

A Subexponential Quantum Algorithm for the Semidirect Discrete Logarithm Problem

Christopher Battarbee¹, Delaram Kahrobaei^{1,2,3,4}, Ludovic Perret⁵, and Siamak F. Shahandashti¹

¹ Department of Computer Science, University of York, UK

² Departments of Computer Science and Mathematics, Queens College, City University of New York, USA

³ Initiative for the Theoretical Sciences, Graduate Center, City University of New York, USA

⁴ Department of Computer Science and Engineering, Tandon School of Engineering, New York University, USA

⁵ Sorbonne University, CNRS, LIP6, PoSys, Paris, France

Abstract. *Group-based* cryptography is a relatively unexplored family in post-quantum cryptography, and the so-called Semidirect Discrete Logarithm Problem (SDLP) is one of its most central problems. However, the complexity of SDLP and its relationship to more well-known hardness problems, particularly with respect to its security against quantum adversaries, has not been well understood and was a significant open problem for researchers in this area. In this paper we give the first dedicated security analysis of SDLP. In particular, we provide a connection between SDLP and group actions, a context in which quantum subexponential algorithms are known to apply. We are therefore able to construct a subexponential quantum algorithm for solving SDLP, thereby classifying the complexity of SDLP and its relation to known computational problems.

Introduction

The goal of Post-Quantum Cryptography (PQC) is to design cryptosystems which are secure against classical and quantum adversaries. A topic of fundamental research for decades, the status of PQC drastically changed with the NIST PQC standardization process [10].

In July 2022, after five years and three rounds of selection, NIST selected a first set of PQC standards for Key-Encapsulation Mechanism (KEM) and Digital Signature Scheme (DSS) protocols, based on lattices and hash functions. The standardization process is still ongoing with a fourth round for KEM and a new NIST call for post-quantum DSS in 2023. Recent attacks [5, 32, 1] against round-3 multivariate signature schemes, *Rainbow* [5] and *GeMSS* [7], as well as the cryptanalysis of round-4 isogeny based KEM *SIKE* [8, 24], emphasise the need to continue the cryptanalysis effort in PQC as well as to increase the diversity in the potential post-quantum hard problems.

A relatively unexplored family of such problems come from *group-based* cryptography (see [20]). In particular we are interested in the so-called Semidirect Discrete Logarithm Problem (SDLP), which initially appears in the 2013 work [16] of Habeeb et al. Roughly speaking, we generalise the standard notion of group exponentiation by employing products of the form $\phi^{x-1}(g) \cdot \dots \cdot \phi(g) \cdot g$, where g is an element of a (semi)group, ϕ is an endomorphism and $x \in \mathbb{N}$ is a positive integer. Our task in SDLP is to recover the integer x given the pair g, ϕ and the value $\phi^{x-1}(g) \cdot \dots \cdot \phi(g) \cdot g$. It turns out that products of this form have enough structure to be cryptographically useful, in a sense we will expand upon later - in particular, protocols based on SDLP are plausibly post-quantum, since there is no known reduction of SDLP to a Hidden Subgroup Problem.

By far the most studied such protocol is known as Semidirect Product Key Exchange (SPDKE), originally proposed in [16] (note that this is the same work in which SDLP first appears). It is a Diffie–Hellman-like key exchange protocol in which products of the form $\phi^{x-1}(g) \cdot \dots \cdot \phi(g) \cdot g$ are exchanged between two parties in such a way as to allow both parties to recover the same shared key. Clearly, the security of SPDKE and the difficulty of SDLP are heavily related - in particular, an adversary able to solve SDLP is also able to break SPDKE.

There is therefore motivation to analyse the difficulty of SDLP. However, the complexity of SDLP and its relationship to more well-known hardness problems, particularly with respect to its security against quantum adversaries, has not been well understood and was a significant open problem for researchers in this area. In this paper we provide the first dedicated analysis of SDLP, obtaining two key contributions. First, we demonstrate that a subset of all possible products of the form $\phi^{x-1}(g) \cdot \dots \cdot \phi(g) \cdot g$ is a set upon which a finite abelian group acts; in other words, that SPDKE is, modulo some context-specific technicality, a variant of the group action-based key exchange schemes originally proposed by Couveignes [14]. In particular, solving SDLP can be translated into a problem with respect to a group action. This surprising connection provides a sharper classification of SPDKE than was previously known, and allows us to derive our second contribution, an application of known tools that gives a quantum algorithm for solving SDLP. The algorithm runs in subexponential time $2^{\mathcal{O}(\sqrt{\log p})}$, where p is the security parameter.

Related Work

Examples of concrete proposals for SPDKE can be found in [16, 15, 21, 28, 29]; respective cryptanalyses can be found in [27, 19, 6, 25, 3, 4]. This body of work proceeds more or less chronologically, in that proposed platforms are a response to cryptanalysis addressing a weakness in an earlier version. As a brief summary: the platform proposed in the first version of the scheme [16] is the multiplicative semigroup of a matrix algebra formed of 3×3 matrices with entries in a group ring. It was pointed out in [27] that the unused addition operation of the matrix algebra was actually a vulnerability allowing for complete shared key recovery, and that moreover any group with a sufficiently low-dimensional representation as a matrix algebra would be vulnerable to a similar kind of attack. In response,

in [21] certain classes of p -groups are proposed, since they admit only extremely high-dimensional representations. Another method of counteracting this linear vulnerability is to mix operations: in [28] the matrix algebra $M_3(\mathbb{Z}_p)$ is employed for some prime p in such a way that both matrix multiplication and matrix addition are called upon. It turns out, however, that this mixing of operations opens the scheme up to a different type of attack [6, 3] based on the so-called ‘telescoping equality’. It appears at time of writing that one can get around this vulnerability (as argued in [4]) by removing some structure; in particular, using matrices over a semiring rather than a ring. An earlier attempt at utilising a semiring is made in [15], though this particular case turned out to admit full shared key recovery due to a partial order allowing for more efficient search algorithms. A more detailed survey of the back-and-forth on this topic can be found in [2].

Despite the relationship between SPDKE and SDLP, none of the works discussed above provide an analysis of SDLP. Indeed, the general direction of research in this area has been either to achieve shared key recovery by exploiting some underlying linearity of a platform (semi)group, or to find examples of (semi)groups with sufficiently lax structure to render these attacks less powerful. In particular, none of the cryptanalyses in this area solve SDLP.

Ideas much closer to the spirit of our work appear in papers that, at first glance, appear unrelated to SPDKE and SDLP. Our results are achieved in part by careful synthesis of the techniques in the two papers [11, 12]: since the set of all products of the form $\phi^{x-1}(g) \cdot \dots \cdot \phi(g) \cdot g$ admits some similarity to that of a monogenic semigroup, we can adapt some ideas from a quantum algorithm in [12] that solves the Semigroup Discrete Logarithm Problem. However, in our setting we are lacking some key structure that allows the direct application of [12]. The full algorithm is constructed by adapting ideas in [11], allowing us to show the important quantum algorithms of Kuperberg [23] and Regev [30] can be used to solve SDLP.

Finally, we note that the connection to group actions alluded to above allows the application of a recent landmark result of Montgomery and Zhandry [26]. The implications of this result on our work are discussed in Section 3.3.

Organisation of the Paper and Main Results

The construction of the algorithm claimed in the title is, from a high-level perspective, achieved by two reduction proofs followed by an application of known algorithms. With this in mind, we make the following contributions.

Section 1. We start with preliminaries in which the necessary background is reviewed. In particular we give a brief discussion of the relevant algebraic objects, a short note on quantum computation, and a full description of SPDKE - including a discussion of appropriate asymptotic assumptions for the growth of the semigroups we work with, derived from currently proposed platforms. In particular we derive a notion of a parametrised family of semigroups called

an ‘easy’ family of semigroups. The section finishes with a kind of glossary of computational problems.

Section 2. It will be immediately convenient to write $s(g, \phi, x)$ for $\phi^{x-1}(g) \cdots g$, not only for clarity of notation but to aid the crucial shift in perspective offered in this paper. Armed with this notation our first task is to study the set of possible exponents $\{s(g, \phi, i) : i \in \mathbb{N}\}$. In Theorem 1 we deduce, borrowing from some standard ideas in semigroup theory, that this set is finite and has the form $\{g, \dots, s(g, \phi, n), \dots, s(g, \phi, n+r-1)\}$ where the integers n, r are a function of the choice of (g, ϕ) (and may, when desirable, be written $r_{g, \phi}, n_{g, \phi}$ to highlight this point). The main result of this section is Theorem 3: an abelian group acts freely and transitively on the set $\{s(g, \phi, n), \dots, s(g, \phi, n+r-1)\}$. This set is called the cycle of g, ϕ and is denoted by the calligraphic letter \mathcal{C} . In particular, we show the following:

Theorem. *Fix $(g, \phi) \in G \times \text{End}(G)$ and let n, r be the index and period corresponding to g, ϕ . Moreover, let \mathcal{C} be the corresponding cycle of size r . The abelian group \mathbb{Z}_r acts freely and transitively on \mathcal{C} .*

where \mathbb{Z}_r is the usual notion of a group of residues modulo r .

Section 3. If the set of exponents with respect to (g, ϕ) consisted entirely of the cycle we would immediately have a reduction to the Group Action Discrete Logarithm Problem (GADLP), also referred to as *Group Action DLog* in [26], or the *Parallelisation Problem* in [14]. Roughly speaking, since we know that an abelian group acts on the cycle, in this case the exponent x to be recovered is precisely the group element acting on g to give $s(g, \phi, x)$. This is not, however, generally the case, and to proceed it will be necessary to extract the pair n, r from the base values g, ϕ . In Theorem 4 we show that one can achieve this in efficient quantum time by using canonical quantum period-finding methods. We can therefore deduce in Theorem 5 that one can solve SDLP efficiently given access to a GADLP oracle; or, if $s(g, \phi, x)$ is not in the cycle of g, ϕ , by invoking a classical procedure exploiting the knowledge of n, r . Indeed, we show the following:

Theorem. *Let $\{G_p\}_p$ be an easy family of semigroups, and fix p . Algorithm 3 solves SDLP with respect to a pair $(g, \phi) \in G_p \times \text{End}(G_p)$ given access to a GADLP oracle for the group action $(\mathbb{Z}_{r_{g, \phi}}, \mathcal{C}_{g, \phi}, \otimes)$. The algorithm runs in time $\mathcal{O}((\log p)^4)$, makes at most a single query to the GADLP oracle, and succeeds with probability $\Omega(1)$.*

Clearly, many of the requisite notions in this statement have not yet been defined. Roughly speaking, an easy family of semigroups is a family of semigroups parameterised by p such that each of the functions taking p as an argument grows polynomially in p .

Section 4. In order to give a full description and complexity analysis of the algorithm it remains to examine the state of the art for solving GADLP. It is reasonably well-known (see [31], [11]) that GADLP reduces to the Abelian Hidden Shift Problem if the action is free and transitive, though we provide a context-specific reduction in Theorem 6. There are two popular choices to solve this problem: an algorithm due to Kuperberg [23] and another due to Regev [30], each of which has trade-offs with respect to time and space complexity. Finally, the full algorithm is given in Theorem 9, though we are essentially assembling the components we have developed throughout the rest of the paper. This main result is the following:

Theorem. *Let $\{G_p\}_p$ be an easy family of semigroups, and fix p . For any pair $(g, \phi) \in G_p \times \text{End}(G_p)$, there is a quantum algorithm solving SDLP with respect to (g, ϕ) with time and query complexity $2^{\mathcal{O}(\sqrt{\log p})}$.*

which proves the claim of a quantum subexponential algorithm for SDLP given in the title.

1 Preliminaries

1.1 Notation

One need only be familiar with the standard $\mathcal{O}()$ and $\Omega()$ notations. Various bespoke notations are introduced throughout the course of the paper, but these will be defined in due course. We note also that all our logarithms are base-2.

1.2 Background Mathematics

We recall a number of group-theoretic notions used throughout this paper. Recall that a group for which one is not guaranteed to have inverse with respect to the group operation is a *semigroup*. Writing the operation multiplicatively, the semigroups G we are interested in all have an element 1 such that $1 \cdot g = g = g \cdot 1$ for each $g \in G$ - such a semigroup is also called a *monoid*. In this work we insist that we do not have a full group, and so write semigroups without meaning to refer to full groups.

We will deal with both abelian groups and non-abelian semigroups in this paper; that is, for a non-abelian semigroup G one cannot expect that $g \cdot h = h \cdot g$ for all $g, h \in G$. For the sake of clarity we will write abelian groups additively. In this case, the operation $g + h$ commutes, and we require an inverse for every element. Note also that the identity is written as 0 in this case.

Consider a function from G to itself, say ϕ . If ϕ preserves multiplication - that is, $\phi(g \cdot h) = \phi(g) \cdot \phi(h)$ for each $g, h \in G$ - we call it an *endomorphism*. Certainly we can compose these functions according to the usual notion, and indeed it is standard that the set of all endomorphisms under function composition defines a semigroup. Since we allow for (and in some cases require) that the endomorphisms are not invertible, we have a semigroup rather than a full group.

In particular, every finite semigroup G immediately induces an *endomorphism semigroup*, denoted $End(G)$.

An important, and in this context frequently invoked source of semigroups come from *matrix algebras*. The set of square matrices of fixed size with entries in some ring R forms an R -module, since we can add matrices together and scale each entry of a matrix by some $r \in R$. The necessary distributivity properties are inherited from the properties of R . However, unlike in a usual R -module, we can also *multiply* elements just by defining multiplication to be the usual notion of matrix multiplication. The resulting matrix algebra is denoted $M_n(R)$, where $n \in \mathbb{N}$ is the fixed size of matrix, and R is the underlying ring. Indeed, consider a matrix algebra under only the multiplication operation. It is again clear that this object is a semigroup; we would have a full group if every matrix was invertible, but of course this is not true. The all-zero matrix, for example, has no multiplicative inverse. A matrix algebra considered only under its multiplication, therefore, is a useful source for concrete examples of semigroups.

It will be useful for us to build a new semigroup from an existing semigroup. One way of doing this is via a structure called the *holomorph*. Let G be a (semi)group and $End(G)$ its endomorphism semigroup. The holomorph is the set $G \times End(G)$ equipped with multiplication

$$(g, \phi) \cdot (g', \phi') = (\phi'(g) \cdot g', \phi \circ \phi')$$

where \circ refers to function composition. In fact, the holomorph is itself a special case of the semidirect product, hence the terminology.

Group Actions. A key idea for us will be that of a group action, and in particular a commutative group action. Roughly speaking such an object allows one to map elements of a set to each other in a cryptographically useful fashion, but in a less structured manner than in more classical settings. More formally:

Definition 1 (Commutative Group Action). *Let G be a finite abelian group and X be a finite set. Consider a function from $G \times X \rightarrow X$, written by convention as $g \star x$, with the following properties:*

1. $1 \star x = x$
2. $(g + h) \star x = g \star (h \star x)$

The tuple (G, X, \star) is a commutative group action. If only the identity fixes an arbitrary element of X the action is free, and if for any $x, y \in X$ there is a g such that $g \star x = y$ the action is transitive.

The group action defined in this paper is commutative, so we will sometimes just write ‘group action’ to mean a commutative group action. It will remain for us, however, to prove that this action is free and transitive. If the action is indeed free and transitive, it follows that for any $x \in X$, all $y \in X$ are such that there exists a unique $g \in G$ with $g \star x = y$. Borrowing notation from Couveignes [14], it will sometimes be convenient for us to write $\delta(y, x)$ to denote this value.

1.3 Quantum Computation

In order to present our quantum algorithm for SDLP (Section 3), the reader needs only be familiar with standard quantum tools, presented for example in [17]. We give a brief summary of the required notions below.

Recall that n qubits can be represented by the complex vector space H_{2^n} , where the basis states are exactly the n -fold tensor products of basis states of H_2 . An ordered system of n qubits is called a *quantum register of length n* , and the basis states are sometimes written $\{|i\rangle : 0 \leq i < 2^n\}$ by identifying i with its binary representation.

We say a state $\mathbf{z} \in H_{2^n}$ is *entangled* if it cannot be written as the tensor product of H_2 states. In particular, an observation of one qubit in an entangled state affects the state of the other: suppose integers $\{0, \dots, M-1\}$ can be represented by a quantum register of length l for some $l, M \in \mathbb{N}$, and moreover that a function f on $\{0, \dots, M-1\}$ is such that $\{f(0), \dots, f(M-1)\}$ can be represented similarly. A state of the form

$$\frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle |f(j)\rangle$$

is such that when observation of the second register gives some $Y \in \{f(0), \dots, f(M-1)\}$, the first register is left in superposition

$$\frac{1}{\sqrt{L}} \sum_{j:f(j)=Y} |j\rangle$$

where L is the number of $j \in \{0, \dots, M-1\}$ such that $f(j) = Y$. The factor $1/\sqrt{L}$ is to normalise the probabilities, ensuring the state is a unit vector.

Finally, recall that we can create the uniform superposition efficiently with a Hadamard gate. For an l -qubit register, the computational basis vector $|0\rangle$ is such that the Hadamard gate (written H_{2^l}) is such that

$$H_{2^l} |0\rangle = \frac{1}{\sqrt{2^l}} \sum_{i=0}^{2^l-1} |i\rangle$$

Moreover, this transformation can be carried out efficiently, in time $\mathcal{O}(l)$.

1.4 Semidirect Product Key Exchange

We here define in full SPDKE. One verifies by induction that holomorph exponentiation takes the form

$$(g, \phi)^x = (\phi^{x-1}(g) \cdot \dots \cdot \phi(g) \cdot g, \phi^x)$$

where ϕ^x denoted the endomorphism ϕ composed with itself x times. Note that this operation involves multiplying (semi)group elements, endomorphisms, and applying an endomorphism to a semigroup element. If all these operations are

efficient, the holomorph exponentiation is efficient since one can apply standard square-and-multiply techniques.

The central idea of SPDKE is to use products of these form as a generalisation of Diffie–Hellman Key-Exchange. Suppose N is the number of all possible distinct holomorph exponents - there are finitely many - then the protocol works as follows:

1. Suppose Alice and Bob agree on a public (semi)group G and hence the integer N , as well as a group element g and endomorphism of G , say ϕ .
2. Alice picks a secret integer x uniformly at random from $\{1, \dots, N\}$, and calculates the holomorph exponent $(g, \phi)^x = (A, \phi^x)$. She sends **only** A to Bob.
3. Bob similarly calculates (B, ϕ^y) corresponding to a random, private integer y , and sends only B to Alice.
4. With her private automorphism ϕ^x Alice can now calculate her key as the group element $K_A = \phi^x(B) \cdot A$; Bob similarly calculates his key $K_B = \phi^y(A) \cdot B$.

We have

$$\begin{aligned} \phi^x(B) \cdot A &= \phi^x(\phi^{y-1}(g) \cdot \dots \cdot g) \cdot (\phi^{x-1}(g) \cdot \dots \cdot g) \\ &= (\phi^{x+y-1} \cdot \dots \cdot \phi^x(g)) \cdot (\phi^{x-1}(g) \cdot \dots \cdot g) \\ &= (\phi^{x+y-1} \cdot \dots \cdot \phi^y(g)) \cdot (\phi^{y-1}(g) \cdot \dots \cdot g) \\ &= \phi^y(A) \cdot B \end{aligned}$$

so $K := K_A = K_B$. Note that $A \cdot B \neq K$ as a consequence of our insistence that the group operation is non-commutative.

Writing these products in full will quickly become rather cumbersome. We therefore introduce some non-standard notation, which is useful both for convenience of exposition and the required shift in perspective we will introduce in this paper.

Definition 2. *Let G be a finite, non-commutative (semi)group, $g \in G$, and $\phi \in \text{End}(G)$. We define the following function:*

$$\begin{aligned} s : G \times \text{End}(G) \times \mathbb{N} &\rightarrow G \\ (g, \phi, x) &\mapsto \phi^{x-1}(g) \cdot \dots \cdot \phi(g) \cdot g \end{aligned}$$

Notice that when g, ϕ are fixed - as in the case of the key exchange - the function s is really only taking integer arguments, analogously to the standard notion of group exponentiation. Indeed, a passive adversary observing a round of SPDKE has access to the values $s(g, \phi, x)$ and $s(g, \phi, y)$ - in order to recover the shared key $s(g, \phi, x+y)$ one strategy they might adopt is to recover the private integers x, y from $s(g, \phi, x), s(g, \phi, y)$ to allow calculation of $s(g, \phi, x+y)$. In short, the security of SPDKE is clearly in some sense related to the Semidirect Discrete Logarithm Problem alluded to in the introduction. We shall have much more to say about this later on.

1.5 Efficiency Considerations

The works discussed in the introduction, as well as the contents of the more comprehensive survey [2], highlight that every extant proposal of a platform for SPDKE suggests for use some variety of matrix algebra. In particular, insofar as parameters are recommended, the convention is to fix a matrix size - usually 3 - and adjust the size of an underlying ring in order to increase security. One very good reason for this is that each scheme requires a user to repeatedly perform matrix multiplication, the complexity of which scales exponentially with the size of the matrix. Table 1 gives examples of platforms over 3×3 matrices, the size of the platform, and the variable that can be considered the security parameter⁶.

Table 1. Growth of Proposed Platforms

Proposed Platform	Size of Platform	Security Parameter
$M_3(G[R])$	$ R ^{ G }$	$ R $
Certain classes of p -group	Polynomial in prime p	Prime p
$M_3(\mathbb{Z}_p)$	p^9	p

In other words, having defined SPDKE above relative to some semigroup G and its endomorphism semigroup $End(G)$, we can think of each such semigroup as one of a family of semigroups $\{G_p\}_p$, where the family $\{G_p\}_p$ is parameterised by a security parameter p . Note that this immediately induces a family of endomorphism semigroups $\{End(G_p)\}_p$, so we can talk about pairs (g, ϕ) from the set $G_p \times End(G_p)$ for each p . Moreover, each semigroup in this family has size polynomial in p .

Another characteristic of the currently proposed platforms is that the endomorphisms suggested for use with SPDKE typically involve multiplication by one or more auxiliary matrices; that is, for a particular semigroup G_p , if $(g, \phi) \in G_p \times End(G_p)$ the group element $\phi(g)$ has the form $A \cdot g \cdot B$, where $A, B \in G$ are fixed. By the above discussion each application of ϕ therefore requires some constant number of operations in the underlying ring of the matrix semigroup, which we may assume has size polynomial in p . The complexity of this matrix multiplication will be dominated by the multiplication in the underlying ring. Since the size of the underlying ring is also polynomial in p , each multiplication has complexity $\mathcal{O}((\log p)^2)$ (since $bigO \log poly(p) = \mathcal{O}(\log p)$). We conclude that both multiplication of element in G_p , and evaluation of $\phi(g)$, can be done in time $\mathcal{O}((\log p)^2)$.

With these observations in mind, we define the following:

Definition 3. Let P some countable indexing set. A family of semigroups $\{G_p\}_{p \in P}$ is said to be easy if

⁶ Note here that $|R|$ is chosen as the parameter for reasons of efficiency of representation.

1. $|G_p|$ grows monotonically and polynomially in p
2. For any p , any tuple $(g, h, \phi) \in G_p \times G_p \times \text{End}(G_p)$ is such that $g \cdot h$ and $\phi(g)$ can be evaluated in time $\mathcal{O}((\log p)^2)$.

Many of the complexity results within the paper assume that we are dealing with an easy family of semigroups, basically in an attempt to model the behaviour of suggested examples of semigroup family.

1.6 Computational Problems

We have already alluded to some of the hard problems to be found in this paper. Here we give full definitions of all of them to serve as a kind of ‘glossary’ section.

Definition 4 (Semidirect Discrete Logarithm Problem). *Given a public (semi)group G , its public endomorphism semigroup $\text{End}(G)$ and a public pair $(g, \phi) \in G \times \text{End}(G)$, let N be the size of the set $\{s(g, \phi, i) : i \in \mathbb{N}\}$. Choose x from $\{1, \dots, N\}$ uniformly at random, calculate $s(g, \phi, x)$ and create the pair $((g, \phi), s(g, \phi, x))$. The Semidirect Discrete Logarithm Problem (SDLP) with respect to (g, ϕ) is to recover the integer x given the pair (g, ϕ) and $s(g, \phi, x)$.*

Definition 5 (Semidirect Computational Diffie–Hellman). *Given a public (semi)group G , its public endomorphism semigroup $\text{End}(G)$ and a public pair $(g, \phi) \in G \times \text{End}(G)$, let N be the size of the set $\{s(g, \phi, i) : i \in \mathbb{N}\}$. Choose x, y from $\{1, \dots, N\}$ uniformly at random, compute the values $(s(g, \phi, x), s(g, \phi, y), s(g, \phi, x+y))$, and create the tuple $((g, \phi), s(g, \phi, x), s(g, \phi, y))$. The Semidirect Computational Diffie–Hellman problem (SCDH) with respect to (g, ϕ) is to recover the value $s(g, \phi, x+y)$.*

The cognoscenti will notice that this version of a Computational Diffie–Hellman-type problem is defined slightly differently than equivalent variants found in this area. We choose this particular weaker form to illustrate the connection to SPDKE. We now give the computational problems that we seek to reduce to. These versions of the problems are taken from [26], but can be found as the *vectorisation* and *parallelisation* problems, respectively, in [14].

Definition 6 (Group Action Discrete Logarithm). *Given a public commutative group action (G, X, \star) , sample $g \in G$ and $x \in X$ uniformly at random, compute $y = g \star x$ and create the pair (x, y) . The Group Action Discrete Logarithm Problem (GADLP) with respect to x is to recover g given the pair (x, y) .*

Definition 7 (Group Action Computational Diffie–Hellman). *Given a public commutative group action (G, X, \star) , sample $g, h \in G$ and $x \in X$ uniformly at random. Compute $y = g \star x$ and $z = h \star x$, and create the tuple (x, z, y) . The Group Action Computational Diffie–Hellman problem (GACDH) is to recover $(g+h) \star y$ given the tuple (x, y, z) .*

These versions of GADLP and GACDH are slightly weaker than those found in [26], which allow for x to be chosen according to *any* distribution. For the purposes of our reduction, the uniform distribution will do. Note also that in our reductions, SDLP and SCDH are defined relative to some fixed pair (g, ϕ) , which will induce some instances of GADLP, GACDH in which the value $x \in X$ is *not* sampled uniformly at random and is instead fixed. We get around this by observing that for each $x \in X$, fixing x defines a GADLP (resp. GACDH) problem with respect to this x . Clearly, a GADLP (resp. GACDH) oracle for the group action (G, X, \star) can solve the GADLP (resp. GACDH) problem with respect to x for each $x \in X$. The distinction is sufficiently subtle for us to suppress this detail for the remainder of the paper. Finally we give a seemingly unrelated problem requiring a small amount of introduction. Let $f, g : A \rightarrow S$ be injective functions, where S is a set and A is a finite abelian group. We say that f, g *hide* some $s \in A$ if one has $g(a) = f(a + s)$ for each $a \in A$.

Definition 8 (Abelian Hidden Shift Problem). *Given a public abelian group A and a set S , suppose two injective functions f, g hide some $s \in A$. The Abelian Hidden Shift Problem (AHSP) is to recover the group element s .*

2 Structure of the Exponents

All of the algorithms in this paper rely on the construction of a certain group action - recall that such an object consists of a group, a set, and a function (Section 1.2, Definition 1). As a general outline to our strategy, we first define and deduce properties of a particular set, from which the appropriate group and function will follow.

With this in mind, we make the following definition. For now we will dispense with our notion of an easy, parameterised family of semigroups, since the results presented in this section apply to any fixed semigroup. In fact, for compactness of exposition, for the remainder of this section by G we mean an arbitrary finite (semi)group, and by $End(G)$ we mean its associated endomorphism semigroup.

Definition 9. *For a pair $(g, \phi) \in G \times End(G)$, define*

$$\mathcal{X}_{(g, \phi)} := \{s(g, \phi, i) : i \in \mathbb{N}\}$$

We will often write $\mathcal{X}_{(g, \phi)}$ as \mathcal{X} when clear from context. Certainly this object is neither a group nor a semigroup - numerous counterexamples can be found whereby multiplication of elements in this set are not contained in the set - but we can make some progress by borrowing from the standard theory of monogenic semigroups; presented, for example, in [18]. Since $\mathcal{X} \subset G$, \mathcal{X} is finite - the set $\{x \in \mathbb{N} : \exists y \ s(g, \phi, x) = s(g, \phi, y)\}$ must therefore be non-empty, else it is in bijection with the natural numbers. We may therefore choose its smallest element, say n . By definition of n the set $\{x \in \mathbb{N} : s(g, \phi, n) = s(g, \phi, n + x)\}$ must also be non-empty, so we may again pick its smallest element and call it r .

The structure of \mathcal{X} is further restricted by the following result:

Lemma 1. *Let $(g, \phi) \in G \times \text{End}(G)$ and $x, y \in \mathbb{N}$, then*

$$\phi^x(s(g, \phi, y)) \cdot s(g, \phi, x) = s(g, \phi, x + y)$$

Proof. Note that $s(g, \phi, x + y) = \phi^{x+y-1}(g) \dots g$. Since ϕ preserves multiplication, applying ϕ^x to $s(g, \phi, y)$ adds x to the exponent of each term. Multiplication on the right by $s(g, \phi, x)$ then completes the remaining terms of $s(g, \phi, x + y)$. \square

Remark 1. One can entirely symmetrically swap the roles of x and y in the above argument, which gives two ways of calculating $s(g, \phi, x + y)$. In essence, therefore, this result gives us a slightly more elegant proof of the correctness of SPDKE.

This method of inducing addition in the integer argument of s is sufficiently important that we will invoke a definition for it.

Definition 10. *Let $(g, \phi) \in G \times \text{End}(G)$ and define a function $f : \mathbb{N} \times \mathcal{X} \rightarrow \mathcal{X}$ by*

$$f(i, s(g, \phi, j)) = \phi^i(s(g, \phi, j)) \cdot s(g, \phi, i)$$

where $f(i, s(g, \phi, j))$ may also be written as $i * s(g, \phi, j)$.

Remark 2. If G is of the type discussed in Section 1.5, the value $i * s(g, \phi, j)$ can be computed in time $\mathcal{O}(\log i)$. This is because computing $\phi^i(s(g, \phi, j))$ requires calculating some fixed number of G -elements to the power i and multiplying, which can be done with $\mathcal{O}(\log i)$ operations by square and multiply; and, as we have seen, computing $s(g, \phi, i)$ requires $\mathcal{O}(\log i)$ operations.

Thus far we have established that corresponding to any fixed pair $(g, \phi) \in G \times \text{End}(G)$ is a set $\mathcal{X}_{g, \phi} = \mathcal{X}$ and a pair of integers n, r . By Lemma 1 we know that $i * s(g, \phi, j) = s(g, \phi, i + j)$ for any $i, j \in \mathbb{N}$, so by definition of n, r we have

$$\begin{aligned} s(g, \phi, n + 2r) &= r * s(g, \phi, n + r) \\ &= r * s(g, \phi, n) \\ &= s(g, \phi, n + r) = s(g, \phi, n) \end{aligned}$$

We conclude, by extending this argument in the obvious way, that $s(g, \phi, n + qr) = s(g, \phi, n)$ for each $q \in \mathbb{N}$. In fact, we have the following:

Lemma 2. *Fix $(g, \phi) \in G \times \text{End}(G)$ and let n, r be the corresponding integer pair as above. One has that*

$$s(g, \phi, n + x + qr) = s(g, \phi, n + x)$$

for all $x, q \in \mathbb{N}$.

We will frequently invoke Lemma 2. Indeed, we immediately get that the set \mathcal{X} cannot contain values other than $\{g, \dots, s(g, \phi, n), \dots, s(g, \phi, n + r - 1)\}$. If any of the values in $\{g, \dots, s(g, \phi, n - 1)$ are equal we contradict the minimality of n , and if any of the values in $\{s(g, \phi, n), \dots, s(g, \phi, n + r - 1)\}$ are equal we contradict the minimality of r . We have shown the following:

Theorem 1. Fix $(g, \phi) \in G \times \text{End}(G)$. The set $\mathcal{X} = \{s(g, \phi, i) : i \in \mathbb{N}\}$ has size $n + r - 1$ for integers n, r dependent on g, ϕ . In particular

$$\mathcal{X} = \{g, \dots, s(g, \phi, n), \dots, s(g, \phi, n + r - 1)\}.$$

We refer to the set $\{g, \dots, s(g, \phi, n - 1)\}$ as the *tail*, written $\mathcal{T}_{g, \phi}$, of $\mathcal{X}_{g, \phi}$; and the set $\{s(g, \phi, n), \dots, s(g, \phi, n + r - 1)\}$ as the *cycle*, written $\mathcal{C}_{g, \phi}$, of $\mathcal{X}_{g, \phi}$. The values $n_{g, \phi}$ and $r_{g, \phi}$ are called the *index* and *period* of the pair (g, ϕ) . We shall feel free to omit the subscript at will when clear from context.

One can see that unique natural numbers correspond to each element in the tail, but infinitely many correspond to each element in the cycle. In fact, each element of the cycle corresponds to a unique residue class modulo r , shifted by the index n . This is a rather intuitive fact, but owing to its usefulness we will record it formally. In the following we assume the function `mod` returns the canonical positive residue.

Theorem 2. Fix $(g, \phi) \in G \times \text{End}(G)$ and let $x, y \in \mathbb{N}$. We have

$$s(g, \phi, n + x) = s(g, \phi, n + y)$$

if and only if $x \bmod r = y \bmod r$.

Proof. In the reverse direction, setting $x' = x \bmod r$ and $y' = y \bmod r$, we have by Lemma 2 that $s(g, \phi, n + x) = s(g, \phi, n + x')$ and $s(g, \phi, n + y) = s(g, \phi, n + y')$. By assumption $x' = y'$, and $0 \leq x', y' < r$. The claim follows since we know values in the range $\{s(g, \phi, n), \dots, s(g, \phi, n + r - 1)\}$ are distinct by Theorem 1.

On the other hand, suppose $s(g, \phi, n + y) = s(g, \phi, n + x)$ but $x \not\equiv y \pmod r$. Without loss of generality we can write $y = x' + u + qr$ for some $q \in \mathbb{N}, 0 < u < r$ and $x' = x \bmod r$. By Lemma 2, since $s(g, \phi, n + y) = s(g, \phi, n + x)$ we must have

$$s(g, \phi, n + x') = s(g, \phi, n + x' + u)$$

where $s(g, \phi, n + x) = s(g, \phi, n + x')$ also by Lemma 2. There are now three cases to consider; we claim each of them gives a contradiction.

First, suppose $x' + u = r$, then $s(g, \phi, n + x') = s(g, \phi, n)$. Since $x' < r$ we contradict minimality of r . The case $x' + u < r$ gives a similar contradiction.

Finally, if $x' + u > r$, without loss of generality we can write $x' + u = r + v$ for some positive integer v , so we have $s(g, \phi, n + x') = s(g, \phi, n + v)$. Since $x' \neq v$ (else we contradict $u < r$), and both values are strictly less than r , we have a contradiction, since distinct integers of this form give distinct evaluations of s . □

2.1 A Group Action

It should be clear by now that we are interested in the argument of s in terms of residue classes modulo r . Recall that the group of residue classes modulo r is denoted \mathbb{Z}_r , and its elements are written as $[i]_r$. We conclude the section by constructing the action of \mathbb{Z}_r on the cycle $\{s(g, \phi, n), \dots, s(g, \phi, n + r - 1)\}$, where we assume that the operator `mod r` returns the unique integer in $\{0, \dots, r - 1\}$ associated to its argument.

Theorem 3. *Fix $(g, \phi) \in G \times \text{End}(G)$ and let n, r be the index and period corresponding to g, ϕ . Moreover, let \mathcal{C} be the corresponding cycle of size r . The abelian group \mathbb{Z}_r acts freely and transitively on \mathcal{C} .*

Proof. First note that Theorem 2 immediately gives that $j * s(g, \phi, i + n) = s(g, \phi, (i + j) \bmod r + n)$ for any $j \in \mathbb{N}$. Our current definition of s is not defined for negative integer arguments; nevertheless, we can extend the range of the operator $*$ as follows. Let $*$: $\mathbb{Z} \times \mathcal{C} \rightarrow \mathcal{C}$ be defined by

$$j * s(g, \phi, i) = \phi^{j \bmod r}(s(g, \phi, i + n)) \cdot s(g, \phi, j \bmod r)$$

Since $j \bmod r \geq 0$, as usual we have $j * s(g, \phi, i + n) = s(g, \phi, i + j \bmod r + n)$; but since $s(g, \phi, i + n) \in \mathcal{C}$, we know $0 \leq i < r$, so $i \bmod r = i$. It follows that $j * s(g, \phi, i + n) = s(g, \phi, (i + j) \bmod r + n)$.

In fact, fix some $i \in \mathbb{N}$, and let $[j]_r$ be a fixed element of \mathbb{Z}_r . By definition, every $k \in [j]_r$ is such that $k \bmod r = j'$ for some $j' \in \{0, \dots, r - 1\}$; without loss of generality, $j' = j$. We may therefore define $\otimes : \mathbb{Z} \times \mathcal{C} \rightarrow \mathcal{C}$ by

$$[j]_r \otimes s(g, \phi, i + n) = s(g, \phi, (i + j) \bmod r + n)$$

where j is the unique element of $[j]_r$ such that $k \bmod r = j$ for each $k \in [j]_r$.

We claim that $(\mathbb{Z}_r, \mathcal{C}, \otimes)$ is a free, transitive group action.

First, let us verify that a group action is indeed defined. Certainly $[0]_r$ fixes every element in \mathcal{C} , since $s(g, \phi, (i + 0) \bmod r + n) = s(g, \phi, i + n)$ for each $i \in \{0, \dots, r - 1\}$. Moreover, one has

$$\begin{aligned} [k]_r \otimes ([j]_r \otimes s(g, \phi, i + n)) &= [k]_r \otimes s(g, \phi, (i + j) \bmod r + n) \\ &= s(g, \phi, ((i + j) \bmod r) + k \bmod r + n) \\ &= s(g, \phi, (i + (j + k)) \bmod r + n) \\ &= [j + k]_r \otimes s(g, \phi, i + n) \\ &= ([k]_r + [j]_r) \otimes s(g, \phi, i + n) \end{aligned}$$

It remains to check that the action is free and transitive. If $[j]_r \in \mathbb{Z}_r$ is such that $[j]_r$ fixes an arbitrary element of \mathcal{C} , say $s(g, \phi, i + n)$, then we have $s(g, \phi, (i + j) \bmod r + n) = s(g, \phi, i + n)$. By Theorem 2, we must have $i + j \equiv i \bmod r$, so $[j]_r = [0]_r$ and the action is free. Moreover, for arbitrary $s(g, \phi, i + n), s(g, \phi, j + n) \in \mathcal{C}$, $[k]_r = [j - i]_r \in \mathbb{Z}_r$ is such that $[k]_r \otimes s(g, \phi, i + n) = s(g, \phi, j + n)$, so the action is also transitive and we are done. \square

We summarise the above by noting that for each $(g, \phi) \in G \times \text{End}(G)$ we have shown the existence of a free, transitive, commutative group action $(\mathbb{Z}_r, \mathcal{C}, \otimes)$, where r and \mathcal{C} depend on the choice of pair (g, ϕ) .

3 Group Action Discrete Logarithms

Now that we have established the group action, we recall the Group Action Discrete Logarithm Problem (GADLP) from the introduction. Roughly speaking,

for a free transitive group action (G, X, \star) , and x, y sampled uniformly at random from the set X , we are tasked with recovering the unique G -element g such that $g \star x = y$. In this section we will show that one can construct a quantum reduction from **SDLP** to **GADLP**.

More precisely, we target the type of structure discussed in Section 1.5; that, is a set of finite semigroups $\{G_p\}_p$ indexed by some parameter p , such that the size of each G_p is polynomial in p - the so-called ‘easy’ families of semigroups. We know that multiplication in each G_p requires a number of operations bounded above by some constant independent of p , and that the complexity of these operations is bounded above by $\mathcal{O}((\log p)^2)$.

With all this in mind let $\{G_p\}_p$ be such a family of semigroups. In the previous section we have shown that for a fixed p , to each pair $(g, \phi) \in G_p \times \text{End}(G_p)$ is associated a pair (n, r) and a set \mathcal{C} . In this section we seek to show there is an efficient quantum algorithm to solve **SDLP** with respect to an arbitrary choice of (g, ϕ) , provided one has access to a **GADLP** oracle for the group action $(\mathbb{Z}_r, \mathcal{C}, \otimes)$. Before giving this reduction there remains a significant obstacle to overcome: for an arbitrary pair (g, ϕ) we have only proved the existence of the corresponding values n, r , but we do not have a means of calculating them. In order to provide a reduction to a **GADLP** oracle, however, we need to specify the appropriate group action. We therefore require access to the values n, r - in the next section, we will provide a quantum method of recovering these integers. We note that assuming access to a quantum computer is, for our purposes, justified since the best-known algorithms for **GADLP** are quantum anyway.

3.1 Calculating the Index and Period

In order to reason on the complexity of our algorithm we will use the following worst-case indicator, defined as follows:

Definition 11. *Let $\{G_p\}_{p \in P}$ be an easy family of finite semigroups parameterised by some set P . Define the following function on P :*

$$N(p) = \max_{(g, \phi) \in G_p \times \text{End}(G_p)} |\mathcal{T}_{g, \phi} + \mathcal{C}_{g, \phi}|$$

The function $N(p)$ gives a bound on the size of $\mathcal{X}_{g, \phi}$ for any $(g, \phi) \in G_p \times \text{End}(G_p)$. Since a crude such bound is the size of an easy semigroup G_p , which is assumed polynomial in p , we have that $N(p)$ is at worst polynomial in p .

Our method of calculating the index and period borrows heavily from ideas in [12, Theorem 1], which is itself a slightly repurposed version of [13, Algorithm 5]. Indeed, after a certain point we will be able to quote methods of these algorithms verbatim - nevertheless, to cater to our specific context it remains incumbent upon us to justify the following.

Lemma 3. *Let $\{G_p\}_p$ be an easy family of semigroups, and for an arbitrary p fix a pair $(g, \phi) \in G_p \times \text{End}(G_p)$. For any $l \in \mathbb{N}$, one can construct the superposition*

$$\frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} |k\rangle |s(g, \phi, k)\rangle$$

in time $\mathcal{O}(\log M(\log p)^2)$, where $M = 2^\ell$.

Proof. When (g, ϕ) is fixed, notice that we can think of $s(g, \phi, i) : G \times \text{End}(G) \times \mathbb{N} \rightarrow G$ as a function $s_{g, \phi}(i) : \mathbb{N} \rightarrow G$. Since $N(p)$ is a bound on the size of $\mathcal{X}_{g, \phi}$, taking m to be smallest integer such that $2^m \geq N(p)$ (note that $m = \mathcal{O}(\log(N(p)))$), the set $\mathcal{X}_{g, \phi}$ has binary representation in the set $\{0, 1\}^m$. By definition the integers $\{0, \dots, M-1\}$ have binary representation in $\{0, 1\}^\ell$, so we can think of the restriction of $s_{g, \phi}$ on $\{0, \dots, M-1\}$ as a function from $\{0, 1\}^\ell$ into $\{0, 1\}^m$. There is therefore a Boolean circuit computing $s_{g, \phi}$; it is standard (say, by [17, Theorem 2.3.2]) that one can construct a circuit implementing $s_{g, \phi}$ using reversible gates. Call this circuit $Q_{s_{g, \phi}}$; since the reversible circuit does no worse than the classical circuit, we can assume a single application of $Q_{s_{g, \phi}}$ takes at worst the time complexity of calculating $s_{g, \phi}(M)$.

What is the time complexity of calculating $s_{g, \phi}(M)$? We know by definition that $(s_{g, \phi}(M), \phi^M) = (g, \phi)^M$, where the exponentiation refers to holomorph exponentiation. Recall that since we have assumed we are working in an easy family of semigroups, each multiplication in the holomorph involves one application of ϕ , followed by some fixed constant number of matrix multiplications independently of p . Calculating the holomorph exponentiation in the standard square-and-multiply fashion, therefore, we expect to perform $\mathcal{O}(\log M)$ holomorph multiplications. We know that evaluating ϕ takes time $\mathcal{O}((\log p)^2)$; since a fixed number of matrix multiplications follow, the total time for calculating $s(g, \phi, M)$ is $\mathcal{O}(\log M(\log p)^2)$.

If we can show a single application of $Q_{s_{g, \phi}}$ gives the desired superposition we are done. It is standard, however, that the uniform superposition of an M -bit quantum register, together with an ancillary m -bit register in the state $|0\rangle$, can be inputted into $Q_{s_{g, \phi}}$ to produce the desired superposition. Since preparing the appropriate uniform superposition can be done by applying a Hadamard gate in time $\mathcal{O}(\log M)$, we are done. \square

Armed with the ability to efficiently calculate the appropriate superposition, we will quickly find ourselves with exactly the kind of state arrived at in [13, Algorithm 5], thereby allowing us to recover the period r in Algorithm 1. A small adaptation of standard binary search techniques completes the task by using knowledge of r to recover the index n .

Theorem 4. *Let $\{G_p\}_p$ be an easy family of semigroups, and fix p . For any pair $(g, \phi) \in G_p \times \text{End}(G_p)$:*

1. *For sufficiently large $M \in \mathbb{N}$, $\text{PeriodRecovery}((g, \phi), M)$ recovers the period r of (g, ϕ) in time $\mathcal{O}((\log p)^3)$, and with constant probability.*
2. *BinarySearch $((g, \phi), 1, M, r)$ returns the index n of g, ϕ in time $\mathcal{O}((\log p)^4)$.*

Proof. 1. Fix a pair $(g, \phi) \in G_p \times \text{End}(G_p)$ and let r be its period. Let $\ell \in \mathbb{N}$ be the smallest positive integer such that $2^\ell \geq (N(p)^2 + N(p))$, and $M = 2^\ell$. In steps 1-3 of Algorithm 1, we prepare the required superposition as described in Lemma 3.

Algorithm 1 PeriodRecovery($((g, \phi), M)$)

Input: Pair $(g, \phi) \in G_p \times \text{End}(G_p)$, upper bound on size of superposition to create M

Output: Period r of (g, ϕ) or 'Fail'

- 1: $R_0 \leftarrow |0\rangle|0\rangle$
 - 2: $R_1 \leftarrow$ Hadamard transform applied to first register
 - 3: $R_2 \leftarrow$ appropriate quantum circuit applied to R_1
 - 4: Measure second register leaving collapsed first register R_3
 - 5: $R_4 \leftarrow$ QFT over \mathbb{Z}_M applied to R_3
 - 6: $R_5 \leftarrow$ measure R_4
 - 7: $r \leftarrow$ continued fraction expansion of R_5/M
 - 8: **if** $r * s(g, \phi, M) \neq s(g, \phi, M)$ **then**
 - 9: **return** 'Fail'
 - 10: **else**
 - 11: **return** r
 - 12: **end if**
-

Algorithm 2 BinarySearch($((g, \phi), start, end, r)$)

Input: Pair (g, ϕ) , integers $start, end$ where $start \leq end$, period r of g, ϕ

Output: Index n of (g, ϕ)

- 1: **if** $start = end$ **then:**
 - 2: **return** $start$
 - 3: **end if**
 - 4: $left \leftarrow start$
 - 5: $right \leftarrow end$
 - 6: $mid \leftarrow \lfloor (left + right)/2 \rfloor$
 - 7: **if** $r * s(g, \phi, mid) \neq s(g, \phi, mid)$ **then**
 - 8: **return** BinarySearch($((g, \phi), mid + 1, right, r)$)
 - 9: **else**
 - 10: **return** BinarySearch($((g, \phi), left, mid, r)$)
 - 11: **end if**
-

In Step 4, we measure the second register. With probability n/M doing so will cause us to observe an element of the tail; that is, some $s(g, \phi, i)$ such that $i < n$. In this case, by the laws of partial observation, the first register is left in a superposition of integers corresponding to this value - but by definition there is only one of these, so the first register consists of a single computational basis state and the algorithm has failed. On the other hand, with probability $(M - n)/M$ measuring the second register gives an element of \mathcal{C} . Now, since $M \geq N(p)^2 + N(p)$, we observe an element of \mathcal{C} with probability

$$\frac{M - n}{M} = 1 - \frac{n}{M} \geq 1 - \frac{n}{N(p)^2 + N(p)}$$

Since by definition one has $n \leq N(p)$, it follows that the relevant probability is better than $N(p)/(N(p) + 1) \geq 1/2$. In other words, we observe an element

of the desired form with constant, positive probability. Provided such an element was observed, after measuring the second register, the superposition of corresponding integers in the first register is the following:

$$\frac{1}{\sqrt{s_r}} \sum_{j=0}^{s_r-1} |x_0 + jr\rangle$$

To see this, note that the function s is periodic of period r , and by Theorem 1 each $s(g, \phi, i)$ such that $i \geq n$ can only assume one of the distinct values $s(g, \phi, n), \dots, s(g, \phi, n + r - 1)$. In particular, the integers in $\{1, \dots, M\}$ that give a specific value of the cycle under s are of the form $x_0 + jr$ for some $x_0 \in \{n, \dots, n + r - 1\}$. The largest such integer, by definition, is $x_0 + s_r r$, where s_r is just the largest integer such that $x_0 + s_r r < M$. Note that the superposition is normalised by this factor so that the sum of the squares of the amplitudes is 1.

We now have exactly the same kind of state found in [13, Algorithm 5]⁷, so we may proceed exactly according to the remaining steps in this algorithm. In Step 5 we apply a Quantum Fourier Transform (QFT) over \mathbb{Z}_M to the state, which can be done in time $\mathcal{O}((\log M)^2)$. In step 6 we measure the state R_4 ; it is shown in [13, Algorithm 5, Step 5] that with probability at least $4/\pi^2$, measuring the resulting state leaves one with the closest integer to one of the at most r multiples of M/r (note that M/r is not necessarily an integer) with probability better than $4/\pi^2$. Writing this closest integer as $\lfloor jM/r \rfloor$ for some $j \in \mathbb{N}$, one checks that the fraction j/r is a distance of at most $1/2M$ from $(\lfloor jM/r \rfloor)/M$; by [17, Theorem 8.4.3], j/r will appear as one of the convergents in the continued fraction expansion of $(\lfloor jM/r \rfloor)/M$ provided $1/2M \leq 1/2r^2$. Certainly this holds, since $r < N(p) < M$. Provided we have observed an integer of the appropriate form, then, it remains to carry out a continued fraction expansion on $(\lfloor jM/r \rfloor)/M$, which we can do with repeated application of the Euclidean algorithm.

Let us summarise the complexity of the algorithm. The dominating factors are the creating of the relevant superposition in time

$$\mathcal{O}((\log M)(\log p)^2) = \mathcal{O}((\log N(p)(\log p))^2) = \mathcal{O}((\log p)^3)$$

where the last equality follows from the easy property of the relevant semi-group family; that is, one has that $N(p)$ is at worst polynomial in p . Similarly, the application of QFT can be done in time $\mathcal{O}((\log p)^2)$, so we have the complexity estimate claimed at the outset. Note also that the algorithm succeeds provided an element of the cycle is observed after the first measurement, and that the second measurement gives an appropriate integer. Since both of these events occur with probability bounded below by a constant, the algorithm succeeds with probability $\Omega(1)$.

2. We prove correctness of the algorithm by proving that any values $start, end$ such that $start \leq n \leq end$ will return n , which we accomplish by strong

⁷ This type of state also occurs in Shor's factoring algorithm.

induction on $k = start - end + 1$. To save on cumbersome notation we assume (g, ϕ) and r are fixed, and write

$$BinarySearch((g, \phi,), start, end, r) = BS(start, end)$$

First, suppose $k = 1$ and $start \leq n \leq end$. Either $n = start$ or $n = start + 1$, and we know that $mid = start$ after the floor function is applied. In the first case, $r * s(g, \phi, mid) = s(g, \phi, mid)$, so $BS(start, mid)$ is returned; but since $start = mid$, $start = n$ is returned. Otherwise, one has $r * s(g, \phi, mid) \neq s(g, \phi, mid)$ and $BS(mid + 1, end)$ is returned, and we are done since $mid + 1 = end = n$. Now for some $k > 1$ suppose all positive integers $start', end'$ such that $start' \leq n \leq end'$ and $end' - start' + 1 < k$ have $BS(start', end') = n$. We should like to show that an arbitrary choice of $start, end$ with $start \leq n \leq end$ and $end - start + 1 = k$ enjoys this same property. To see that it does we can again consider the two cases.

The algorithm first calculates $mid = \lfloor (end - start) / 2 \rfloor$. Suppose $r * s(g, \phi, mid) = s(g, \phi, mid)$, then $BS(start, mid)$ is run. Since n is the smallest integer such that $r * s(g, \phi, n) = s(g, \phi, n)$ and $n \geq start$ by assumption, we know $start \leq n \leq mid$. Moreover, $mid - start + 1 < end - start + 1 < k$. By inductive hypothesis $BS(start, mid)$ returns n .

The other case is similar; this time, if $r * s(g, \phi, mid) \neq s(g, \phi, mid)$ we know $n \geq mid + 1$ by definition of n . We also know that $end - (mid + 1) + 1 = end - mid < end - mid + 1 = k$, so the algorithm returns $BS(mid + 1, end) = n$ by inductive hypothesis.

Notice that each time *BinarySearch* is called the calculation of $r * s(g, \phi, mid)$ is required. We know already that $s(g, \phi, mid)$ can be calculated in time $\mathcal{O}(\log mid (\log p)^2) = \mathcal{O}((\log p)^3)$. Given ϕ^r , $s(g, \phi, r_{g, \phi})$ and $s(g, \phi, mid)$, the calculation of $r * s(g, \phi, mid)$ requires evaluating an endomorphism and a semigroup multiplication - we have argued already that this can be done in time $\mathcal{O}((\log p)^2)$. Recall that in the proof of Lemma 3, we showed that one can calculate the holomorph exponent $(g, \phi)^r = (s(g, \phi, r), \phi^r)$ in time $\mathcal{O}(\log r (\log p)^2)$, so the total calculation is done in time $\mathcal{O}((\log p)^3)$ since $r < M$. Clearly, *BinarySearch* will be called $\mathcal{O}(\log M) = \mathcal{O}(\log p)$ times, since the size of the interval to search halves at each iteration, and we conclude that *BinarySearch* recovers the index in time $\mathcal{O}((\log p)^4)$.

□

3.2 From SDLP to GADLP

Let us assemble the components developed so far in this section into a reduction of SDLP to GADLP.

Theorem 5. *Let $\{G_p\}_p$ be an easy family of semigroups, and fix p . Algorithm 3 solves SDLP with respect to a pair $(g, \phi) \in G_p \times \text{End}(G_p)$ given access to a GADLP oracle for the group action $(\mathbb{Z}_r, \text{mathcal{C}}, \otimes)$. The algorithm runs in time $\mathcal{O}((\log p)^4)$, makes at most a single query to the GADLP oracle, and succeeds with probability $\Omega(1)$.*

Algorithm 3 Solving SDLP with GADLP oracle

Input: $(g, \phi), s(g, \phi, x)$ **Output:** x

```

1:  $r \leftarrow \text{PeriodRecovery}((g, \phi), M)$  for sufficiently large  $M$ 
2: if  $r = \text{'Fail'}$  then
3:   return 'Fail'
4: end if
5:  $n \leftarrow \text{BinarySearch}((g, \phi), 1, M, r)$ 
6: if  $r * s(g, \phi, x) = s(g, \phi, x)$  then
7:    $d \leftarrow s(g, \phi, n)$ 
8:    $x' \leftarrow \text{GADLP oracle applied to } d, s(g, \phi, x)$ 
9:    $x \leftarrow n + x'$ 
10: else
11:    $t \leftarrow \text{BinarySearch2}(s(g, \phi, x), 1, n, r)$ 
12:    $x \leftarrow n - t$ 
13: end if
14: return  $x$ 

```

Proof. Consider an instance of SDLP whereby we are given the pair (g, ϕ) and the value $s(g, \phi, x)$, for some x sampled uniformly at random from the set $\{1, \dots, n + r - 1\}$. We show that Algorithm 2 recovers x .

We start by applying Algorithms 1 and 2 to the pair (g, ϕ) , recovering the pair n, r with constant probability. By Theorem 4, we can do so in time $\mathcal{O}((\log p)^4)$. Now, $s(g, \phi, x)$ might be in tail or in the cycle - but with our knowledge of r we can check in Step 6 which is true by verifying whether $r * s(g, \phi, x) = s(g, \phi, x)$. As discussed in the proof of Theorem 4, we can perform this check in time $\mathcal{O}((\log p)^3)$.

There are now two cases to consider. First, suppose that the check in Step 6 is passed, then $s(g, \phi, x)$ is in the cycle, and we may proceed as follows. Compute $s(g, \phi, n)$ in time $\mathcal{O}((\log p)^3)$, and query the GADLP oracle on input $s(g, \phi, n), s(g, \phi, x)$ (Step 8) to recover the \mathbb{Z}_r element $[y]_r$. Without loss of generality the smallest positive representative of this class, say x' , is such that $n + x' = x$, so we recover x in Step 9.

Now suppose that $s(g, \phi, x)$ is in the tail. We run the algorithm BinarySearch2 to recover t , the smallest integer such that $t * s(g, \phi, x)$ is invariant under r . BinarySearch2 is precisely the same as Algorithm 2, except that in the verification step, we check if $r * (mid * s(g, \phi, x)) = mid * s(g, \phi, x)$. It is not hard to adapt the proof of correctness to show that BinarySearch2 does indeed return t in time $\mathcal{O}((\log p)^4)$. Moreover, by minimality of n and the additivity of $*$, we must have $x + t = n$, from which we recover $x = n - t$.

Finally, we note that the only probabilistic step of this algorithm is the application of Algorithm 1, so we successfully recover x with the same success probability as Algorithm 1, and we are done. \square

In summary, we have an efficient quantum reduction from SDLP to GADLP: an efficient quantum procedure extracts the period r , and from there a classical

procedure gives the index n . In order to recover x , it remains to either carry out an efficient classical procedure, or recover x with a single query to a GADLP oracle. Moreover, assuming the GADLP oracle always succeeds, the success probability is precisely that of Algorithm 1 - that is, bounded below by a positive constant independently of p .

Remark 3. The factor $\log p$ in the complexity estimate is really coming from the ‘length’ of a binary representation of G_p ; that is, the number of bits required to represent G_p . In our case the size of G_p happens to be polynomial in p , and therefore the relevant ‘length’ is of order $\mathcal{O}(\log p)$. One might be used to seeing the complexity of similar period-finding routines, such as Shor’s algorithm, presented as cubic in the length of a binary representation of the relevant parameters - see for example [17, Section 3.3.3]. In our case, the total complexity is quartic in the length of a binary representation, essentially because after the quantum part of the algorithm we still need to compute $\mathcal{O}(\log p)$ evaluations of the function s in order to compute the index. In a sense, then, we can think of this extra $\log p$ factor as the extra cost incurred from the slightly more complicated scenario inherent to the problem.

3.3 Towards equivalence of SCDH and SDLP

Let us briefly recall the classical setting for the Discrete Logarithm Problem. It is well-known (see, for example, [22]) that the relationship between the Discrete Logarithm Problem and the Computational Diffie–Hellman Problem is not well understood. In particular, one can certainly solve the Computational Diffie–Hellman Problem provided one can solve the Discrete Logarithm Problem, but the converse is not known.

The situation in the group action setting is rather clearer. In recent work [26], Montgomery and Zhandry demonstrate full quantum equivalence of GADLP and GACDH. This remarkable result clarifies the landscape of complexity problems as summarised by the following diagram, where a directed arrow between two nodes indicates the problem in the former node can be solved by an oracle for the problem in the latter. Moreover, a solid arrow indicates a reduction demonstrated in this paper, whereas a dashed arrow denotes a reduction that was already known.

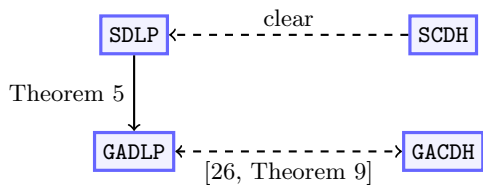


Fig. 1. Landscape of known reductions.

The relationship between GACDH and SCDH as we have defined them remains unclear and is left open to further study.

4 Quantum Algorithms for GADLP

Now that we have shown SDLP can be efficiently solved with access to an appropriate GADLP oracle it remains to examine the state of the art for GADLP. It is here that the Abelian Hidden Shift Problem (Definition 8) comes in to play. Roughly speaking, we are given two injective functions f, g from a group A to a set S that differ by a constant ‘shift’ value, and our task is to recover the shift value.

It is reasonably well-known (see [31, 11]) that GADLP reduces to AHSP. In this section, we provide a context-specific proof of this fact, before discussing the best known algorithms for AHSP.

4.1 Group Actions to Hidden Shift

The following result is found more or less verbatim in, for example, [11]. We here give a context-specific reduction, for completeness.

Theorem 6. *Let $\{G_p\}_p$ be an easy family of semigroups and fix p . For some pair $(g, \phi) \in G_p \times \text{End}(G_p)$ let $(\mathbb{Z}_r, \mathcal{C}, \otimes)$ be the associated group action defined in Theorem 3. One can efficiently solve GADLP in $(\mathbb{Z}_r, \mathcal{C}, \otimes)$ given access to an AHSP oracle with respect to $\mathbb{Z}_r, \mathcal{C}$.*

Proof. Suppose we are given an instance of GADLP in $(\mathbb{Z}_r, \mathcal{C}, \otimes)$; that is, we are given a pair $(s(g, \phi, n + i), s(g, \phi, n + j)) \in \mathcal{C}$ for some $i, j \in \{1, \dots, r\}$ and tasked with finding the unique $[k]_r \in \mathbb{Z}_r$ such that $[k]_r \otimes s(g, \phi, n + i) = s(g, \phi, n + j)$. Our strategy is to construct injective functions $f_A, f_B : \mathbb{Z}_r \rightarrow \mathcal{C}$ that hide $[k]_r$, and use the AHSP oracle to recover this value.

Set $f_A, f_B : \mathbb{Z}_r \rightarrow \mathcal{C}$ as $f_A([x]_r) = [x]_r * s(g, \phi, n + i)$ and $f_B([x]_r) = [x]_r * s(g, \phi, n + j)$. Then

$$\begin{aligned} f_B([x]_r) &= [x]_r * s(g, \phi, n + j) \\ &= [x]_r * ([k]_r * s(g, \phi, n + i)) \\ &= ([x]_r + [k]_r) * s(g, \phi, n + i) \\ &= f_A([x]_r + [k]_r) \end{aligned}$$

In other words, f_A, f_B hide $[k]_r$. To complete the setup of an instance of AHSP we require the functions to be injective, which follows from the action being free and transitive. \square

Note that we have in this case left out complexity estimates. This is because in order to give a full description of the functions f_A, f_B we need to compute the group \mathbb{Z}_r , which can be done efficiently with knowledge of r . However, since we have already described a method of recovering r , we will discuss the complexity in the full SDLP algorithm at the end of this section.

4.2 Hidden Shift Algorithms

We have finally arrived at the problem for which there are known quantum algorithms. The fastest known is of subexponential complexity, and is presented in [23, Proposition 6.1] as a special case of the Dihedral Hidden Subgroup Problem.

Theorem 7 (Kuperberg’s Algorithm). *There is a quantum algorithm that solves AHSP with respect to $\mathbb{Z}_r, \mathcal{C}$ with time and query complexity $2^{\mathcal{O}(\sqrt{\log r})}$.*

Kuperberg’s algorithm also requires quantum space $2^{\mathcal{O}(\log r)}$. For a slower but less space-expensive algorithm, we can also use a generalised version of an algorithm due to Regev [30]. The generalised version appears in [11, Theorem 5.2].

Theorem 8 (Regev’s Algorithm). *There is a quantum algorithm that solves AHSP with respect to $\mathbb{Z}_r, \mathcal{C}$ with time and query complexity*

$$e^{\sqrt{2+o(1)}\sqrt{\ln r \ln \ln r}}$$

and space complexity $\mathcal{O}(\text{poly}(\log r))$.

We note that both Kuperberg’s and Regev’s algorithms succeed with constant probability.

4.3 Solving SDLP

We finish the section by stitching all the components together into an algorithm that solves SDLP. For brevity of exposition we include only complexity estimates for using Kuperberg’s algorithm - but finding the bounds in the case of Regev’s algorithm is very similar.

Theorem 9. *Let $\{G_p\}_p$ be an easy family of semigroups, and fix p . For any pair $(g, \phi) \in G_p \times \text{End}(G_p)$, there is a quantum algorithm solving SDLP with respect to (g, ϕ) with time and query complexity $2^{\mathcal{O}(\sqrt{\log p})}$.*

Proof. Let $(g, \phi) \in G_p \times \text{End}(G_p)$ and suppose we are given the value $s(g, \phi, x)$ for some x sampled uniformly from the set $\{1, \dots, N\}$, where N is the size of $\mathcal{X}_{g, \phi}$. The following steps recover x :

1. Run Algorithms 1 and 2 on the pair (g, ϕ) . By Theorem 4, with positive probability we recover the index and period of (g, ϕ) , the pair (n, r) , in time $\mathcal{O}((\log p)^4)$.
2. By Theorem 5, either we are done efficiently, or it remains to solve an instance of GADLP with respect to the group action $(\mathbb{Z}_r, \mathcal{C}, \otimes)$.
3. By Theorem 6, once we have computed the group action $(\mathbb{Z}_r, \mathcal{C}, \otimes)$ it remains to solve an instance of AHSP with respect to $\mathbb{Z}_r, \mathcal{C}$. This can be done with access to the index and period n, r .
4. Solve AHSP using Kuperberg’s algorithm or Regev’s algorithm.

In summary, the total quantum complexity of solving an SDLP instance for any pair in $G_p \times \text{End}(G_p)$ is either $\mathcal{O}((\log p)^4)$, or if a call to the GADLP oracle is required, $2^{\mathcal{O}(\sqrt{\log r})} = 2^{\mathcal{O}(\sqrt{\log p})}$ since G_p is from an easy family of semigroups. Depending on constants, we expect this latter term to dominate the complexity. Moreover, we note that since both our algorithm to extract the period and Kuperberg’s algorithm succeed with constant probability, we expect our algorithm to succeed with constant probability also. \square

5 Conclusion

We have provided the first dedicated analysis of SDLP, showing a reduction to a well-studied problem. Perhaps the most surprising aspect of the work is the progress made by a simple rephrasing; we made quite significant progress through rather elementary methods, and we suspect much more can be made within this framework.

The reader may notice that we have shown that SPDKE is an example of a commutative action-based key exchange, and that breaking all such protocols can be reduced to the Abelian Hidden Shift Problem. Indeed, this work shows the algebraic machinery of SPDKE is a candidate for what Couveignes calls a *hard homogenous space*⁸ [14], which was not known until now. In line with the naming conventions in this area we propose a renaming of SPDKE to SPDH, which stands for ‘Semidirect Product Diffie–Hellman’, and should be pronounced *spud*.

We would also like to stress the following sentiment. The purpose of this paper is not to claim a general purpose break of SPDKE (or, indeed, SPDH) - the algorithm presented is subexponential in complexity, which has been treated as tolerable in classical contexts. Instead, the point is to show a connection between SDLP and a known hardness problem, thereby providing insight on a problem about which little was known.

References

- [1] John Baena, Pierre Briaud, Daniel Cabarcas, Ray A. Perlner, Daniel Smith-Tone, and Javier A. Verbel. “Improving Support-Minors rank attacks: applications to GeMSS and Rainbow”. In: *IACR Cryptol. ePrint Arch.* (2021), p. 1677. URL: <https://eprint.iacr.org/2021/1677>.
- [2] Christopher Battarbee, Delaram Kahrobaei, and Siamak F Shahandashti. “Semidirect Product Key Exchange: the State of Play”. In: *arXiv preprint arXiv:2202.05178* (2022).
- [3] Christopher Battarbee, Delaram Kahrobaei, and Siamak F. Shahandashti. “Cryptanalysis of Semidirect Product Key Exchange Using Matrices Over Non-Commutative Rings”. In: *Mathematical Cryptology 1.2* (2022), pp. 2–9.

⁸ Another major example of which arises from the theory of isogenies between elliptic curves - see, for example, [9]

- [4] Christopher Battarbee, Delaram Kahrobaei, Dylan Taylor, and Siamak F Shahandashti. “On the efficiency of a general attack against the MOBS cryptosystem”. In: *arXiv:2111.05806; to appear in Journal of Mathematical Cryptology* (2022).
- [5] Ward Beullens. “Breaking Rainbow Takes a Weekend on a Laptop”. In: *IACR Cryptol. ePrint Arch.* (2022), p. 214. URL: <https://eprint.iacr.org/2022/214>.
- [6] Daniel Brown, Neal Koblitz, and Jason Legrow. “Cryptanalysis of ‘MAKE’”. In: *Journal of Mathematical Cryptology* 16.1 (2022), pp. 98–102.
- [7] Antoine Casanova, Jean-Charles Faugere, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. “G_eMSS : a great multivariate short signature”. PhD thesis. UPMC-Paris 6 Sorbonne Universités; INRIA Paris Research Centre, PolSys Team, 2017.
- [8] Wouter Castryck and Thomas Decru. *An efficient key recovery attack on SIDH (preliminary version)*. Cryptology ePrint Archive, Paper 2022/975. 2022. URL: <https://eprint.iacr.org/2022/975>.
- [9] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. “CSIDH: an efficient post-quantum commutative group action”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2018, pp. 395–427.
- [10] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on Post-Quantum Cryptography*. Research report NISTIR 8105. NIST, 2016. URL: http://csrc.nist.gov/publications/drafts/nistir-8105/nistir_8105_draft.pdf.
- [11] Andrew Childs, David Jao, and Vladimir Soukharev. “Constructing elliptic curve isogenies in quantum subexponential time”. In: *Journal of Mathematical Cryptology* 8.1 (2014), pp. 1–29.
- [12] Andrew M Childs and Gábor Ivanyos. “Quantum computation of discrete logarithms in semigroups”. In: *Journal of Mathematical Cryptology* 8.4 (2014), pp. 405–416.
- [13] Andrew M Childs and Wim Van Dam. “Quantum algorithms for algebraic problems”. In: *Reviews of Modern Physics* 82.1 (2010), p. 1.
- [14] Jean-Marc Couveignes. “Hard homogeneous spaces”. In: *Cryptology ePrint Archive* (2006). URL: <https://eprint.iacr.org/2006/291.pdf>.
- [15] Dima Grigoriev and Vladimir Shpilrain. “Tropical cryptography II: extensions by homomorphisms”. In: *Communications in Algebra* 47.10 (2019), pp. 4224–4229.
- [16] Maggie Habeeb, Delaram Kahrobaei, Charalambos Koupparis, and Vladimir Shpilrain. “Public key exchange using semidirect product of (semi) groups”. In: *International Conference on Applied Cryptography and Network Security*. Springer. 2013, pp. 475–486.
- [17] Mika Hirvensalo. *Quantum computing*. Springer Science & Business Media, 2003.
- [18] John Mackintosh Howie. *Fundamentals of semigroup theory*. 12. Oxford University Press, 1995.

- [19] Steve Isaac and Delaram Kahrobaei. “A closer look at the tropical cryptography”. In: *International Journal of Computer Mathematics: Computer Systems Theory* (2021), pp. 1–6.
- [20] Delaram Kahrobaei, Ramon Flores, and Marialaura Noce. “Group-based Cryptography in the Quantum Era”. In: *The Notices of American Mathematical Society, to appear* (2022). URL: <https://arxiv.org/abs/2202.05917>.
- [21] Delaram Kahrobaei and Vladimir Shpilrain. “Using semidirect product of (semi) groups in public key cryptography”. In: *Conference on Computability in Europe*. Springer, 2016, pp. 132–141.
- [22] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.
- [23] Greg Kuperberg. “A subexponential-time quantum algorithm for the dihedral hidden subgroup problem”. In: *SIAM Journal on Computing* 35.1 (2005), pp. 170–188.
- [24] Luciano Maino and Chloe Martindale. *An attack on SIDH with arbitrary starting curve*. Cryptology ePrint Archive, Paper 2022/1026. 2022. URL: <https://eprint.iacr.org/2022/1026>.
- [25] Chris Monico. “Remarks on MOBS and cryptosystems using semidirect products”. In: *arXiv preprint arXiv:2109.11426* (2021).
- [26] Hart Montgomery and Mark Zhandry. “Full Quantum Equivalence of Group Action DLog and CDH, and More”. In: *Cryptology ePrint Archive* (2022).
- [27] Alexei Myasnikov and Vitalii Roman’kov. “A linear decomposition attack”. In: *Groups Complexity Cryptology* 7.1 (2015), pp. 81–94.
- [28] Nael Rahman and Vladimir Shpilrain. “MAKE: A matrix action key exchange”. In: *Journal of Mathematical Cryptology* 16.1 (2022), pp. 64–72.
- [29] Nael Rahman and Vladimir Shpilrain. “MOBS: Matrices Over Bit Strings public key exchange”. In: <https://eprint.iacr.org/2021/560> (2021).
- [30] Oded Regev. “A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space”. In: *arXiv preprint quant-ph/0406151* (2004).
- [31] Anton Stolbunov. “Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves”. In: *Advances in Mathematics of Communications* 4.2 (2010), p. 215.
- [32] Chengdong Tao, Albrecht Petzoldt, and Jintai Ding. “Efficient Key Recovery for All HFE Signature Variants”. In: *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. Lecture Notes in Computer Science. Springer, 2021, pp. 70–93. URL: https://doi.org/10.1007/978-3-030-84242-0%5C_4.