieve, one can solve the hidden shift problem, and therefore the vectorization problem, for any finite abelian $2^t p^k$ -torsion group in polynomial time and using mostly classical work; here t, k are any fixed non-negative integers and p is any fixed prime number.

Keywords: group actions, vectorization problem, linear disequations

1 Introduction

This paper discusses two unrelated aspects of the *vectorization problem* for abelian group actions, which specializes to the classical discrete logarithm problem in the case of exponentiation in finite cyclic groups.

The first formal study of cryptographic group actions was carried out in 1990 by Brassard and Yung [8], but non-discrete-logarithm-based examples go back, at least, to the work of Brassard and Crépeau from 1986 [6]. However, none of the early concrete instances were genuinely novel, perhaps with the exception of finite symmetric groups (or abelian subgroups thereof) acting on sets of graphs, whose vectorization problem is just the graph isomorphism problem, famously solved by Babai in 2017 [4, 19]. This situation changed with the independent works of Couveignes [12] and Rostovtsev–Stolbunov [30, 35], who proposed to use ideal-class groups acting on sets of elliptic curves over finite fields through isogenies. Also CSIDH [10] fits within this framework. It is Couveignes who coined the term *vectorization*. The isogeny-based construction attracted a lot of attention, lately, because the corresponding vectorization problem is supposed to be hard even in the presence of quantum adversaries. At the same time, being an abelian group action, it inherits many of the features of the celebrated exponentiation map.

To date, the list of cryptographically interesting group actions remains rather limited, but since it concerns such a basic and flexible concept, it is well imaginable that new constructions remain to be discovered, both for use in a classical

and in a (post-)quantum context, e.g., see [22] for a candidate based on tensors. General statements on the hardness of the vectorization problem help in understanding the fundamental features and limitations of group-action-based cryptography. We present two such statements, which are small addenda to the existing literature, including surveys such as [1, 16, 34], but which appear to have been missed and therefore seem worth reporting.

A pre-quantum observation. Our first statement is classical and negative in nature: very simple constructions show that, classically, one cannot expect in general that the vectorization problem for an abelian group action reduces in polynomial time to the *parallelization problem* (unless in the event of cataclysmic discoveries like $P=NP$). This contrasts with the post-quantum setting, where the vectorization and parallelization problems become computationally equivalent [15, 26]. Our conclusion also contrasts with the discrete logarithm problem, which is believed to be no harder than the computational Diffie–Hellman problem in view of the Maurer–Wolf reduction [25]. It had already been pointed out, e.g. by Smith [34, §11] and Gnilke–Zumbrägel [16, p3], that Maurer–Wolf does not extend to the group action framework. But, as far as we are aware, the existence of alternative classical reductions was not ruled out yet. To the contrary: it has been conjectured that such a reduction should exist, see e.g. [11, §1.2]. The current work refutes this.

A post-quantum observation. Our second observation revisits [9, §3], where it was shown how to combine a classical (= pre-quantum) method due to Friedl et al. [14, §3] for solving systems of linear disequations modulo p with a Kuperberg-style sieve [23]. This led to an easy polynomial-time quantum algorithm which solves the *hidden shift problem*, and therefore the vectorization problem, for groups of the form

$$\mathbb{Z}_{2^{t_1}} \times \mathbb{Z}_{2^{t_2}} \times \cdots \times \mathbb{Z}_{2^{t_m}} \times \mathbb{Z}_p^n, + \quad (1)$$

while relying mostly on classical computations; in particular, the requirements in terms of quantum memory are very limited. Here, p is a fixed prime number and the exponents t_i are bounded by a fixed integer t , but n and m can vary freely. For $t = 1$ and $n = 0$ the algorithm specializes to Simon’s method [33].

In [9] it was left unnoticed that a generalization of the method of Friedl et al. due to Ivanyos [21], capable of solving systems of linear disequations modulo *powers* of p , is equally compatible with Kuperberg’s sieve. This allows one to extend the algorithm from (1) to groups of the form

$$\mathbb{Z}_{2^{t_1}} \times \mathbb{Z}_{2^{t_2}} \times \cdots \times \mathbb{Z}_{2^{t_m}} \times \mathbb{Z}_{p^{k_1}} \times \mathbb{Z}_{p^{k_2}} \times \cdots \times \mathbb{Z}_{p^{k_n}}, + \quad (2)$$

for any fixed prime number p and any number of exponents t_i , resp. k_i , that are bounded by fixed integers t , resp. k . Without affecting the polynomial runtime and the memory-efficiency, that is.

Moreover, as in the case of [9], this extended algorithm can be combined with Kuperberg’s collimation sieve [24, 28], yielding the following refinement of [9, Thm. 1.2]:

Theorem 1. *For any fixed prime number p and non-negative integers t, k , there exists a quantum algorithm for solving the hidden shift problem in any finite abelian group $G, +$ with time, query and QROM-complexity*

$$\text{poly}(\log |G|) \cdot 2^{\mathcal{O}(\sqrt{\log |2^t p^k G|})}$$

and requiring storage of $\text{poly}(\log |G|)$ qubits.

Here QROM stands for quantum read-only memory; this is also known as quantum random-access classical memory (QRACM), see [24, §2].

Paper organization. In Section 2 we quickly recall the vectorization and parallelization problems as well as their connection to the abelian hidden shift problem. In Section 3 we present examples of group actions proving the non-equivalence between vectorization and parallelization in a classical context. We devote a separate Section 4 to solving systems of linear disequations, because a secondary aim of our paper is to make this interesting problem (which is open for moduli as small as 6) more widespread in the cryptographic community. In Section 5 we describe our method for finding hidden shifts in finite abelian $2^t p^k$ -torsion groups, while spending time on recalling the details of its most important plug-in: Ivanyos' algorithm from [21]. We take the opportunity to correct a minor error and to considerably sharpen the estimated runtime. The final Section 6 gives some concluding remarks.

Acknowledgements. The author names are in alphabetical order: see <https://www.ams.org/profession/leaders/CultureStatement04.pdf>. The paper was written in the context of the second-listed author's participation in the Honours @ KU Leuven Programme and was supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement ISOCRYPT - No. 101020788), by CyberSecurity Research Flanders with reference number VR20192203, and by the Research Council KU Leuven under grant number C14/18/067.

2 Vectorization, parallelization and hidden shift

Let $G, +$ be an abelian group. An *action* of G on a finite set X is a map

$$\star : G \times X \rightarrow X : (g, x) \mapsto g \star x$$

satisfying $0 \star x = x$ and $g_1 \star (g_2 \star x) = (g_1 + g_2) \star x$ for all $g_1, g_2 \in G$ and all $x \in X$. Throughout, we make the implicit assumption that the action is only ever evaluated in elements of G and X that admit an efficient representation, and that computing this evaluation is efficient as well. The *stabilizer* of an element $x \in X$ is the subgroup $\text{St}(x) = \{g \mid g \star x = x\} \subseteq G$. The *orbit* of $x \in X$ is the subset $\text{Or}(x) = \{g \star x \mid g \in G\} \subseteq X$ and as soon as G is finite we have

$|\text{Or}(x)| \cdot |\text{St}(x)| = |G|$ for all $x \in X$. Two orbits either coincide or are disjoint, and together the orbits partition X . All elements within one orbit have the same stabilizer. The action is called *free* if all stabilizers are trivial. It is called *transitive* if there is one orbit, only.

Definition 2. *The vectorization problem for \star is about explicitly determining $g \bmod \text{St}(x)$ upon input of $x, g \star x \in \text{Or}(x)$.*

One basic example of a group action is the exponentiation map

$$\mathbb{Z}_n^* \times X : (g, x) \mapsto x^g$$

in a finite cyclic group X of order n . Here, the vectorization problem specializes to the discrete logarithm problem. Note that the generators of X form one orbit, and when restricting the action to this orbit it becomes free and transitive.

The classical Diffie–Hellman key exchange protocol naturally generalizes from exponentiation in cyclic groups to arbitrary abelian group actions. Indeed, after Alice and Bob agree on a base element $x \in X$, Alice acts with a secret $g_0 \in G$ on x and sends the result $g_0 \star x$ to Bob, and likewise Bob acts with a secret $g_1 \in G$ on x and sends $g_1 \star x$ to Alice. Both parties can now compute

$$(g_0 + g_1) \star x = g_1 \star (g_0 \star x) = g_0 \star (g_1 \star x), \quad (3)$$

which can be fed to a key derivation function; note that (3) uses the assumption that G is abelian. Breaking this protocol directly relates to:

Definition 3. *The parallelization problem for \star is about explicitly determining $(g_0 + g_1) \star x$ upon input of $x, g_0 \star x, g_1 \star x \in \text{Or}(x)$.*

The parallelization problem straightforwardly reduces to the vectorization problem but the converse reduction, as we will see in Section 3, does not apply in general. We recall that this story changes in the presence of quantum adversaries, where the converse reduction does apply [15, 26].

When studying the hardness of vectorization and parallelization, one can assume that the action is free and transitive. Indeed, it clearly suffices to assume transitivity because the vectorization problem and the parallelization problem are formulated within one orbit. But then all $x \in X$ have the same stabilizer S , so we can assume freeness by acting with G/S rather than with G . Free and transitive actions necessarily satisfy $|G| = |X|$.

Remark 4. The explicit determination of the stabilizer S can be viewed as an instance of the *hidden subgroup problem* in the abelian group G . Quantumly, this is easy using Shor’s algorithm [32], but classically this may be a hard problem. Nevertheless, it is possible to compute in G/S without knowing S explicitly, because testing equivalence mod S is easy: $g_0 - g_1 \in S$ if and only if $g_0 \star x = g_1 \star x$ for whatever $x \in X$ (assuming transitivity).

For free actions, the vectorization problem can be viewed as an instance of:

Definition 5. Given oracle access to injective functions $f_0, f_1 : G \rightarrow X$ such that there exists an $s \in G$ such that for all $g \in G$ we have $f_0(g) = f_1(g + s)$, the (abelian) hidden shift problem is about finding s .

Indeed, from an input $x, s \star x$ to the vectorization problem we can construct the functions $f_i : G \rightarrow X$ as

$$\begin{aligned} f_0 : g &\mapsto g \star (s \star x), \\ f_1 : g &\mapsto g \star x, \end{aligned}$$

which hide the shift s . Assuming access to an oracle for evaluating the functions f_0, f_1 on arbitrary superpositions over elements of G , there exist quantum algorithms due to Kuperberg [23, 24] for solving the hidden shift problem in subexponential time

$$2^{\mathcal{O}(\sqrt{\log |G|})} \tag{4}$$

as well as subexponential quantum space; more precisely the algorithm from [24] requires storage of $\text{poly}(\log |G|)$ qubits and an amount (4) of QROM. Kuperberg studied this in the context of the hidden subgroup problem in the associated dihedral group $\text{Dih}(G)$, which turns out to be equivalent with the hidden shift problem in G , see [23, §6].

Remark 6. There exist non-injective versions of the abelian hidden shift problem, where the problem of breaking the Legendre pseudo-random function is arguably the best-known instance in cryptography. Such versions may be easier to tackle, quantumly, and will not be considered here; see [18, Ch. 7].

While Kuperberg’s algorithm admits variants with different time-memory trade-offs, see e.g. [29], none of them breaks through the $L_{|G|}(1/2)$ -barrier in general. This does not mean that better quantum algorithms are not possible for special classes of G . Famously, this is true for 2-torsion groups, which can be handled in polynomial time using Simon’s method [33]. This generalizes to 2^t -torsion for any fixed t using Kuperberg’s sieve, see [5]. In a different direction, it generalizes to p -torsion for any fixed prime p using the aforementioned method due to Friedl et al. [14]. The latter authors also present a self-reducibility tool, allowing for a polynomial-time quantum solution to the abelian hidden shift problem in finite abelian groups of *any* fixed exponent r . However, this requires a quantization of otherwise classical post-processing steps, resulting in more complicated quantum algorithms with more restrictive memory requirements; in particular the self-reducibility does not seem suitable for obtaining memory-friendly statements like Theorem 1. As mentioned, in [9, §3] it was shown that for $r = 2^t p$ there exists an easy workaround; we revisit this in Section 5, where we generalize it to $r = 2^t p^k$.

3 Non-equivalence of vectorization and parallelization

We claim that the vectorization problem and the parallelization problem are *not* equivalent as soon as one believes in the existence of one-way group homomorphisms, see e.g. [7, §5]. This does not contradict the results from [15, 26] because,

in the presence of quantum adversaries, no such one-way homomorphisms exist. But pre-quantumly we have several very classical candidates.

The construction is really simple: consider two finite abelian groups $G_0, +$ and $G_1, +$ along with an easy-to-compute but hard-to-invert group homomorphism $f : G_0 \rightarrow G_1$. Then the map

$$\star : G_0 \times G_1 \rightarrow G_1 : (g, x) \mapsto g \star x := x + f(g)$$

is a well-defined action of G_0 on G_1 . The vectorization problem amounts to extracting g from a pair $x, x + f(g)$, which is of course equivalent to extracting g from $f(g)$: this is hard by assumption. On the other hand, the parallelization problem is about computing $x + f(g_0 + g_1) = x + f(g_0) + f(g_1)$ from $x, x + f(g_0)$ and $x + f(g_1)$, which is trivial.

Example 7. One classical example of a one-way group homomorphism is the squaring map

$$\mathbb{Z}_N^*, \cdot \rightarrow \mathbb{Z}_N^*, \cdot : x \mapsto x^2$$

in the unit group of the ring of integers modulo an RSA modulus N . So the vectorization problem for the corresponding group action $(\mathbb{Z}_N^* \times \mathbb{Z}_N^*) \rightarrow \mathbb{Z}_N^* : (g, x) \mapsto g^2 x$ is hard, while parallelization is trivial.

Example 8. A free and transitive example can be obtained from exponentiation

$$\mathbb{Z}_n, + \rightarrow X, \cdot : g \mapsto \alpha^g$$

in a cyclic order- n group $X = \langle \alpha \rangle$ in which the discrete logarithm problem is believed to be hard. The vectorization problem for the corresponding group action $(\mathbb{Z}_n \times X) \rightarrow X : (g, x) \mapsto x\alpha^g$ is hard, and parallelization is straightforward.

Interestingly, Example 7 may have been the first non-exponentiation based group action that saw study in the context of cryptography [6], yet for the purpose of bit commitment rather than key exchange.

4 Systems of linear disequations and the standard approach to hidden shift finding

A system of *linear disequations* over an integer residue ring \mathbb{Z}_r , for some $r > 1$, is a system of the form

$$\begin{cases} a_{11}s_1 + a_{12}s_2 + \dots + a_{1n}s_n \neq b_1, \\ a_{21}s_1 + a_{22}s_2 + \dots + a_{2n}s_n \neq b_2, \\ \vdots \\ a_{m1}s_1 + a_{m2}s_2 + \dots + a_{mn}s_n \neq b_m, \end{cases}$$

with known $a_{ij}, b_i \in \mathbb{Z}_r$, where one wants to solve for s_1, \dots, s_n . It is an intriguing (and not very widespread) open problem how to do this in general. Of course,

for $r = 2$ one just faces a system of linear equations in disguise. More generally, for $r = p$ a prime number, one can re-express every disequation as

$$(a_{i1}s_1 + a_{i2}s_2 + \dots + a_{in}s_n - b_i)^{p-1} = 1, \quad (5)$$

thus obtaining a system of non-linear (as soon as $p > 2$) equations, which can be fed to a Gröbner basis calculation. Alternatively, if we have unlimited access to random disequations then we can solve this by linearization: this is the approach from [14] and it runs in polynomial time for fixed p . This can be generalized to $r = p^k$ for any $k \geq 1$, following Ivanyos [21], but away from prime powers we are clueless about how to approach this problem. Even seemingly harmless rings such as $\mathbb{Z}_6 \cong \mathbb{Z}_2 \times \mathbb{Z}_3$ remain unsolved. Let us stress that the algorithms from [14, 21] are pre-quantum. This being said, we do not know of quantum algorithms that perform significantly better than their pre-quantum counterparts (apart from speed-ups of Grover type [17] in search steps).

Systems of linear disequations naturally show up in the “standard” quantum approach to solving the hidden shift problem in a finite abelian group $G, +$, which can always be assumed to be of the form

$$\mathbb{Z}_{r_1} \times \mathbb{Z}_{r_2} \times \dots \times \mathbb{Z}_{r_n}, +$$

for integers r_i . This standard approach works by generating many *phase vectors*:

Definition 9. *Given a finite abelian group G , let G^\vee denote the dual group. Then for any $\chi \in G^\vee$ and $s \in G$ the quantum state*

$$|\Psi_s(\chi)\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \chi(s)|1\rangle)$$

is called a phase vector over G .

Within our context, the value of $s = (s_1, s_2, \dots, s_n)$ will always be the hidden shift we are looking for: therefore we drop the subscript and just write $|\Psi(\chi)\rangle$. Creating such a phase vector for some uniformly random $\chi \in G^\vee$ is standard practice and comes at the cost of two quantum Fourier transforms, one call to f_0 and one call to f_1 [23, 29]. We treat this as a black box and assume throughout that we have oracle access to phase vectors. We stress that the result of an oracle call is $|\Psi(\chi)\rangle$ with χ a uniformly random, *known* element of G^\vee . The amplitude $\chi(s)$ is unknown, though.

Phase vectors serve as input to the hidden shift finding algorithms due to Kuperberg and others [23, 24, 28, 29]. These algorithms proceed by gradually converting the phase vectors into more interesting ones through a process of combination and measurement; a basic version of Kuperberg’s sieve will appear as a subroutine in Section 5.

For now, we just note that when measuring $|\Psi(\chi)\rangle$ in the $|\pm\rangle$ -basis, where as usual

$$|\pm\rangle = \frac{|0\rangle \pm |1\rangle}{\sqrt{2}},$$

we measure ‘ $-$ ’ with probability $|1 - \chi(s)|^2/4$. Upon such a measurement we can conclude that $\chi(s) \neq 1$. Writing

$$\chi : (g_1, g_2, \dots, g_n) \mapsto \exp\left(2\pi\mathbf{i}\left(\frac{a_1g_1}{r_1} + \frac{a_2g_2}{r_2} + \dots + \frac{a_ng_n}{r_n}\right)\right)$$

for known a_i , this translates into a disequation

$$\frac{r}{r_1}a_1s_1 + \frac{r}{r_2}a_2s_2 + \dots + \frac{r}{r_n}a_ns_n \not\equiv 0 \pmod{r} \quad (6)$$

where $r = \text{lcm}(r_1, r_2, \dots, r_n)$ denotes the exponent of G . Querying many phase vectors leads to a large system of linear disequations, unless $s = (0, 0, \dots, 0)$ in which case one never measures ‘ $-$ ’; but this will be noticed quickly (or it can be tested beforehand). This means that we have effectively reduced the hidden shift problem over G to the problem of solving a system of linear disequations. A more formal discussion will be given in Section 5.2.

Remark 10. Clearly, disequations of the form (6) are not arbitrary. The presence of the cofactors r/r_i is totally natural, since we can only expect to determine s_i modulo r_i . But we also see that each disequation is homogeneous, i.e., all constants b_i are zero. Consequently, this approach will only allow to determine (s_1, s_2, \dots, s_n) up to multiplication with an unknown scalar $\lambda \in \mathbb{Z}_r^*$. This means that, after solving the system, one is still left with the task of determining this scalar, e.g., by exhaustive search.

Unfortunately, as mentioned before, the only moduli r for which we have a solution with polynomial run-time (for fixed r) are prime powers. Our objective however lies in solving the hidden shift problem and, as shown in [9, §3], it is possible to get rid of powers of 2 using a Kuperberg-style sieve prior to running the above reduction. This is recalled, in a generalized setting, in the next section.

5 Finding hidden shifts in $2^t p^k$ -torsion groups

This section covers our algorithm for solving the hidden shift problem in finite abelian $2^t p^k$ -torsion groups. It is an adaptation of [9, §3], where we aim for an incorporation of Ivanyos’ algorithm rather than that of Friedl et al. We can assume that our group $G, +$ is of the form (2) with $t = t_1 \geq \dots \geq t_m \geq 1, k = k_1 \geq \dots \geq k_n \geq 1$ for integers $m, n \geq 0$, and p an odd prime. The hidden shift s is written as $s = (s'_1, \dots, s'_m, s_1, \dots, s_n)$ with $s'_i \in \mathbb{Z}_{2^{t_i}}$ and $s_i \in \mathbb{Z}_{p^{k_i}}$.

5.1 Kuperberg sieve

The goal of this first part of the algorithm is to turn phase vectors over G into phase vectors over the subgroup $H = \mathbb{Z}_{p^{k_1}} \times \dots \times \mathbb{Z}_{p^{k_n}}$. This is done through Kuperberg’s process of *combining phase vectors*, which is about merging $|\Psi(\chi_1)\rangle$ and $|\Psi(\chi_2)\rangle$ into $|\Psi(\chi_1\chi_2^\pm)\rangle$, as follows:

1. Tensor the two phase vectors together:
 $|\Psi(\chi_1)\rangle |\Psi(\chi_2)\rangle = \frac{1}{2}(|00\rangle + \chi_2(s)|01\rangle + \chi_1(s)|10\rangle + \chi_1(s)\chi_2(s)|11\rangle).$
2. Perform a CNOT gate on the second qubit:
 $\frac{1}{2}(|00\rangle + \chi_2(s)|01\rangle + \chi_1(s)|11\rangle + \chi_1(s)\chi_2(s)|10\rangle).$
3. Measure the second qubit:
 $|\Psi(\chi_1\chi_2^\pm)\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \chi_1(s)\chi_2^\pm(s)|1\rangle).$

More generally, one can combine q phase vectors $|\Psi(\chi_1)\rangle, |\Psi(\chi_2)\rangle, \dots, |\Psi(\chi_q)\rangle$ into one phase vector $|\Psi(\chi_1\chi_2^\pm \dots \chi_q^\pm)\rangle$ by repeating this procedure $q-1$ times.

We can use this to obtain phase vectors that are ℓ -divisible for increasing values of ℓ , in the following sense:

Definition 11. *If the character $\chi \in G^\vee$ satisfies*

$$\chi^{2^{t-\ell}p^k} = 1$$

for some $0 \leq \ell \leq t$, then the phase vector $|\Psi(\chi)\rangle$ is said to be ℓ -divisible.

More precisely, if we let r_ℓ denote the largest positive integer for which $t_{r_\ell} \geq t-\ell$, then one can combine $r_\ell + 1$ ℓ -divisible phase vectors

$$|\Psi(\chi_1)\rangle, |\Psi(\chi_2)\rangle, \dots, |\Psi(\chi_{r_\ell+1})\rangle$$

into a single $(\ell + 1)$ -divisible phase vector. Indeed, write every χ_i as

$$(g_1, \dots, g_m, h_1, \dots, h_n) \mapsto \exp\left(2\pi i \left(\frac{a_{i,1}g_1}{2^{t_1}} + \dots + \frac{a_{i,m}g_m}{2^{t_m}} + \frac{b_{i,1}h_1}{p^{k_1}} + \dots + \frac{b_{i,n}h_n}{p^{k_n}}\right)\right)$$

By assumption, for all $1 \leq j \leq r_\ell$ we have $2^{t_j-t+\ell} \mid a_{i,j}$. Setting

$$c_{i,j} := a_{i,j}/2^{t_j-t+\ell} \pmod{2}$$

thus yields $r_\ell + 1$ vectors of the form $(c_{i,1}, \dots, c_{i,r_\ell})$ for $1 \leq i \leq r_\ell + 1$. Furthermore, these vectors are linearly dependent in \mathbb{Z}_2 , which means that there are coefficients $d_1, \dots, d_{r_\ell+1} \in \mathbb{Z}_2$ such that

$$d_1 c_{1,j} + \dots + d_{r_\ell+1} c_{r_\ell+1,j} = 0 \pmod{2}$$

for all $1 \leq j \leq r_\ell$. We can calculate these coefficients classically, and combine the phase vectors $|\Psi(\chi_i)\rangle$ for which $d_i = 1$, in the sense of Kuperberg. The result is a phase vector $|\Psi(\chi)\rangle$ for which

$$\chi : (g_1, \dots, g_m, h_1, \dots, h_n) \mapsto \exp\left(2\pi i \left(\frac{a_1 g_1}{2^{t_1}} + \dots + \frac{a_m g_m}{2^{t_m}} + \frac{b_1 h_1}{p^{k_1}} + \dots + \frac{b_n h_n}{p^{k_n}}\right)\right)$$

is such that the coefficients a_j satisfy $2 \mid \frac{a_j}{2^{t_j}} 2^{t-\ell}$ for $1 \leq j \leq r_\ell$. This implies that the phase vector is in fact $(\ell + 1)$ -divisible. Note that in the procedure above, the phase vectors $|\Psi(\chi_i)\rangle$ for which $d_i = 0$ need not be discarded: they can be kept aside for possible later use.

Pipelining this procedure for $\ell = 0, 1, \dots, t-1$ eventually yields a phase vector $|\Psi(\chi)\rangle$ where $\chi \in G^\vee$ is such that all the coefficients a_1, \dots, a_m are zero. This means that χ depends only on h_1, \dots, h_n . We can therefore interpret this phase vector as a phase vector over H .

5.2 Disequations

Now that we have a procedure returning phase vectors over $H = \mathbb{Z}_{p_1^{k_1}} \times \dots \times \mathbb{Z}_{p_n^{k_n}}$, we can use these for generating linear disequations over \mathbb{Z}_{p^k} along the lines of Section 4. Here we discuss this more formally, while explaining how these disequations can be solved for (s_1, \dots, s_n) . This follows Ivanyos [21], but we take the opportunity to fix a small error in step (a) and to provide a sharper degree bound in step (d), leading to an improved complexity estimate. We stress that these steps are entirely classical. We need the notion of *near uniformity*:

Definition 12. *Given a probability distribution over a finite set A along with a subset $A' \subseteq A$, we say that the distribution is nearly uniform over A' with tolerance $c \geq 1$ if $\Pr(a) = 0$ when $a \in A \setminus A'$ and*

$$\frac{1}{c|A'|} \leq \Pr(a) \leq \frac{c}{|A'|}$$

when $a \in A'$.

For any finite abelian group $G, +$ and tolerance $c \geq 1$, we formally define the problems RLD-s(G, c) and RLD-d(G, c), which are the search resp. decision versions of the homogeneous random linear disequations problem:

Definition 13. *RLD-s(G, c) is about finding any generator of a secret cyclic subgroup $\langle s \rangle \subseteq G$, given access to samples from a nearly uniform distribution with tolerance c over the subset $\{\chi \in G^\vee \mid \chi(s) \neq 1\} \subseteq G^\vee$.*

It should be clear from the definition that, indeed, one can only hope to find a generator of $\langle s \rangle$ rather than s itself. This directly relates to the fact that the corresponding linear disequations are homogeneous, see Remark 10.

Definition 14. *Given unlimited access to characters $\chi \in G^\vee$ which are consistently sampled from either*

- *a nearly uniform distribution with tolerance c over $\{\chi \in G^\vee \mid \chi(s) \neq 1\}$ for some fixed $s \in G \setminus \{0\}$, or*
- *a nearly uniform distribution with tolerance c over the entirety of G^\vee ,*

the RLD-d(G, c) problem is about deciding which is the case.

Of course, in our case, we will apply these definitions to the group

$$H = \mathbb{Z}_{p^{k_1}} \times \dots \times \mathbb{Z}_{p^{k_n}},$$

and the element s in the above problems will take the value of the corresponding component (s_1, \dots, s_n) of our hidden shift.

(a) **From finding (s_1, \dots, s_n) to RLD-s($H, 3$)** To sample from

$$H_{s_1, \dots, s_n}^\vee = \{ \chi \in H^\vee \mid \chi(s_1, \dots, s_n) \neq 1 \},$$

we use the following method. First, using the Kuperberg sieve from Section 5.1, we generate a phase vector $|\Psi(\chi)\rangle$ over H , where it is easy to check that $\chi \in H^\vee$ is uniformly random. We then measure this phase vector in the $|\pm\rangle$ -basis. When measuring ‘+’ we reject the sample and start over. When measuring ‘-’, we return χ^j for some uniformly random $j \in \{0, 1, \dots, p^k - 1\}$ that is coprime to p .

Note that the overall probability of measuring ‘-’ is

$$\frac{1}{|H|} \sum_{\chi \in H^\vee} \frac{|1 - \chi(s_1, \dots, s_n)|^2}{4} = \frac{1 - \delta_{(s_1, \dots, s_n), (0, \dots, 0)}}{2}.$$

If we fail to measure ‘-’ for (say) 128 consecutive times then with overwhelming probability $(s_1, \dots, s_n) = (0, \dots, 0)$ and we are done. Else, it follows from Bayes’ theorem that the above procedure samples $\chi \in H^\vee$ with probability

$$\frac{1}{2\varphi(p^k)|H|} \sum_{\substack{j=0 \\ \gcd(j,p)=1}}^{p^k-1} |1 - \chi^j(s_1, \dots, s_n)|^2$$

which equals 0 if $\chi(s_1, \dots, s_n) = 1$, i.e., if $\chi \notin H_{s_1, \dots, s_n}^\vee$, and is contained in the interval $[1/2|H|, 2/|H|]$ in the other case; see [21, Lem. 2]. Therefore the resulting distribution is nearly uniform over H_{s_1, \dots, s_n}^\vee with tolerance $2|H|/|H_{s_1, \dots, s_n}^\vee| \leq 2p/(p-1) \leq 3$. Thus, by solving RLD-s($H, 3$) we can find a generator of the cyclic group $\langle (s_1, \dots, s_n) \rangle$; note that there is a small error in the corresponding statement in Ivanyos’ paper [21, Prop. 1], who reduces to RLD-s($H, 2$) instead. Finding the exact value of (s_1, \dots, s_n) then amounts to exhaustive search over a set of size $|\langle (s_1, \dots, s_n) \rangle| \leq p^k$.

Remark 15. Test whether a guess $(\tilde{s}_1, \dots, \tilde{s}_n)$ is correct can be done as explained in [9, §3], by transforming phase vectors $|\Psi(\chi)\rangle$ into

$$|\Psi_s(\chi)\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \chi(\tilde{s}_1, \dots, \tilde{s}_n)^{-1} \chi(s_1, \dots, s_n) |1\rangle)$$

before measuring it in the $|\pm\rangle$ -basis. As soon as we measure ‘-’, the guess is wrong. If we fail to measure ‘-’ for (say) 128 consecutive times then the guess was correct with overwhelming probability.

(b) **From RLD-s($H, 3$) to RLD-d($S, 6$)** For any subgroup $S \subseteq H$, we obtain a distribution on S^\vee by restricting the domain of the characters from H to S . Depending on whether $(s_1, \dots, s_n) \in S$ or not, this distribution is nearly uniform over

$$S_{s_1, \dots, s_n}^\vee = \{ \chi \in S^\vee \mid \chi(s_1, \dots, s_n) \neq 1 \} \quad \text{or} \quad \text{the entirety of } S^\vee$$

where the tolerance doubles at worst; see [21, Lem.3]. This can be used to reduce $\text{RLD-s}(H, 3)$ to $O(p(k_1 + \dots + k_n))$ instances of $\text{RLD-d}(S, 6)$ for varying subgroups $S \subseteq H$, as follows. The first goal is to find a cyclic subgroup containing (s_1, \dots, s_n) . To this end, we will assume that H is non-cyclic; if H is already cyclic, we can skip the next paragraph.

We start by setting $S = H$ and repeat the following procedure. Choose an isomorphism

$$\iota : S \xrightarrow{\cong} \mathbb{Z}_{p^{k'_1}} \times \dots \times \mathbb{Z}_{p^{k'_r}} \quad (7)$$

with $r \in \{2, \dots, n\}$ and all k'_i positive. Pick any two indices $i, j \in \{1, \dots, r\}$ and consider the $p + 1$ index- p subgroups

$$S_{(\lambda_i : \lambda_j)} = \iota^{-1}\{(x_1, \dots, x_r) \in S \mid \lambda_i x_i + \lambda_j x_j \equiv 0 \pmod{p}\}$$

with $(\lambda_i : \lambda_j) \in \mathbb{P}^1(\mathbb{F}_p) = \{(a : 1) \mid a \in \mathbb{F}_p\} \cup \{(1 : 0)\}$. We distinguish between two cases:

- (i) if $\iota(s_1, \dots, s_n)$ has zero components at indices i and j then $(s_1, \dots, s_n) \in S_{(\lambda_i : \lambda_j)}$ for all $(\lambda_i : \lambda_j)$,
- (ii) if not, then $(s_1, \dots, s_n) \in S_{(\lambda_i : \lambda_j)}$ for exactly one $(\lambda_i : \lambda_j)$.

Using at most 2 calls of the form $\text{RLD-d}(S_{(\lambda_i : \lambda_j)}, 6)$ we can figure out whether we are in case (i) or (ii), and in the latter case at most $p - 2$ further calls identify the unique $(\lambda_i : \lambda_j)$ for which $(s_1, \dots, s_n) \in S_{(\lambda_i : \lambda_j)}$. In the former case we know

$$(s_1, \dots, s_n) \in \bigcap_{(\lambda_i : \lambda_j) \in \mathbb{P}^1(\mathbb{F}_p)} S_{(\lambda_i : \lambda_j)} = \iota^{-1}\{(x_1, \dots, x_r) \mid x_i \equiv x_j \equiv 0 \pmod{p}\}.$$

Thus we have replaced S by a subgroup of index p or p^2 .

Repeating this process eventually leads to a cyclic subgroup

$$S \cong \mathbb{Z}_{p^{k'}}$$

of H that contains (s_1, \dots, s_n) . This means that $\langle (s_1, \dots, s_n) \rangle = p^{i-1}S$, where $i \in \{1, \dots, k'\}$ is minimal such that a call to $\text{RLD-d}(p^i S, 6)$ reveals near uniformity over the entirety of $p^i S$.

(c) Reduction to the case of free modules Reconsider the isomorphism ι from (7) and write $k' = \max_i k'_i$. Through composition of ι with

$$\varepsilon : \mathbb{Z}_{p^{k'_1}} \times \dots \times \mathbb{Z}_{p^{k'_r}} \hookrightarrow \mathbb{Z}_{p^{k'}}^r : (x_1, \dots, x_n) \mapsto (x_1 p^{k'-k'_1}, \dots, x_r p^{k'-k'_r})$$

we can embed S in the free $\mathbb{Z}_{p^{k'}}$ -module $\mathbb{Z}_{p^{k'}}^r$. We turn our distribution on S^\vee into a distribution on $\mathbb{Z}_{p^{k'}}^{r\vee}$ as follows: for any sample χ we can write

$$\chi \circ \iota^{-1} : (x_1, \dots, x_r) \mapsto e^{2\pi i \left(\frac{a_1 x_1}{p^{k'_1}} + \dots + \frac{a_r x_r}{p^{k'_r}} \right)}$$

and we lift a_i to $\tilde{a}_i = a_i + fp^{k'_i}$ for some uniformly random $f \in \{0, \dots, p^{k'_i} - 1\}$ in order to end up with a character

$$\tilde{\chi} : (x_1, \dots, x_r) \mapsto e^{2\pi i \left(\frac{\tilde{a}_1 x_1 + \dots + \tilde{a}_r x_r}{p^{k'}} \right)}. \quad (8)$$

The resulting distribution is nearly uniform over either

$$\left\{ \psi \in \mathbb{Z}_{p^{k'}}^{r \vee} \mid \psi(\varepsilon(\iota(s_1, \dots, s_n))) \neq 1 \right\} \quad \text{or} \quad \text{all of } \mathbb{Z}_{p^{k'}}^{r \vee}$$

depending on whether the distribution on S^\vee was nearly uniform over S_{s_1, \dots, s_n}^\vee or all of S^\vee . The tolerance is not affected. Thus the calls to $\text{RLD-d}(S_{(\lambda_i: \lambda_j)}, 6)$ from above can be replaced with calls to $\text{RLD-d}(\mathbb{Z}_{p^{k'}}^r, 6)$.

(d) Solving RLD-d for free modules From now on we simply assume that

$$H = \mathbb{Z}_{p^k}^n \quad \text{and} \quad s = (s_1, \dots, s_n)$$

and we recall Ivanyos' method for solving $\text{RLD-d}(H, 6)$; in order to use this method in the above reduction, one needs to replace $k \leftarrow k'$, $n \leftarrow r$, $s \leftarrow \varepsilon(\iota(s))$. Along the way, we reduce the value $D = (p-1)((2p-2)^k - 1)/(2p-3)$ from Ivanyos' paper by roughly a factor 2^{k-1} .

Concretely, we let $D = p^k - 1$ and consider the space

$$V = \mathbb{Z}_p^D[x_{1,0}, \dots, x_{1,k-1}, \dots, x_{n,0}, \dots, x_{n,k-1}]$$

of polynomials in nk variables of total degree at most D , where each variable occurs in degree at most $p-1$; we can assume $D \leq nk(p-1)$. The dimension of V admits the crude estimate

$$\dim V \leq \binom{nk + D}{nk} = O(n^D)$$

(remember that p and k are treated as fixed constants). We refer the interested reader to [3, Thm. 5.5] for a precise formula for $\dim V$ and to [13] for alternative upper bounds obtained from Cramér's theorem. For every character

$$\chi : (x_1, \dots, x_n) \mapsto e^{2\pi i \left(\frac{a_1 x_1 + \dots + a_n x_n}{p^k} \right)} \quad (9)$$

that we sample, we add a new row to a matrix M with entries in \mathbb{Z}_p having $\dim V$ columns, as follows. By applying the base- p expansion map

$$\delta : \mathbb{Z}_{p^k} \rightarrow \mathbb{Z}_p^k : x_0 + x_1 p + \dots + x_{k-1} p^{k-1} \mapsto (x_0, x_1, \dots, x_{k-1})$$

component-wise to (a_1, \dots, a_n) we end up with a vector of length nk : the corresponding row then consists of the evaluations of the monomials in V at this vector. After sampling N characters, we have $M \in \mathbb{Z}_p^{N \times \dim V}$.

If our distribution is nearly uniform over the entirety of H^\vee , then the kernel of M describes polynomials in V that vanish at N nearly uniformly randomly sampled points of \mathbb{Z}_p^{nk} . Since V does not contain non-zero polynomials that vanish everywhere, this kernel must eventually become trivial as N grows. To estimate how large N must be taken, Ivanyos makes the following beautiful (but crude) reasoning: if $\ker M$ contains a non-zero polynomial P then this polynomial is non-vanishing in at least about $p^{nk-D/(p-1)}$ points, in view of [3, Cor. 5.26]. Therefore the probability that P gets removed when passing to the next sample, and therefore the probability that $\dim \ker M$ drops, is at least roughly

$$\frac{p^{nk-D/(p-1)}}{p^{nk}} = p^{-D/(p-1)}.$$

So, incorporating our tolerance $c = 6$, we can expect about $6p^{D/(p-1)} \dim V = O(n^D)$ samples to be sufficient for revealing that $\ker M = \{0\}$.

On the other hand, if all characters χ are non-vanishing at (s_1, \dots, s_n) then the kernel is never empty. We quickly recall the argument, while highlighting the source of the improved value of D : this comes from a sharp estimate on the degree of the *carry-polynomial*

$$C = \sum_{i=1}^{p-1} (1 - (x-i)^{p-1}) \sum_{j=p-i}^{p-1} (1 - (y-j)^{p-1}) \in \mathbb{Z}_p[x, y]$$

which for all $a, b \in \mathbb{Z}_p$ satisfies $C(a, b) = 1$ if $a+b \geq p$ and $C(a, b) = 0$ if $a+b < p$, thereby explaining its name. Ivanyos, who describes C using Lagrange basis polynomials, provided the naive bound $\deg C \leq 2p-2$, but from [20, Thm. 1] applied to $C(x+p-1, y)$ it follows that the degree is actually p . Through a repeated use of this carry-polynomial, for any positive integer T it is easy to construct polynomials $Q_i \in \mathbb{Z}_p[x_{1,0}, \dots, x_{1,k-1}, \dots, x_{T,0}, \dots, x_{T,k-1}]$ of degree at most $(\deg C)^i = p^i$ such that

$$\delta(a_1 + \dots + a_T) = \left(Q_0(\delta(a_1), \dots, \delta(a_T)), \dots, Q_{k-1}(\delta(a_1), \dots, \delta(a_T)) \right) \quad (10)$$

for all $a_1, \dots, a_T \in \mathbb{Z}_{p^k}$, see [21, Lem. 5]. Choosing $T = (p^k - 1)n$, for every tuple (a_1, \dots, a_n) coming from a character χ as in (9), we can use (10) to view

$$\delta(a_1 s_1 + \dots + a_n s_n) = \delta(\underbrace{a_1 + \dots + a_1}_{\times s_1} + \dots + \underbrace{a_n + \dots + a_n}_{\times s_n} + \underbrace{0 + \dots + 0}_{\times (T - s_1 - \dots - s_n)})$$

as the evaluation in $(\delta(a_1), \dots, \delta(a_n))$ of a tuple of fixed but unknown polynomials

$$P_0, \dots, P_{k-1} \in \mathbb{Z}_p[x_{1,0}, \dots, x_{1,k-1}, \dots, x_{n,0}, \dots, x_{n,k-1}],$$

of degrees satisfying $\deg P_i \leq p^i$. So we know that $\chi(s_1, \dots, s_n) \neq 1$ if and only if the polynomial P obtained from

$$\prod_{j=0}^{k-1} (P_j^{p-1} - 1)$$

by reduction mod $x_{1,0}^p - x_{1,0}, \dots, x_{n,k-1}^p - x_{n,k-1}$ vanishes at $(\delta(a_1), \dots, \delta(a_n))$. This is the desired non-zero element of V .

Remark 16. We can also view

$$\delta(a_1 s_1 + \dots + a_n s_n) = \delta(\underbrace{s_1 + \dots + s_1}_{\times a_1} + \dots + \underbrace{s_n + \dots + s_n}_{\times a_n} + \underbrace{0 + \dots + 0}_{\times (T - a_1 - \dots - a_n)})$$

as a tuple P'_0, \dots, P'_{k-1} of *known* polynomials evaluated in the unknown entries of $(\delta(s_1), \dots, \delta(s_n))$. The polynomial

$$\prod_{j=0}^{k-1} (P'_j - 1)$$

then serves as an analogue of (5): gathering enough such polynomials should allow one to recover the hidden shift (or rather the cyclic subgroup it generates) using Gröbner bases, or via linearization. We expect this to run in time $O(n^D)$, although a precise runtime analysis of this direct search approach seems hard.

Remark 17. As was suggested to us by Frederik Vercauteren, instead of using base- p expansions it may be enlightening to work with Witt vector expansions [31, §II.6], for which formulae for addition (i.e., analogues of the above polynomials Q_i) and multiplication have seen more systematic study. But we will not pursue this track here.

5.3 Kuperberg sieve, again

Once s_1, \dots, s_n are found, we can define $f'_0, f'_1 : \mathbb{Z}_{2^{t_1}} \times \dots \times \mathbb{Z}_{2^{t_m}} \rightarrow X$ by letting

$$f'_0(g_1, \dots, g_m) = f_0(g_1, \dots, g_m, 0, \dots, 0)$$

and

$$f'_1(g_1, \dots, g_m) = f_1(g_1, \dots, g_m, s_1, \dots, s_n).$$

This gives a new hidden shift problem with hidden shift (s'_1, \dots, s'_m) . We solve this by rerunning Kuperberg's sieve from Section 5.1. Concretely, we sieve until we obtain $(t-1)$ -divisible phase vectors. Measuring these in the $|\pm\rangle$ -basis results in linear disequations mod 2 in the least significant bits of s'_1, \dots, s'_m . Of course, these disequations can be seen as exact equations; also note that both '+' and '-' give rise to an equation. After solving this system of linear equations, we repeat this process for $(t-2)$ -divisible phase vectors, obtaining the second most significant bits. We continue until we have found all of (s'_1, \dots, s'_m) .

5.4 Complexity

The cost of determining (s_1, \dots, s_n) is dominated by the runs of the decision algorithm from Step 5.2(d) on the (free module versions of the) groups $S_{(\lambda_i: \lambda_j)}$ from Step 5.2(b). There are $O(n)$ such groups to be considered. In order to run the decision algorithm, we need to prepare

$$O(n^D) \text{ characters } \chi \in H_{s_1, \dots, s_n}^\vee \quad (11)$$

and transform them into characters

$$\tilde{\chi} \in \mathbb{Z}_p^{r \vee}, \quad r \leq n, k' \leq k$$

via the methods described in Steps 5.2(b-c); the costs of these transformations are largely dominated by the estimates that follow.

Once we have the characters $\tilde{\chi}$ at our disposal, we can build the $O(n^D) \times O(n^D)$ matrix M and compute its kernel, requiring $O(n^{D\omega})$ time and $O(n^{2D})$ space, where $\omega \approx 2.373$ denotes the Alman–Williams constant [2]. Note that the same characters (11) can be reused during each run of the decision algorithm. Each character (11) is obtained by generating $O(1)$ phase vectors over H via the Kuperberg sieve from Section 5.1 and proceeding as in Step 5.2(a). In turn, each phase vector over H requires us to combine $O(m^t)$ phase vectors over G , and this combination takes $O(m^t)$ quantum gates, $O(m)$ quantum space, $O(m^{t-1+\omega})$ classical work and $O(m^2)$ classical space. Finally, generating a phase vector over G costs two quantum Fourier transforms over G and one call to f_0, f_1 each.

If we measure the cost of the quantum Fourier transform by $O(\log^2 |G|)$ time and $O(\log |G|)$ space [27, §5.1], we arrive at the following overall estimates for retrieving (s_1, \dots, s_n) :

- $O(n^D m^t (m+n)^2)$ quantum gates, $O(m+n)$ qubits and $O(n^D m^t)$ oracle calls to f_0, f_1 ,
- $O(n^D m^{t-1+\omega} + nn^{D\omega})$ classical time and $O(m^2 + n^{2D})$ classical space.

The cost of determining (s'_1, \dots, s'_m) once (s_1, \dots, s_n) is found again amounts to the combination of $O(m^t)$ phase vectors via Kuperberg’s sieve, but now over the smaller group

$$\mathbb{Z}_{2^{t_1}} \times \dots \times \mathbb{Z}_{2^{t_m}}.$$

So this cost is dominated by the above estimates.

We stress that the implicit constants in the O -notations above strongly depend on p, k, t , which are treated as fixed values. Finally, revisiting Remark 16, we expect that a direct search variant would reduce the classical runtime from $O(n^D m^{t-1+\omega} + nn^{D\omega})$ to $O(n^D m^{t-1+\omega} + n^{D\omega})$.

5.5 Hidden shift finding in groups with large $2^t p^k$ -torsion

An almost word-by-word copy of the discussion from [9, §5] shows that the algorithm described in Section 5 naturally merges into Peikert’s “least-significant

bit first” variant [28] of Kuperberg’s collimation sieve [24]. More concretely, all one needs to do is make the following adjustments to [9, Alg. 3] and [9, Alg. 4]: replace each occurrence of p with p^k , and at the point where [9, Alg. 1] is invoked, call the algorithm from Section 5 instead. This yields Theorem 1.

6 Conclusion

In this paper, we have presented two unrelated addenda to the existing literature on cryptographic group actions.

The first addendum is the observation that, classically, the vectorization problem does not in general admit a polynomial-time (or even sub-exponential time) reduction to the parallelization problem. This contrasts with the quantum setting, where both problems were shown to be computationally equivalent [15, 26]. It also contrasts with the special case of exponentiation in finite cyclic groups, where convincing arguments in favour of the existence of a classical polynomial-time reduction were provided by Maurer and Wolf [25].

The second addendum is the remark that an algorithm due to Ivanyos [21] for solving systems of linear disequations over \mathbb{Z}_{p^k} (p prime, k a positive integer) can be combined with a Kuperberg-style sieve in order to obtain a polynomial-time quantum algorithm for solving the hidden shift problem in finite abelian $2^t p^k$ -torsion groups (t a positive integer, p, k, t fixed) that involves mostly classical work; in particular, the requirements in terms of quantum memory are very limited. This extends the observation from [9, §3] from $k = 1$ to arbitrary fixed values of k . Along the way, we fixed a small error in Ivanyos’ reduction and we provided a sharper complexity estimate. More importantly, we hope that this paper succeeds in bringing the intriguing problem of solving systems of linear disequations to the attention of a wider audience.

As in [9], our algorithm can be merged with Kuperberg’s collimation sieve into a single quantum algorithm for solving the hidden shift problem in any finite abelian group G in time

$$\text{poly}(\log |G|) \cdot 2^{O(\sqrt{\log |2^t p^k G|})},$$

where the main memory requirements are in terms of quantum read-only memory: only polynomially many qubits are needed. The consequences for group-action based cryptography are as discussed in [9, Ex. 2.3]: the vectorization problem is weakened by the presence of a large $2^t p^k$ -torsion group, however ideal-class groups, as used by Couveignes [12], Rostovtsev–Stolbunov [30] and in CSIDH [10], are well-protected against this, in view of the Cohen–Lenstra heuristics. Nevertheless, wariness of this potential weakness is advisable.

References

- [1] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In *Asiacrypt 2*, volume 12492 of *Lecture Notes in Computer Science*, pages 411–439. Springer, 2020.

- [2] Josh Alman and Virginia V. Williams. A refined laser method and faster matrix multiplication. In *SODA 2020*, pages 522–539. SIAM, 2021.
- [3] Edward F. Assmus, Jr. and Jennifer D. Key. Polynomial codes and finite geometries. In *Handbook of coding theory, Vol. I, II*, pages 1269–1343. 1998.
- [4] László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *STOC'16*, pages 684–697. ACM, New York, 2016.
- [5] Xavier Bonnetain and María Naya-Plasencia. Hidden shift quantum cryptanalysis and implications. In *Asiacrypt 1*, volume 11272 of *Lecture Notes in Computer Science*, pages 560–592. Springer, 2018.
- [6] Gilles Brassard and Claude Crépeau. Non-transitive transfer of confidence: A perfect zero-knowledge interactive protocol for SAT and beyond. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 188–195, 1986.
- [7] Gilles Brassard, Claude Crépeau, and Moti Yung. Everything in NP can be argued in perfect zero-knowledge in a bounded number of rounds. In *ICALP 1989*, volume 372 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 1989.
- [8] Gilles Brassard and Moti Yung. One-way group actions. In *CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 94–107. Springer, 1991.
- [9] Wouter Castryck, Ann Dooms, Carlo Emerencia, and Alexander Lemmens. A fusion algorithm for solving the hidden shift problem in finite abelian groups. In *PQCrypto*, volume 12841 of *Lecture Notes in Computer Science*, pages 133–153. Springer, 2021.
- [10] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In *Asiacrypt 3*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.
- [11] Mathilde Chenu-de La Morinerie. *Supersingular group actions and post-quantum key exchange*. PhD thesis, Institut Polytechnique de Paris, 2021.
- [12] Jean-Marc Couveignes. Hard homogeneous spaces, 1997. IACR Cryptology ePrint Archive, Report 2006/291, <https://ia.cr/2006/291>.
- [13] Jordan S. Ellenberg and Dion Gijswijt. On large subsets of \mathbb{F}_q^n with no three-term arithmetic progression. *Annals of Mathematics*, 185(1):339–343, 2017.
- [14] Katalin Friedl, Gábor Ivanyos, Frédéric Magniez, Miklos Santha, and Pranab Sen. Hidden translation and orbit coset in quantum computing. In *STOC'03*, pages 1–9. ACM, New York, 2003.
- [15] Steven Galbraith, Lorenz Panny, Benjamin Smith, and Frederik Vercauteren. Quantum equivalence of the DLP and CDHP for group actions. *Mathematical Cryptology*, 1(1):40–44, 2021.
- [16] Oliver W. Gnilke and Jens Zumbrägel. Cryptographic group and semigroup actions. In *WCC 2022, Designs, Codes and Cryptography*. Springer, to appear.
- [17] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *STOC'96*, pages 212–219. ACM, 1996.
- [18] Sean Hallgren. *Quantum Fourier sampling, the hidden subgroup problem, and beyond*. PhD thesis, University of California, Berkeley, 2000.
- [19] Harald Andrés Helfgott. Isomorphismes de graphes en temps quasi-polynomial [d’après Babai et Luks, Weisfeiler-Leman, . . .]. *Astérisque*, (407):135–182, 2019. Séminaire Bourbaki (exp. no. 1125).
- [20] Iliia Iliashenko and Vincent Zucca. Faster homomorphic comparison operations for BGV and BFV. In *PETS 2021 (3)*, pages 246–264. De Gruyter, 2021.
- [21] Gábor Ivanyos. On solving systems of random linear disequations. *Quantum Information & Computation*, 8(6-7):579–594, 2008.

- [22] Zhengfeng Ji, Youming Qiao, Fang Song, and Aaram Yun. General linear group action on tensors: A candidate for post-quantum cryptography. In *TCC 2019*, volume 11891 of *Lecture Notes in Computer Science*, pages 251–281. Springer, 2019.
- [23] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal of Computing*, 35(1):170–188, 2005.
- [24] Greg Kuperberg. Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In *TQC 2013*, volume 22 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20–34, 2013.
- [25] Ueli Maurer and Stefan Wolf. The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. *SIAM Journal on Computing*, 28(5):1689–1721, 1999.
- [26] Hart Montgomery and Mark Zhandry. Full quantum equivalence of group action DLog and CDH, and more. In *Asiacrypt*, Lecture Notes in Computer Science. Springer, 2022. To appear.
- [27] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- [28] Chris Peikert. He gives C-sieves on the CSIDH. In *Eurocrypt 2*, volume 12106 of *Lecture Notes in Computer Science*, pages 463–492, 2020.
- [29] Oded Regev. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space, 2004. Cornell University arXiv, <https://arxiv.org/abs/quant-ph/0406151>.
- [30] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies, 2006. IACR Cryptology ePrint Archive, Report 2006/145, <https://ia.cr/2006/145/>.
- [31] Jean-Pierre Serre. *Local fields*, volume 67 of *Graduate Texts in Mathematics*. Springer-Verlag, 1979. Translated from the French by Marvin Jay Greenberg.
- [32] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
- [33] Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.
- [34] Benjamin Smith. Pre- and post-quantum Diffie-Hellman from groups, actions, and isogenies. In *WAIFI 2018*, volume 11321 of *Lecture Notes in Computer Science*, pages 3–40, 2018.
- [35] Anton Stolbunov. *Cryptographic schemes based on isogenies*. PhD thesis, Norwegian University of Science and Technology, 2012.