Anton Stolbunov

Cryptographic Schemes Based on Isogenies

Thesis for the degree of Philosophiae Doctor

Trondheim, January 2012

Norwegian University of
Science and Technology
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Telematics

NTNU

Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Telematics

ISBN 978-82-471-3295-1 (printed ver.)
ISBN 978-82-471-3296-8 (electronic ver.)
ISSN 1503-8181

Doctoral Theses at NTNU, 2012:16

Printed by Tapir Uttrykk

Abstract

In this thesis we use isogenies between ordinary elliptic curves for the construction of new cryptographic schemes. The thesis is organized into an introductory chapter followed by three articles.

In the introduction we motivate our work by the necessity of exploring new computationally hard problems applicable to cryptography. We describe our work and results, and survey the related work. We also give the background material from algebraic number theory and provide explanatory examples.

In the first paper we propose a number of cryptographic schemes based on the group action on a set: a public-key encryption scheme $\mathcal{PE}$, a key agreement protocol $\mathcal{KA}1$, three authenticated key agreement protocols, and some related schemes. We construct an implementation of these schemes for the action of the ideal class group $\mathcal{CL}(\mathcal{O}_K)$ of an imaginary quadratic field K on the set $\mathcal{ELL}_{p,n}(\mathcal{O}_K)$ of isomorphism classes of elliptic curves over $\mathbb{F}_p$ with n points and the endomorphism ring $\mathcal{O}_K$. Implementation details, such as representation of set and group elements, group action, sampling from $\mathcal{CL}(\mathcal{O}_K)$, and cryptosystem parameter generation, are described as well. The paper presents speed measurements of our trial implementation.

In the second paper we provide security reductions for the protocol $\mathcal{KA}1$ and the encryption scheme $\mathcal{PE}$. For the $\mathcal{KA}1$ protocol we use the notion of session key security in the authenticated-link model proposed by Canetti and Krawczyk. For the $\mathcal{PE}$ scheme we use a version of the semantic security notion proposed by Goldwasser and Micali. We prove that the security of the $\mathcal{KA}1$ protocol and the $\mathcal{PE}$ scheme is based on the decisional Diffie-Hellman group action (DDHA) problem, which is defined in our paper. The class-group DDHA problem is reducible to the isogeny problem: given two isogenous ordinary elliptic curves, compute an isogeny between them.

The isogeny problem is studied in our third paper. A low storage algorithm for this problem was proposed by Galbraith, Hess and Smart (GHS) in 2002. We give an improvement of the GHS algorithm by modifying the pseudorandom walk so that lower-degree isogenies are used more frequently. This is motivated by the fact that high degree isogenies are slower to compute than low degree ones. We analyse the running time of the parallel collision search algorithm when the partitioning is uneven. We also give experimental results. We conclude that our isogeny problem algorithm is around 14 times faster than the GHS algorithm when constructing horizontal isogenies between random isogenous elliptic curves over a 160-bit prime field. The expected running time of our improved algorithm indicates that the computational complexity of the isogeny problem is currently exponential.

Acknowledgements

This thesis contains my research work done between November 2006 and July 2011. Most of this time I was employed by the Department of Telematics at the Norwegian University of Science and Technology (NTNU). My advisor has been Prof. Stig Frode Mjølsnes, and my co-advisor was Prof. Alexei Rudakov, until he moved from Trondheim in 2008. I am very grateful to Stig and Alexei for the lessons they have taught me and for their support. I am also thankful to Prof. Kristian Gjøsteen for his tremendous help and expertise.

From March to May 2010 I visited Prof. Steven Galbraith in Auckland, New Zealand. Numerous constructive discussions in a hospitable atmosphere evolved into a joint research work with Steven. I thank him for this opportunity.

I am also grateful to Prof. Alexander Rostovtsev for his lectures and advisory during my master's studies in Saint-Petersburg, Russia, and for the suggestion of the topic of this research.

Contents

1 Introduction **1**
- 1.1 Research Questions . . . 1
- 1.2 Our Work and Results . . . 4
- 1.3 Related Work: Cryptographic Schemes Based on Isogenies . . . 6
- 1.4 Background: the Class Group Action on the Set of j-invariants . . . 10

2 Constructing Cryptographic Schemes Based on Isogenies **21**
- 2.1 Introduction . . . 21
- 2.2 Public-Key Cryptography Based on Group Action . . . 22
- 2.3 Isogenous Elliptic Curves over Prime Fields . . . 27
- 2.4 Elements of $\mathcal{ELL}_{p,n}$ and $\mathcal{CL}$ . . . 30
- 2.5 Implementation of $\mathcal{CL}$ Action on $\mathcal{ELL}_{p,n}$ . . . 32
- 2.6 Implementation of Sampling from $\mathcal{CL}$ . . . 35
- 2.7 Security of $\mathcal{ELL}$-Based Cryptographic Schemes . . . 37
- 2.8 Numerical Experiments . . . 40
- 2.9 Concluding Remarks . . . 41
- 2.A System Parameters Used in Time Measurements . . . 43
- 2.B More Schemes Based on Group Action . . . 43

3 Security Reductions for Schemes Based on Group Action **51**
- 3.1 Introduction . . . 51
- 3.2 Notation . . . 52
- 3.3 Cryptographic Schemes . . . 53
- 3.4 Computational Problems . . . 55
- 3.5 Reductionist Security Arguments . . . 57
- 3.6 Concluding Remarks . . . 63

4 Improved Algorithm for the Isogeny Problem **67**
- 4.1 Introduction . . . 67
- 4.2 Definitions and Notation . . . 69
- 4.3 Algorithm for Solving the GAIP and the $\mathcal{CL}$-GAIP . . . 71
- 4.4 Theoretical Analysis of the Algorithm . . . 76
- 4.5 Comparing Theory and Practice . . . 80
- 4.6 The Algorithm in Practice . . . 86
- 4.7 Conclusion . . . 90
- 4.A Proof of Theorem 4.1 . . . 90
- 4.B Argument for Heuristic 4.2 . . . 99
- 4.C Numerical Experiments . . . 102

Bibliography **111**

Chapter 1

Introduction

1.1 Research Questions

Encryption was first applied thousands of years ago. This started a contest between cryptanalysis and improvement of cryptographic means. A steady advancement of attackers' intelligence and computational power led to failure of many cryptographic schemes.

Security of public-key cryptographic schemes relies on the hardness of particular computational problems, called *hard problems*. Among the most broadly used hard problems nowadays are the integer factoring problem, the discrete logarithm problem (DLP) in a multiplicative subgroup of integers modulo a prime, and the DLP on an ordinary elliptic curve. Cryptographic properties of these problems are summarized in Table 1.1. Here s denotes the desired security level, e.g. $s = 96$ is believed to provide protection until the year 2020 [41]. We see that among the three named problems only the elliptic-curve DLP can be used with short keys, because the complexity of the other two problems has been already reduced to subexponential. Needless to say, the situation will change for the worse if stronger attacks are invented in the future.

The anticipated arrival of quantum computers adds fuel to the flame, as quantum computers can solve the aforementioned and many other hard problems in polynomial time. The main challenge of the post-quantum cryptography is to identify hard problems that remain hard also for quantum computers. Among the most promising alternatives are the hash collision and preimage problems, the lattice-based problems, e.g. the shortest vector problem, and the problem of decoding certain linear codes [10]. These problems are listed in Table 1.1 as well. To solve all problems presented in the table, only a relatively small quantum register is needed (having a poly(s) qubits). However, the best known quantum algorithms for the hash-, code- and lattice-based problems require an exponential number of quantum gates. These presumably quantum-resistant problems also have some disadvantages. Hash-based signatures are long (tens of kilobytes), require a signature key per each message, and

Hard problem	Example scheme	Recomm. public key size, bits	Operation complexity	Encryption overhead, bits	Quantum complexity
Integer factoring [1]	RSA	$\frac{0.05(s+14)^3}{\log(s+14)^2}$	$O\left(\frac{s^6}{\log(s)^3}\right)$	0	$O\left(\frac{s^9}{\log(s)^5}\right)$
DLP in $\mathbb{F}_p^*$ [2]	ElGamal	$\frac{0.05(s+14)^3}{\log(s+14)^2}$	$O\left(\frac{s^6}{\log(s)^3}\right)$	$\frac{0.05(s+14)^3}{\log(s+14)^2}$	$O\left(\frac{s^9}{\log(s)^5}\right)$
Ell. curve DLP [3]	ECIES	$2s$	$O(s^{2.6})$	$4s$	$O(s^3)$
Hash collision and preimage [4]	Merkle's sign.	$2s$	$O(s$ hash$)$	$12s^2+20s$	$O(2^{0.67s}$ hash$)$
Linear code decoding [5]	McEliece	$2.25s^2\log(s)^2$	$O(s^2\log(s)^2)$	$1.5s\log(s)$	$O(2^{0.5s})$
Lattice shortest vector problem [6]	NTRU	$3s\log(s)+1000$	$O(s\log(s)^2)$	$2.5s\log(s)$+840	$O(2^s)$
Isogeny problem [7]	$\mathcal{PE}$	$4s-8\log(s)-16$	$O(s^{5.3})$	$4s-8\log(s)-16$	$O\left(2^{6\sqrt{s\log(s)}}\right)$

Table 1.1: Comparison of hard problems used in cryptography, for the security level of s bits. Provided values are approximated. All example schemes are public-key encryption schemes, except for the Merkle's digital signature scheme. In the column "Recommended public key size" we assume that common system parameters (e.g. the DLP group order) are not included in public keys. The "Operation complexity" column shows the asymptotic number of bit operations in one encryption or signature verification operation. The "Encryption overhead" column contains the difference between the ciphertext length and the "default" message length (e.g. the RSA message length is $\log(n)$ bits, where n is the modulus), or the length of a digital signature for the Merkle's scheme. This absolute overhead size is important when sending short messages over low-bandwidth channels, for instance SMS messages. The column "Quantum complexity" contains an O-bound for the expected number of quantum gates needed to solve a random problem instance.

[1] The security level of the RSA encryption scheme in $\mathbb{Z}/n\mathbb{Z}^*$ is $(64/9)^{1/3}\log(e)\ln(n)^{1/3}\ln(\ln(n))^{2/3}-14$ bits [41, p. 27], hence one should choose $\log(n)\approx 0.05(s+14)^3/\log(s+14)^2$. Decryption takes $O(\log(n)^2\log(\log(n)))$ bit operations with fast multiplication. Factoring n on a quantum computer requires $O(\log(n)^3\log(\log(n)))$ gates [4].

[2] The complexity of the DLP in a sufficiently large cyclic subgroup of $\mathbb{F}_p^*$ is similar to the complexity of factoring an RSA modulus n such that $\log(n)\approx\log(p)$ [41].

[3] We consider the Elliptic Curve Integrated Encryption Scheme (ECIES). In elliptic curve cryptosystems over $\mathbb{F}_q$ it is customary to choose q having $2s$ bits. We assume that the length of the message authentication code in the ECIES is $2s$, hence the overhead is $4s$. One point multiplication takes $O(\log(q)^{2.6})$ bit operations using the modular multiplication by Karatsuba's and Montgomery's algorithms. Solving the ECDLP on a quantum computer requires $O(\log(q)^3)$ gates [86, §6].

[4] The security of Merkle's signature scheme with an n-bit hash function is $n/2$ bits [10, p. 89]. We assume that the tree height is 10 and that the Lamport-Diffie one-time signature scheme is used. The size of Merkle's signature is $3n^2+10n$, verification requires $n+11$ hash evaluations. Quantum complexity is approximately $2^{n/3}$ hash evaluations.

[5] We consider the McEliece's hidden-Goppa-code public-key encryption scheme. Decoding a code of length n requires approximately $2^{(0.5+o(1))n/\log(n)}$ operations [9], hence $n\approx 3s\log(s)$. For a code of rate 0.5 the public key size is $n^2/4$, and the ciphertext of a $0.5n$-bit message is n bits long. Encryption takes $O(n^2)$ bit operations. The fastest quantum attack takes time approximately $2^{(0.25+o(1))n/\log(n)}$ [9].

[6] We consider the NTRU encryption scheme in $\mathbb{Z}_q[X]/(X^N-1)$ for $q=1024$. Based on [56, Table 5] recommended values of N roughly approximate as $0.3s\log(s)+100$. The public key size is $N\log(q)$, encryption is O-bound by the polynomial multiplication, thus it is $O(N\log(N))$ bit operations when q is constant. Encryption overhead is $N(\log(q)-\log(3))$ bits. Quantum complexity has been shown to approximately follow the classical one [117].

[7] We consider the $\mathcal{PE}$ public-key encryption scheme defined in Fig. 3.2. The complexity of the isogeny problem over $\mathbb{F}_q$ is proportional to $q^{1/4}\log(q)^2\log(\log(q))$ operations in $\mathbb{F}_q$ (see Chapter 4). Hence we take $\log(q)\approx 4s-8\log(s)-16$. One encryption takes $O(\log(q)^{3.7})$ field operations (see Section 2.5), and each modular multiplication is $O(\log(q)^{1.6})$ bit operations. Solving the isogeny problem over $\mathbb{F}_q$ with the quantum algorithm proposed by Childs, Jao and Soukharev [26] requires $\exp[(\sqrt{3}/2+\sqrt{2}+o(1))\sqrt{\ln(q)\ln(\ln(q))}]$ quantum operations.

the number of signatures that can be verified by one public key is limited. Code-based schemes require very long public keys (hundreds of kilobytes). Lattice-based schemes have relatively long public keys and overheads (kilobits).

The discussion above shows that it is important to examine *new* hard problems. In this work we investigate the potential of isogenies of elliptic curves for building secure cryptographic schemes.

Let K be a field of characteristic larger than 3. An *elliptic curve* E over the field K is a non-singular algebraic curve defined by the equation

$$Y^2 = X^3 + aX + b, \tag{1.1}$$

where $a, b \in K$. For a field $L \supseteq K$, the set of points $(x, y) \in L \times L$ satisfying (1.1), together with an extra "point at infinity" O, is denoted by $E(L)$. The set $E(L)$ is an additive abelian group with the zero element O.

For elliptic curves E_1 and E_2 defined over K, an *isogeny* ϕ from E_1 to E_2 is a group homomorphism

$$\phi : E_1(\overline{K}) \to E_2(\overline{K})$$

that is given by rational functions. If the coefficients of these rational functions lie in K, the isogeny ϕ is said to be defined over K. The x-coordinate map of an isogeny ϕ can be expressed as a univariate rational function $p(X)/q(X)$ in reduced form. The maximum of the degrees of $p(X)$ and $q(X)$ is called the *degree* of the isogeny ϕ. Two elliptic curves are said to be *isogenous* if there exists a non-constant isogeny between them. According to the theorem of Tate, two elliptic curves E_1 and E_2 over a finite field F are isogenous over F if and only if $\#E_1(F) = \#E_2(F)$.

Since the Schoof's algorithm and its extension, the Schoof-Elkies-Atkin algorithm, were proposed, isogenies have been used for efficient calculation of the number of points on an elliptic curve over a finite field [96]. Other cryptographic applications of isogenies include the reduction of the DLP between isogenous elliptic curves [45, 61], computation of the endomorphism ring of an elliptic curve [71], computation of modular polynomials [18, 23] and Hilbert class polynomials [6]. Isogenies have been applied in constructing key escrow systems, ordered signature schemes and hash functions. These cryptographic constructions are fundamentally different from our proposal, as we discuss later in Section 1.3.

For an elliptic curve E over a field K, the set of all isogenies $E(\overline{K}) \to E(\overline{K})$ defined over $\overline{K}$ is a ring under the addition $(\phi + \psi)(P) = \phi(P) + \psi(P)$ and the multiplication $(\phi\psi)(P) = \phi(\psi(P))$. It is called the *endomorphism ring* of E and denoted $\mathrm{End}_{\overline{K}}(E)$. An elliptic curve E over a finite field F is called *ordinary* if $\mathrm{End}_{\overline{F}}(E)$ is an order in an imaginary quadratic field, defined later in Section 1.4.

We consider the following problem as a potential hard problem suitable for cryptography:

Problem 1.1 (Isogeny Problem)**.** *Let E_1 and E_2 be ordinary elliptic curves over a finite field F satisfying $\#E_1(F) = \#E_2(F)$. Compute an isogeny $\phi : E_1(\overline{F}) \to E_2(\overline{F})$ defined over F.*

The isogeny problem was studied by Galbraith [44] and Galbraith, Hess and Smart [45]. Both papers proposed exponential-time algorithms, which indicates that the isogeny problem may indeed be interesting for cryptography.

The following questions naturally arise in the scope of our work:

Question 1. How can isogenies between ordinary elliptic curves be used for building cryptographic schemes? Which schemes can be built? What is the efficiency of such schemes?

Question 2. On which computational problems does the security of the proposed schemes depend?

Question 3. What is the computational complexity of these problems?

1.2 Our Work and Results

Answer to Question 1. Isogenies between ordinary elliptic curves allow efficient evaluation of a particular abelian group action: the class group action on a set of j-invariants of isogenous elliptic curves, defined later in Section 1.4. In Chapter 2 we build cryptographic schemes based on this group action. We describe implementation details and ways to improve efficiency, suggest a method for cryptosystem parameter selection, and report results of our practical implementation. Our implementation of the class group action is available in an open-source software package `ClassEll` [104].

The asymptotic complexity of one cryptographic operation, such as an encryption or a key agreement protocol run, is $O(s^{5.3})$ bit operations, where s is the desired security level in bits. Practical speed measurements of our unoptimized implementation show that for $s = 96$ one encryption takes under a minute with a dual-core desktop processor.

We propose isogeny-based cryptographic schemes for public-key encryption, authenticated key agreement, digital signature, secret-key encryption (i.e. the Pohlig-Hellman encryption scheme), no-key secret message transfer, and commitment. The proposed schemes are generalizations of the corresponding DLP-based schemes to the context of a group action on a set. In particular, let B be a cyclic multiplicative group of a prime order q, and let a DLP-based cryptographic scheme $\mathcal{S}$ be defined in B. If the only arithmetic operations involved in $\mathcal{S}$ are the exponentiation in B, the multiplication and the multiplicative inverse in $\mathbb{Z}_q$, then the scheme $\mathcal{S}$ can be defined in a more general setting where a group G (generalized from $\mathbb{Z}_q$) acts on a set X (generalized from B). Consequently, such a scheme $\mathcal{S}$ can be implemented using isogenies between ordinary elliptic curves.

Answer to Question 2. In Chapter 3 we provide security reductions for two of the proposed schemes, thus proving that their security relies on the hardness of

the isogeny problem and related computational problems. More specifically, our proofs use the decisional Diffie-Hellman group action problem (DDHAP), defined in Section 3.4. The class-group DDHAP is not harder than the isogeny problem, and currently there is no faster way to solve the class-group DDHAP than through the solution of the associated isogeny problem.

Answer to Question 3. In Chapter 4 we propose an algorithm for solving the isogeny problem that improves upon the previously known algorithm of Galbraith, Hess and Smart (GHS). Our improvement is due to the idea of modifying the random walk on the isogeny graph such that small-degree isogenies are used more frequently. This provides an order of magnitude speed-up for feasible problem sizes. Since our algorithm is currently the fastest, we conclude that the computational complexity of the isogeny problem over $\mathbb{F}_q$ is proportional to $q^{1/4}\log(q)^2\log(\log(q))$ operations in $\mathbb{F}_q$.

Sections 4.1–4.7 constitute a preprint of a joint paper with Prof. Steven Galbraith [43]. Below we identify author's independent contribution to this paper. The key idea of using smaller-degree isogenies more frequently first appeared in author's earlier work (see Section 2.6). The author then discussed the possibility of applying this idea to the GHS algorithm with Galbraith. Author's independent contribution included writing the original text of the paper. It was then edited and partially rewritten by Galbraith, but approximately 70 % of the text remained untouched. All experimental work was performed by the author using a computer cluster at the Department of Telematics, NTNU. Appendices 4.A–4.C were written by the author and were not included into the paper in order to reduce its size.

The results of our work can be summarized as follows. The isogeny problem can be used as a hard problem for building cryptographic schemes. It satisfies the two key requirements for a cryptographically interesting hard problem: an exponential complexity of the problem and a polynomial complexity of the cryptographic operations. However the latter complexity is also the main drawback of the isogeny-based schemes. In practice these schemes are currently slower than contemporary alternatives.

The isogeny problem is compared with some other hard problems in Table 1.1. We see that in the pre-quantum scenario, assuming no stupendous attacks on the elliptic-curve DLP are found, isogeny-based schemes have no advantage over the elliptic-curve-based ones. It only makes sense to use our schemes in off-line applications, such as a scheduled batch processing, where speed is not a concern. Isogeny-based schemes can also be used in combination with other hard problems in order to diversify the set of security assumptions, e.g. by onion-like encrypting a message with different algorithms.

In the post-quantum scenario, however, our schemes look more interesting. First of all, it is not clear whether the superpolynomial quantum attack of Childs, Jao and Soukharev [26] will pose a realistic threat. The attack requires $O(2^{6\sqrt{s\log(s)}})$

quantum gates. Physicists are in doubt about the possibility of large-scale quantum computations, because of errors introduced by the quantum decoherence [39]. If no key length adjustment will be needed to protect against the named attack, then the isogeny-based schemes will offer, in general, shorter keys and more efficient bandwidth usage, as compared to other quantum-resistant hard problems. But this will come at a cost of lower operational speeds. If, on the other hand, scientists will find a way to implement large quantum circuits, then the key length will have to be increased and the isogeny-based schemes will loose their advantage over the most promising post-quantum candidates, such as NTRU.

1.3 Related Work: Cryptographic Schemes Based on Isogenies

The use of isogenies for building cryptographic schemes is a recent topic. We shall list isogeny-based cryptosystem proposals in the order of their publication. We shall describe protocols that rely on the isogeny problem in separate figures, for completeness.

The first proposal known to the author is by Teske [112, 113]. In her key escrow system, a secret curve E_s isogenous to the public curve E_pb is stored at a trusted authority. E_pb is used in a conventional elliptic curve cryptosystem, while E_s is chosen such that the ECDLP is feasible using the Weil descent attack, hence E_s can be used for key escrow. Teske reasons that it is very difficult for an attacker to construct an isogeny from E_pb to a curve suitable for the Weil descent attack, because the proportion of such curves is approximately 2^{-68} when E_pb is defined over $\mathbb{F}_{2^{161}}$. This task is more complex than the isogeny problem in the sense that the attacker only knows E_pb and has to find any one of approximately 2^{12} curves among approximately 2^{80} curves isogenous to E_pb. The best solution seems to be to use a random walk on the isogeny graph (i.e. the graph consisting of isomorphism classes of elliptic curves connected by isogenies of degrees smaller than some bound l_max) starting from E_pb until a vulnerable curve is found.

Rostovtsev, Makhovenko and Shemyakina [90] described an ordered digital signature scheme, where an existing signature algorithm, e.g. ECDSA or the elliptic-curve Schnorr signature, is used to sign a sequence of documents. In order to impose the sequential ordering of signatures, an isogeny of small degree (2, 3 or 5) is applied to compute the elliptic curve and the points which are used for signing of the next document. This is convenient in the case of blind signatures, where the signer cannot add a sequence number or a timestamp to the document. Note that the isogeny problem is not used in this cryptosystem.

The author began to work on the topic of isogeny-based cryptographic schemes in his master's thesis in early 2004 [105]. Building on this, a preprint of our joint work with Prof. Alexander Rostovtsev was archived on-line in April 2006 [89]. We

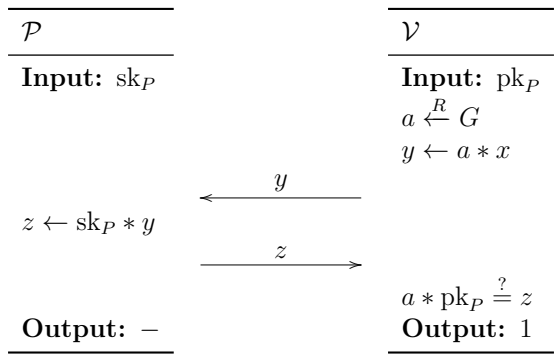

Figure 1.1: Two-pass authentication protocol of Couveignes [30].

proposed to build an ElGamal-like encryption scheme using isogenies between ordinary elliptic curves (a "hashed" variant of this scheme is presented in Fig. 3.2). We described the choice of system parameters, gave implementation details and briefly analysed the security of the proposed algorithm. A numerical example of encryption was included as well. This work was presented at the European Network of Excellence for Cryptology (ECRYPT) workshop Curves, Isogenies and Cryptologic Applications in July 2006.

A few months later a preprint by Couveignes [30] was made public. The preprint was written in 1997, but not published. Couveignes proposed to use homogeneous spaces (group acting simply transitively on a set) for constructing cryptographic primitives. He presented three protocols: a generalized Diffie-Hellman key exchange protocol (Fig. 3.1), a two-pass authentication protocol pictured in Fig. 1.1 and a Σ-protocol for authentication pictured in Fig. 1.2. Couveignes then proposed to use the action of imaginary quadratic ideals on ordinary elliptic curves for a conjectural hard homogeneous space. He also discussed implementation details and security. There are many similarities between Couveignes' work and our preprint [89] and Chapter 2. Presence of this independent research shows the importance of our topic.

All figures in this section use the following notation: a finite abelian group G acts on a set X, and an element $x \in X$ is publicly known. An entity P has a private key $\mathrm{sk}_P \in G$ and a public key $\mathrm{pk}_P = \mathrm{sk}_P * x$. The comparison operator $a \stackrel{?}{=} b$ is equivalent to "if $a \neq b$ then output 0". That is, when the comparison operator returns false, the algorithm immediately aborts and outputs zero, indicating a failure.

The Σ-protocol in Fig. 1.2 stems from the graph isomorphism proof by Goldreich, Micali and Wigderson [48, 49], and from the protocol for demonstrating possession of discrete logarithms proposed by Chaum, Evertse and van de Graaf [25, Protocol 1]. The protocol should be repeated t times to reduce the probability of cheating to 2^{-t}.

Charles, Goren and Lauter [24] designed a hash function based on an isogeny

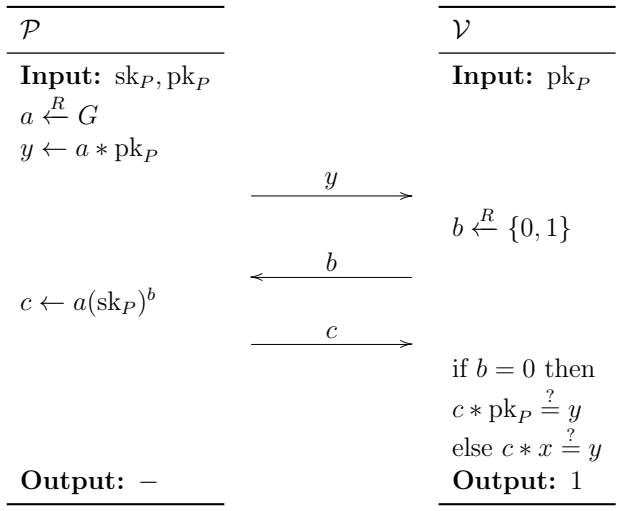

Figure 1.2: Σ-protocol of Couveignes [30].

graph of supersingular elliptic curves over $\mathbb{F}_{p^2}$. For any prime isogeny degree $l \neq p$, the l-isogeny graph of supersingular curves is $(l+1)$-regular. For a fixed hash function, the walk is always started from a fixed curve. For every hop of the walk, the input to the hash function is used to choose one of the l edges, without backtracking. The hash output is the ending vertex of the walk. We note that this construction is very different from ours. E.g., the endomorphism ring of a supersingular elliptic curve is isomorphic to an order in a quaternion algebra [100, Ch. V Theorem 3.1]. Since quaternion algebras are non-commutative, it is not straightforward how to use isogeny graphs of supersingular elliptic curves for constructing many of the schemes proposed in our work. This issue has been addressed by Jao and De Feo in their recent paper [60].

Weiwei and Debiao [120], based on our work [89], proposed to use isogenies between ordinary elliptic curves to implement the key agreement protocol of Popescu [85]. Popescu's protocol, generalized to the context of group action, is shown in Fig. 1.3. In addition to the notation introduced earlier, the protocol uses a hash function $H()$ which is publicly known. Popescu's protocol has the following security attributes proposed by Blake-Wilson, Johnson and Menezes [15]:

- *Known session key security.* A protocol still achieves its goal in the face of an adversary who has learned some previous session keys.
- *Forward secrecy.* If long-term secrets of one or more entities are compromised, the secrecy of previous session keys is not affected.
- *Unknown key-share.* Entity A cannot be coerced into sharing a key with entity

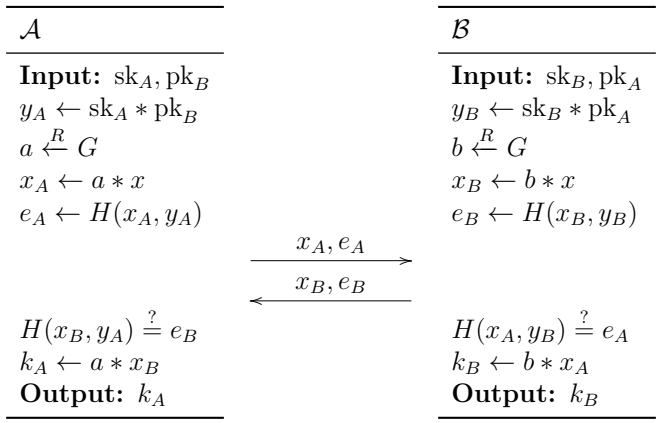

Figure 1.3: Generalized key agreement protocol of Popescu [85].

B without A's knowledge, i.e., when A believes the key is shared with some entity $C \neq B$.

Contrary to what is stated in [85, 120], the protocol does not provide security to *key-compromise impersonation*. If A's private key sk_A is compromised, then an adversary can impersonate other entities to A. E.g. to impersonate B to A, it is sufficient to compute $y_B = \mathrm{sk}_A * \mathrm{pk}_B$ and participate in the protocol with A on behalf of B.

A random number generator based on isogenies was proposed by Debiao, Jianhua and Jin [34]. The algorithm is seeded with a secret elliptic curve E and an isogeny path r_1. On round n the algorithm computes the curve $E_n = r_n * E$. The pseudo-random number output by the round n is $x_n = A_n \oplus B_n$, i.e. the XOR-ed equation coefficients of the curve E_n. The path r_{n+1} is computed as a function of x_n and n.

We note that starting from an output value x_n it is straightforward for an attacker to compute r_{n+1} (n is a small integer, in some circumstances it is known or can be guessed). Thus, by observing the algorithm's output, it is possible to collect pairs $(r_i, A_i \oplus B_i)$ satisfying $E_i = r_i * E$. This information might be used for calculating the secret curve E and for predicting the algorithm's output more efficiently than by brute-forcing.

One more authenticated key agreement protocol using isogenies was proposed by Debiao, Jianhua and Jin [35]. A generalized and slightly modified version of their protocol is shown in Fig. 1.4. The protocol of Debiao, Jianhua and Jin provides the security properties listed above. In addition to this it protects against the key-compromise impersonation. The authors also provide a security reduction in the Bellare-Rogaway authenticated key exchange model [7] with a random oracle.

Microsoft Corporation holds three patents [63, 64, 65] that cover various aspects of using isogenies for design of cryptosystems.

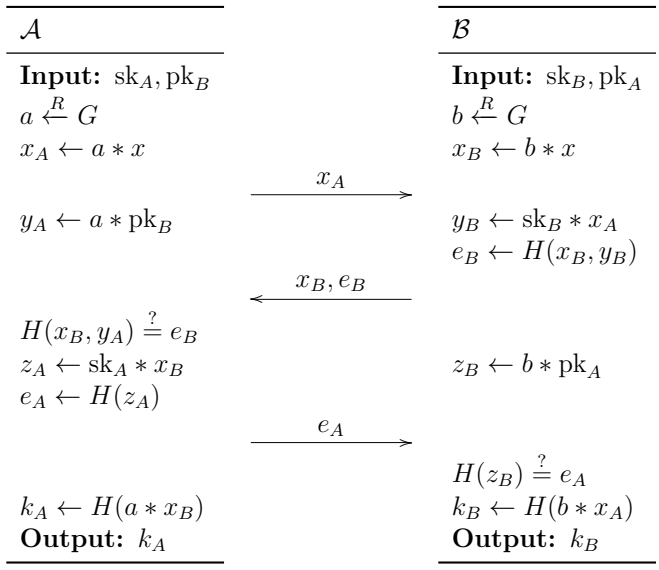

Figure 1.4: Generalized key agreement protocol of Debiao, Jianhua and Jin [35].

1.4 Background: the Class Group Action on the Set of j-invariants

In this section we introduce facts from algebraic number theory that will be necessary for our work. Further details can be found in textbooks of the following authors: Schertz [92, §10], Cox [31, §14], Lang [73, Part Two], Washington [118, §10] and Cohen [28, §7.2]. Throughout the section we provide a running example that extends the one given in Fig. 2.5. The example is marked by a vertical bar on the left.

A *number field* L is a field containing $\mathbb{Q}$ which, considered as a $\mathbb{Q}$-vector space, is finite dimensional. The dimension of L over $\mathbb{Q}$ is called the *degree* of the number field L.

Let θ be a root of the polynomial $A(X) = X^2+38$ over $\mathbb{C}$. Then $K = \mathbb{Q}(\theta)$ is a number field of degree 2. A basis of K over $\mathbb{Q}$ is $\{1, \theta\}$. The field K is isomorphic to the quotient ring $\mathbb{Q}[X]/\langle A\rangle$ by the map $(1, \theta) \mapsto (1 + \langle A\rangle, X + \langle A\rangle)$.

Theorem 1.1 ([28, Th. 4.1.8])**.** *Let L be a number field of degree n. Then there exists an element $\theta \in L$ such that $L = \mathbb{Q}(\theta)$. Its minimal polynomial A (i.e. the smallest degree polynomial with integer coefficients, such that $A(\theta) = 0$, A is primitive over $\mathbb{Z}$ and the leading coefficient of A is greater than zero) is an irreducible polynomial of degree n. There exist exactly n field embeddings of L in $\mathbb{C}$, given by $\theta \mapsto \theta_i$, where*

the θ_i are the roots in $\mathbb{C}$ of the polynomial A. The images L_i of these embeddings in $\mathbb{C}$ are isomorphic to L.

The images $L_i \subseteq \mathbb{C}$ are called the *conjugate fields* of L. A number field L is said to be *Galois* over $\mathbb{Q}$ if L is invariant by the n embeddings of L in $\mathbb{C}$ (in other words, if the conjugate fields L_i coincide).

Let A, θ and K be as above. The polynomial A is minimal, it has two complex roots: $\theta_1 = \sqrt{-38}$ and $\theta_2 = -\sqrt{-38}$. This gives two embeddings of K in $\mathbb{C}$, one defined by $\theta \mapsto \theta_1$ and the other one by $\theta \mapsto \theta_2$. Now the image of the first embedding, denoted $\mathbb{Q}(\sqrt{-38})$, is an isomorphic copy of K. The same holds for the image of the second embedding $\mathbb{Q}(-\sqrt{-38})$. It is easy to see that K is Galois, since $\theta_2 = -\theta_1$.

An element $\alpha \in \mathbb{C}$ is called an *algebraic integer* if there exists a monic (i.e. with leading coefficient equal to 1) polynomial $A \in \mathbb{Z}[X]$ such that $A(\alpha) = 0$, and A not identically zero. The set of algebraic integers of a number field L is a ring. It is called the *ring of integers* of L and denoted $\mathcal{O}_L$.

Theorem 1.2 ([28, Th. 4.4.2])**.** *The ring $\mathcal{O}_L$ is a free $\mathbb{Z}$-module (i.e. a module that has a basis) of rank equal to the degree of L.*

Let L be a number field of degree n, σ_i be the n embeddings of L in $\mathbb{C}$, and α_j be n algebraic integers forming a basis of $\mathcal{O}_L$ over $\mathbb{Z}$. Then $\Delta(L) = \det(\sigma_i(\alpha_j))^2$ is called the *discriminant* of the field L. The field discriminant is independent of the choice of an integral basis α_j.

The elements θ_1 and θ_2 defined above are examples of algebraic integers. The set $\{1, \theta\}$ is a $\mathbb{Z}$-basis of the ring of integers $\mathcal{O}_K$. Indeed, any element $\alpha = a + b\theta$, where $a, b \in \mathbb{Z}$, is a root of the monic polynomial $Y^2 - 2aY + a^2 + 38b^2$ with integer coefficients. Thus we can write $\mathcal{O}_K = \mathbb{Z} + \theta\mathbb{Z}$.

The discriminant of the field K is

$$\Delta(K) = \det \begin{pmatrix} \sigma_1(1) & \sigma_1(\theta) \\ \sigma_2(1) & \sigma_2(\theta) \end{pmatrix}^2 = \det \begin{pmatrix} 1 & \theta_1 \\ 1 & \theta_2 \end{pmatrix}^2 = (\theta_2 - \theta_1)^2 = (-2\sqrt{-38})^2 = -152.$$

An *integral ideal* $\mathfrak{a} \subseteq \mathcal{O}_L$ is a sub-$\mathbb{Z}$-module of $\mathcal{O}_L$ such that for every $\alpha \in \mathcal{O}_L$ and $a \in \mathfrak{a}$ we have $\alpha a \in \mathfrak{a}$.

Theorem 1.3 ([28, Pr. 4.6.3])**.** *Let $\mathfrak{a}$ be a non-zero integral ideal of $\mathcal{O}_L$. Then $\mathfrak{a}$ is a module of maximal rank. In other words, $\mathcal{O}_L/\mathfrak{a}$ is a finite ring.*

The cardinality of $\mathcal{O}_L/\mathfrak{a}$ is called the *norm* of $\mathfrak{a}$ and denoted $N(\mathfrak{a})$. An integral ideal $\mathfrak{a}$ is said to be *prime* if $\mathfrak{a} \neq \mathcal{O}_L$ and the quotient ring $\mathcal{O}_L/\mathfrak{a}$ is an integral domain (i.e. $xy \in \mathfrak{a}$ implies $x \in \mathfrak{a}$ or $y \in \mathfrak{a}$). In a ring of algebraic integers $\mathcal{O}_L$, a non-zero ideal $\mathfrak{a}$ is prime if and only if the quotient ring $\mathcal{O}_L/\mathfrak{a}$ is a field.

We write $a\mathbb{Z}+b\mathbb{Z}$ to denote the $\mathbb{Z}$-module generated by the elements a and b. Let $\mathfrak{a}_2 = 3\mathbb{Z}+(\theta+2)\mathbb{Z} \subset \mathcal{O}_K$. The quotient ring $\mathcal{O}_K/\mathfrak{a}_2$ consists of three elements: $0+\mathfrak{a}_2$, $1+\mathfrak{a}_2$ and $2+\mathfrak{a}_2$. Hence $N(\mathfrak{a}_2) = 3$. Furthermore, $\mathfrak{a}_2$ is a prime ideal.

A *fractional ideal* $\mathfrak{a}$ of a number field L is a non-zero submodule of L such that there exists a non-zero integer a with $a\mathfrak{a}$ an integral ideal of $\mathcal{O}_L$. It is clear that any integral ideal of $\mathcal{O}_L$ is also a fractional ideal.

The ideal $(3/2)\mathbb{Z} + (\theta/2+1)\mathbb{Z}$ is a fractional ideal that is not integral. Multiplication by 2 turns it into the integral ideal $\mathfrak{a}_2$.

The product of two fractional ideals $\mathfrak{a}$ and $\mathfrak{b}$ of $\mathcal{O}_L$ is defined as

$$\mathfrak{a}\mathfrak{b} = \left\{ \sum ij \mid i \in \mathfrak{a}, j \in \mathfrak{b} \right\}.$$

Theorem 1.4 ([28, Th. 4.6.14])**.** *Every fractional ideal of $\mathcal{O}_L$ is invertible. In other words, if $\mathfrak{a}$ is a fractional ideal and if we set $\mathfrak{a}^{-1} = \{\alpha \in L \mid \alpha\mathfrak{a} \subset \mathcal{O}_L\}$, then $\mathfrak{a}\mathfrak{a}^{-1} = \mathcal{O}_L$. The set of fractional ideals of $\mathcal{O}_L$ is an abelian group.*

We say that two fractional ideals $\mathfrak{a}$ and $\mathfrak{b}$ are *equivalent* if there exists $\alpha \in L^*$ such that $\mathfrak{b} = \alpha\mathfrak{a}$. The set of equivalence classes is called the *class group* of $\mathcal{O}_L$ and is denoted $\mathcal{CL}(\mathcal{O}_L)$.

Theorem 1.5 ([28, Th. 4.9.2])**.** *For any number field L, the class group $\mathcal{CL}(\mathcal{O}_L)$ is a finite abelian group.*

The cardinality of $\mathcal{CL}(\mathcal{O}_L)$ is called the *class number* and denoted $h(\mathcal{O}_L)$.

The class group of $\mathcal{O}_K$ consists of six equivalence classes of ideals represented by:

$$\begin{aligned}
\mathfrak{a}_1 &= \mathcal{O}_K, \\
\mathfrak{a}_2 &= 3\mathbb{Z} + (\theta+2)\mathbb{Z}, \\
\mathfrak{a}_3 &= 6\mathbb{Z} + (\theta+4)\mathbb{Z}, \\
\mathfrak{a}_4 &= 2\mathbb{Z} + \theta\mathbb{Z}, \\
\mathfrak{a}_5 &= 6\mathbb{Z} + (\theta-4)\mathbb{Z}, \\
\mathfrak{a}_6 &= 3\mathbb{Z} + (\theta-2)\mathbb{Z}.
\end{aligned}$$

The class group is isomorphic to the additive group $\mathbb{Z}/6\mathbb{Z}$ and is generated by $[\mathfrak{a}_2]$. Let us show that $[\mathfrak{a}_2^2] = [\mathfrak{a}_3]$ in $\mathcal{CL}(\mathcal{O}_K)$. First we find that $\mathfrak{a}_2^2 = 9\mathbb{Z} + (\theta+5)\mathbb{Z}$. To show the equivalence we take $\alpha = -2/3 + \theta/6$ and verify that $\alpha\mathfrak{a}_3 = \mathfrak{a}_2^2$.

A *lattice* Λ is an additive subgroup of $\mathbb{C}$ which is generated by two complex numbers which are linearly independent over $\mathbb{R}$. Let K be an *imaginary quadratic*

field (i.e. a number field of degree 2 with a negative discriminant). Then the ring of integers $\mathcal{O}_K$ and all non-zero fractional ideals of $\mathcal{O}_K$ are lattices.

The *j-invariant* $j(\Lambda)$ of the lattice Λ is defined to be the complex number

$$j(\Lambda) = 1728\frac{g_2(\Lambda)^3}{g_2(\Lambda)^3 - 27g_3(\Lambda)^2}, \quad \text{where} \tag{1.2}$$
$$g_2(\Lambda) = 60 \sum_{\omega\in\Lambda\setminus\{0\}} \frac{1}{\omega^4}, \quad g_3(\Lambda) = 140 \sum_{\omega\in\Lambda\setminus\{0\}} \frac{1}{\omega^6}.$$

Theorem 1.6 ([31, Pr. 10.7]). *If Λ is a lattice, then $g_2(\Lambda)^3 - 27g_3(\Lambda)^2 \neq 0$.*

To compute approximated j-invariants of the ideals $\mathfrak{a}_1, \ldots, \mathfrak{a}_6$ defined above we use Magma [16] with precision set to 25 decimal digits. For the lattices of ideals $\mathfrak{a}_i$ in the embedding $\mathbb{Q}(\sqrt{-38})$ we get the following j values:

$$j_1 = j(\mathfrak{a}_1) \approx 66246265662298399.44546049,$$
$$j_2 = j(\mathfrak{a}_2) \approx -201569.2502947957453982949 + 350415.9857691279142845026\sqrt{-1},$$
$$j_3 = j(\mathfrak{a}_3) \approx 247.9253742178749886512781 + 328.7306929095367517930038\sqrt{-1},$$
$$j_4 = j(\mathfrak{a}_4) \approx 257384243.2043806673300122,$$
$$j_5 = j(\mathfrak{a}_5) \approx 247.9253742178749886512781 - 328.7306929095367517930038\sqrt{-1},$$
$$j_6 = j(\mathfrak{a}_6) \approx -201569.2502947957453982949 - 350415.9857691279142845026\sqrt{-1}.$$

In the conjugate embedding $\mathbb{Q}(-\sqrt{-38})$ some j-invariants interchange: $\mathfrak{a}_2$ has the value j_6, $\mathfrak{a}_6$ has j_2, $\mathfrak{a}_3$ has j_5, and $\mathfrak{a}_5$ has j_3.

Theorem 1.7 ([31, Th. 10.9]). *If Λ and Λ' are lattices, then $j(\Lambda) = j(\Lambda')$ if and only if Λ and Λ' are homothetic (i.e. there exists $\alpha \in \mathbb{C}$ such that $\Lambda' = \alpha\Lambda$).*

Theorem 1.7 implies that equivalent fractional ideals of $\mathcal{O}_K$ have equal j-invariants.

For a fixed embedding σ of K in $\mathbb{C}$ we define the *action* $*$ of the class group $\mathcal{CL}(\mathcal{O}_K)$ on the set $\mathcal{ELL}_\sigma(\mathcal{O}_K)$ of j-invariants of the fractional ideals of $\mathcal{O}_K$ as follows:

$$\begin{aligned} * : \ \mathcal{CL}(\mathcal{O}_K) \times \mathcal{ELL}_\sigma(\mathcal{O}_K) &\to \mathcal{ELL}_\sigma(\mathcal{O}_K) \\ ([\mathfrak{a}], j(\mathfrak{b})) &\mapsto [\mathfrak{a}] * j(\mathfrak{b}) = j(\mathfrak{a}^{-1}\mathfrak{b}). \end{aligned}$$

Table 1.2 illustrates the action of the class group of $K = \mathbb{Q}(\sqrt{-38})$ on the set of j-invariants $\mathcal{ELL}_\sigma(\mathcal{O}_K)$.

Theorem 1.8 ([28, Th. 7.2.14]). *Let K be an imaginary quadratic field. Then $j(\mathcal{O}_K)$ is an algebraic integer of degree exactly equal to $h(\mathcal{O}_K)$. The minimal polynomial of $j(\mathcal{O}_K)$ over $\mathbb{Z}$ is $\prod_{i=1}^{h(\mathcal{O}_K)}(X - j(\mathfrak{a}_i))$, where $\mathfrak{a}_i$ runs over representatives of the equivalence classes of fractional ideals of $\mathcal{O}_K$.*

$[\mathfrak{a}] \in \mathcal{CL}$	Action $[\mathfrak{a}] * \mathcal{ELL}_\sigma$
$[\mathfrak{a}_1]$	$(j_1)(j_2)(j_3)(j_4)(j_5)(j_6)$
$[\mathfrak{a}_2]$	$(j_1\ j_6\ j_5\ j_4\ j_3\ j_2)$
$[\mathfrak{a}_3]$	$(j_1\ j_5\ j_3)(j_6\ j_4\ j_2)$
$[\mathfrak{a}_4]$	$(j_1\ j_4)(j_2\ j_5)(j_3\ j_6)$
$[\mathfrak{a}_5]$	$(j_1\ j_3\ j_5)(j_2\ j_4\ j_6)$
$[\mathfrak{a}_6]$	$(j_1\ j_2\ j_3\ j_4\ j_5\ j_6)$

Table 1.2: Action of $\mathcal{CL}(\mathcal{O}_K)$ on $\mathcal{ELL}_\sigma(\mathcal{O}_K)$, where $K = \mathbb{Q}(\sqrt{-38})$.

The minimal polynomial of $j(\mathcal{O}_K)$, for K quadratic of discriminant -152, is

$$\begin{aligned} B(X) = {} & X^6 - 66246265919280000X^5 + 17024071380555203520000000X^4 + \\ & 685454429479948368896000000000X^3 + \\ & 27830586247870936142929920000000000000X^2 - \\ & 13805041714261257587916800000000000000000X + \\ & 47239074813873128026931200000000000000000. \end{aligned}$$

The polynomial $B(X)$ can be obtained from the six j-invariant values given above, however a higher precision is needed to get the right coefficients.

Note that $\mathbb{Q}(j(\mathcal{O}_K))$ is not necessarily Galois. However, we have the following result:

Theorem 1.9 ([31, Cor. 11.34])**.** *If K is an imaginary quadratic field, then $K(j(\mathcal{O}_K))$ is a Galois extension of degree $h(\mathcal{O}_K)$ over K.*

The field $H = K(j(\mathcal{O}_K))$ is called the *Hilbert class field*, and the polynomial $B(X) = \prod_{i=1}^{h(\mathcal{O}_K)}(X - j(\mathfrak{a}_i))$ is called the *Hilbert class polynomial*.

Let η be a root of the polynomial $B(X)$, and consider the field $H = K(\eta)$. A K-basis of H is $\{1, \eta, \eta^2, \eta^3, \eta^4, \eta^5\}$. The polynomial $B(X)$ splits completely over the field H. Its six roots, written as elements of H, are:

$$\eta_1 = \eta,$$

$$\eta_2 = \frac{-6643170661453754199713403200721690\theta + 7036203355242182387338025312177151}{240323110893153632236073765614960414825266544640000000000000}\eta^5 + \frac{28952976985734469155991396300624543202650351\theta - 30665934114234015908980495215436503825275930 97}{158107309798127389628995898430895009753464832000000000000}\eta^4 + \frac{-88216837022837308843021098376082071885774095736973\theta + 937295675821346527895945423612868452157618980150267}{18775243038527627518443262938668782408223948880000000}\eta^3 + \frac{-88842282719856059420824700942966263586654813068762270\theta - 3362239162854072706390406789680238931702086109156 91}{2346905379815953439805407867333597801027993600000}\eta^2 + \frac{82478905162476953827089325204401620641793701331 30\theta + 29657802561821746541946427199083900499763011993}{4408161870428161983105574506637110820864 0}\eta + \frac{-65922867207607915872482524989630144998975\theta - 74555798412426582855883179741 2257255175}{10291082262424993612445813433856},$$

$$\eta_3 = \frac{14088897027990165610802646038468704230\theta - 7036203281986169006978219139140607}{2403231108931536322360737656149604148252665446400000000000000}\eta^5 + \frac{-12280747618757719626192535392563559917854379778 90\theta + 61331867589923645561986775980840565983663198 9}{3162146195962547792579917968617900195069296640000000000}\eta^4 + \frac{18738293518073914154890056857719860355245818099059365 10\theta - 937295669719857352364305135414528549040742827244539}{18775243038527627518443262938668782408223948880000000}\eta^3 + \frac{18873534699606812517294977107106127774677720232145086970\theta + 67244844740923864476524347571664589381162895633071}{4693810759631906879610815734667195602055987200 0}\eta^2 + \frac{59475931021734124082895691682505739354426930005113690\theta - 24491205351500718103486967387293931468859856 1}{3643108983824927258764937607 9100067840}\eta + \frac{-41634114309722434311112790258530551225626869375\theta + 34087288682216347629507375657564977106769129037 5}{10291082262424993612445813433856}.$$

Embedding	$\sigma(\eta_1)$	$\sigma(\eta_2)$	$\sigma(\eta_3)$	$\sigma(\eta_4)$	$\sigma(\eta_5)$	$\sigma(\eta_6)$
$\sigma_{11} : \theta \mapsto \sqrt{-38}, \eta \mapsto j_1$	j_1	j_2	j_3	j_4	j_5	j_6
$\sigma_{12} : \theta \mapsto \sqrt{-38}, \eta \mapsto j_2$	j_2	j_3	j_4	j_5	j_6	j_1
$\sigma_{13} : \theta \mapsto \sqrt{-38}, \eta \mapsto j_3$	j_3	j_4	j_5	j_6	j_1	j_2
$\sigma_{14} : \theta \mapsto \sqrt{-38}, \eta \mapsto j_4$	j_4	j_5	j_6	j_1	j_2	j_3
$\sigma_{15} : \theta \mapsto \sqrt{-38}, \eta \mapsto j_5$	j_5	j_6	j_1	j_2	j_3	j_4
$\sigma_{16} : \theta \mapsto \sqrt{-38}, \eta \mapsto j_6$	j_6	j_1	j_2	j_3	j_4	j_5
$\sigma_{21} : \theta \mapsto -\sqrt{-38}, \eta \mapsto j_1$	j_1	j_6	j_5	j_4	j_3	j_2
$\sigma_{22} : \theta \mapsto -\sqrt{-38}, \eta \mapsto j_2$	j_2	j_1	j_6	j_5	j_4	j_3
$\sigma_{23} : \theta \mapsto -\sqrt{-38}, \eta \mapsto j_3$	j_3	j_2	j_1	j_6	j_5	j_4
$\sigma_{24} : \theta \mapsto -\sqrt{-38}, \eta \mapsto j_4$	j_4	j_3	j_2	j_1	j_6	j_5
$\sigma_{25} : \theta \mapsto -\sqrt{-38}, \eta \mapsto j_5$	j_5	j_4	j_3	j_2	j_1	j_6
$\sigma_{26} : \theta \mapsto -\sqrt{-38}, \eta \mapsto j_6$	j_6	j_5	j_4	j_3	j_2	j_1

Table 1.3: Embeddings of the Hilbert class field $K(\eta)$ and the roots η_i in $\mathbb{C}$.

$$\eta_4 = -\frac{37241004347143131}{6108627540937402236689599108301191168000000000000}\eta^5 + \frac{81153864260029773771101227 49}{200941695425572441996368391720433920000000000}\eta^4 - \frac{62036022552911826789545432232483869}{954473053271469099482749860672061120000000}\eta^3 - \frac{19535256704748057815260482019925267}{74568207286833523397089832865004775 00}\eta^2 - \frac{98590292859218222240497622286383}{92601979853380014277576460705}\eta + \frac{92775223230304695511717881920 0}{14899755406818988620688087},$$

the coefficients of η_5 are the K-conjugates of the corresponding coefficients of η_3, and the coefficients of η_6 are the K-conjugates of the ones of η_2. The elements η_i are algebraic integers and hence lie in $\mathcal{O}_H$.

We now consider embeddings of H in $\mathbb{C}$. Table 1.3 lists how the field elements η_i map to the complex numbers j_i under different embeddings.

Let p be a prime number and $\mathfrak{p}$ a prime ideal of the ring of integers $\mathcal{O}_L$ of some number field L. Then $\mathfrak{p}$ is said to be a prime ideal *above* p if $\mathfrak{p} \cap \mathbb{Z} = p\mathbb{Z}$.

Theorem 1.10 ([28, Th. 4.8.3])**.** *Let p be a prime number. There exist positive integers e_i such that $p\mathcal{O}_L = \prod \mathfrak{p}_i^{e_i}$, where the $\mathfrak{p}_i$ are all the prime ideals above p.*

The prime p is said to be *inert* if $p\mathcal{O}_L = \mathfrak{p}$. p is said to *split completely* if $p\mathcal{O}_L = \prod_{i=1}^{n} \mathfrak{p}_i$, where n is the degree of L over $\mathbb{Q}$, and all $\mathfrak{p}_i$ are different. Finally, p is *ramified* if there is an e_i which is greater than or equal to 2 (in other words if $p\mathcal{O}_L$ is not squarefree).

In the quadratic number field K of discriminant -152, the prime 3 splits completely: $3\mathcal{O}_K = (3\mathbb{Z} + (\theta + 2)\mathbb{Z})\,(3\mathbb{Z} + (\theta - 2)\mathbb{Z})$; the prime 5 is inert because $5\mathcal{O}_K$ is a prime ideal; and the prime 19 is ramified: $19\mathcal{O}_K = (19\mathbb{Z} + \theta\mathbb{Z})^2$.

Theorem 1.11 ([31, Th. 5.26])**.** *Let H be the Hilbert class field of $K = \mathbb{Q}(\sqrt{-n})$, where n is squarefree, $n \not\equiv 3 \mod 4$, and let p be an odd prime not dividing n. Then p splits completely in H if and only if there exist integers x, y such that $p = x^2 + ny^2$.*

Field	$\overline{\eta_1}$	$\overline{\eta_2}$	$\overline{\eta_3}$	$\overline{\eta_4}$	$\overline{\eta_5}$	$\overline{\eta_6}$
$\mathcal{O}_H/\mathfrak{p}_1$	12	15	24	41	19	27
$\mathcal{O}_H/\mathfrak{p}_2$	19	27	12	15	24	41
$\mathcal{O}_H/\mathfrak{p}_3$	27	12	15	24	41	19
$\mathcal{O}_H/\mathfrak{p}_4$	15	24	41	19	27	12
$\mathcal{O}_H/\mathfrak{p}_5$	24	41	19	27	12	15
$\mathcal{O}_H/\mathfrak{p}_6$	41	19	27	12	15	24
$\mathcal{O}_H/\mathfrak{p}_7$	15	12	27	19	41	24
$\mathcal{O}_H/\mathfrak{p}_8$	24	15	12	27	19	41
$\mathcal{O}_H/\mathfrak{p}_9$	41	24	15	12	27	19
$\mathcal{O}_H/\mathfrak{p}_{10}$	12	27	19	41	24	15
$\mathcal{O}_H/\mathfrak{p}_{11}$	19	41	24	15	12	27
$\mathcal{O}_H/\mathfrak{p}_{12}$	27	19	41	24	15	12

Table 1.4: Reductions of η_i at prime ideals of $\mathcal{O}_H$ above 47.

Consider the prime $p = 47 = 3^2+38*1^2$. The ideal $p\mathcal{O}_H$ factors into a product of twelve distinct prime ideals $\mathfrak{p}_1, \ldots, \mathfrak{p}_{12}$ above p. Each of these ideals can be written as $\mathfrak{p}_i = 47\mathcal{O}_H + \alpha_i\mathcal{O}_H$. We give α_7 below; other ideal generators α_i can be obtained by running the following code in Magma:

```
D := -152;
K<t> := QuadraticField(D);
H<n> := ext< K | HilbertClassPolynomial(D) >;
p := Factorization(47*RingOfIntegers(H));
for i := 1 to #p do
  print "P", i; for g in Generators(p[i][1]) do print H!g; end for;
end for;
```

$$\alpha_7 = \frac{239\theta-4446}{3076135819432366492621744199871493309763411771392000000000000000}\eta^5 + \frac{-5386568045946380445\theta+20707375010996628762}{4047547130832061174502294999830912249688699699200000000000}\eta^4 + \frac{-237985090680904726807570250531128837\theta-76819373778551000431853994197009462}{240323110893153632236073765614960414825266544640000000000}\eta^3 + \frac{-889584356128270969488268327246861695856887\theta-237954892300936917295934149743545977706641 8}{60080777723288408059018441403740103706316636160000000}\eta^2 + \frac{501534662468414329673984774548642192814804 57\theta-703173561946129981796040175142352010734656 18}{5642447194148047338375135368495501850705920000}\eta + \frac{630659538848714846791091095280294046 9\theta-8144978113728473018895194891049092500 2}{1053806823672319345914451295626854 4}.$$

For the Hilbert class field H of an imaginary quadratic field K, and a prime ideal $\mathfrak{p}$ of $\mathcal{O}_H$, the quotient ring $F = \mathcal{O}_H/\mathfrak{p}$ is a finite field. Since the roots η_i of the Hilbert class polynomial $B(X)$ lie in $\mathcal{O}_H$, their images $\overline{\eta_i}$ in F are well-defined. The elements $\overline{\eta_i} \in F$ are called the *reductions* of η_i at $\mathfrak{p}$.

Table 1.4 lists reductions of the six algebraic integers η_i at the prime ideals $\mathfrak{p}_i$ above 47.

Reduction modulo a prime ideal of $\mathcal{O}_H$ induces the action $\overline{*}$ of the class group

$\mathcal{CL}(\mathcal{O}_K)$ on the set of reduced j-invariants. More specifically, for a fixed embedding σ of H in $\mathbb{C}$, the reduction modulo a prime ideal $\mathfrak{p}$ of $\mathcal{O}_H$ defines a bijection between the complex values j_i and their reductions $\overline{j_i}$ in the finite field $F = \mathcal{O}_H/\mathfrak{p}$. The action of $\mathcal{CL}(\mathcal{O}_K)$ on the set of reduced j-invariants $\mathcal{ELL}_{\sigma,\mathfrak{p}}(\mathcal{O}_K) \subset F$ is defined as follows:

$$\begin{aligned} \overline{*}: \ \mathcal{CL}(\mathcal{O}_K) \times \mathcal{ELL}_{\sigma,\mathfrak{p}}(\mathcal{O}_K) &\rightarrow \mathcal{ELL}_{\sigma,\mathfrak{p}}(\mathcal{O}_K) \\ \left([\mathfrak{a}], \overline{j(\mathfrak{b})}\right) &\mapsto [\mathfrak{a}] \,\overline{*}\, \overline{j(\mathfrak{b})} = \overline{j(\mathfrak{a}^{-1}\mathfrak{b})}. \end{aligned}$$

Choose one of the embeddings $\sigma_{11}, \ldots, \sigma_{16}$ listed in Table 1.3 and one of the ideals $\mathfrak{p}_7, \ldots, \mathfrak{p}_{12}$ listed in Table 1.4. Then Fig. 2.5 depicts the action $\overline{*}$ of $\mathcal{CL}(\mathcal{O}_K)$ on $\mathcal{ELL}_{\sigma,\mathfrak{p}}(\mathcal{O}_K)$ (cf. Table 1.2).

In our work we use the action $\overline{*}$ for building cryptographic schemes. In order to evaluate the action $\overline{*}$ we employ elliptic curves over the finite field $\mathcal{O}_H/\mathfrak{p}$, this is described in Sections 2.3–2.5. The action $\overline{*}$ can be evaluated in time polynomial of $\log(N(\mathfrak{p}))$, when the class group element is represented as a product of prime ideals of small norms. Another important result is that the inverse problem: given two j-invariants $\overline{j_1}$ and $\overline{j_2}$ find a fractional ideal $\mathfrak{a}$ such that $[\mathfrak{a}] * \overline{j_1} = \overline{j_2}$ (equivalently, the isogeny problem for curves with the endomorphism ring $\mathcal{O}_K$), has exponential complexity with the best up-to-date algorithm proposed in Chapter 4.

Paper I

Chapter 2, excluding Appendix 2.B, is published in

Anton Stolbunov

Constructing Public-Key Cryptographic Schemes Based on Class Group Action on a Set of Isogenous Elliptic Curves

Advances in Mathematics of Communications, Volume 4 (2010), 215–235

Chapter 2

Constructing Public-Key Cryptographic Schemes Based on Class Group Action on a Set of Isogenous Elliptic Curves

Abstract. We propose a public-key encryption scheme and key agreement protocols based on a group action on a set. We construct an implementation of these schemes for the action of the class group $\mathcal{CL}(\mathcal{O}_K)$ of an imaginary quadratic field K on the set $\mathcal{ELL}_{p,n}(\mathcal{O}_K)$ of isomorphism classes of elliptic curves over $\mathbb{F}_p$ with n points and the endomorphism ring $\mathcal{O}_K$. This introduces a novel way of using elliptic curves for constructing asymmetric cryptography.

2.1 Introduction

The two most practical mathematical problems constituting the basis for security of modern asymmetric cryptographic schemes are integer factoring and computing discrete logarithms. Security of the former problem has decreased fast as new factoring methods and computer technology are developed. The latter problem remains exponential-time for some groups, e.g. elliptic curves. However, the Shor's algorithm can solve factoring and discrete logarithm problems in polynomial time when sufficiently large quantum computer registers become available [98]. These facts put a need for the development of asymmetric schemes that are based on *new* hard computational problems.

A potential mathematical object for this purpose is a low degree isogeny graph of ordinary elliptic curves over a finite field. Vertices in this graph are elliptic curves and edges are morphisms between them. Among the popular applications of low degree isogenies are the elliptic curve point counting, e.g. the Schoof-Elkies-Atkin (SEA) algorithm [96], reduction of the elliptic curve discrete logarithm problem (ECDLP)

between different elliptic curves [45, 61], computation of the endomorphism ring of an elliptic curve [71] and computation of modular polynomials [18, 23]. Galbraith [44] and Galbraith, Hess and Smart [45] have proposed algorithms for constructing an isogeny between two given elliptic curves, i.e. searching for a route on an isogeny graph. Elliptic curve isogeny graphs have also been proposed for building cryptographic primitives. Rostovtsev et al. [90] have described an ordered digital signature scheme that implements the sequence number functionality for digitally signed documents using a small degree isogeny sequence. Teske [113] has constructed a key escrow system where a curve isogenous to the public curve is stored at a trusted authority and can be used to feasibly solve the ECDLP on the public curve, if needed. Charles, Goren and Lauter [24] have designed a hash function based on an isogeny graph of supersingular elliptic curves.

In this paper we use elliptic curve isogeny graphs for constructing new asymmetric cryptographic schemes. First we generalize some existing cryptographic schemes to the context of a group action on a set, and discuss their security. We then apply results of the complex multiplication theory to implement the proposed cryptographic schemes. Namely we use the action of the ideal class group $\mathcal{CL}(\mathcal{O}_K)$ of an imaginary quadratic field K on the set $\mathcal{ELL}_{p,n}(\mathcal{O}_K)$ of isomorphism classes of elliptic curves over $\mathbb{F}_p$ with n points and the endomorphism ring $\mathcal{O}_K$. The involved implementation-related solutions are then explained in more detail. We take advantage of available computational algorithms on elliptic curves and ideal class groups to implement the necessary operations. Finally we present our experimental results. Besides being interesting from the theoretical point of view, the proposed cryptographic schemes might also have an advantage against quantum computer attacks. However this question requires a further research, as we only point at the inapplicability of some currently known quantum algorithms.

This paper develops ideas of Rostovtsev and Stolbunov [89] originally appeared in their draft article. Independently of this work, research on a similar topic has been reported by Couveignes [30].

2.2 Public-Key Cryptography Based on Group Action

2.2.1 Notation

We start with the basic notation. Let G be a finite abelian group, and X a set. A (left) action of G on X is a map

$$\begin{aligned} G \times X &\to X \\ (g, x) &\mapsto g * x, \end{aligned}$$

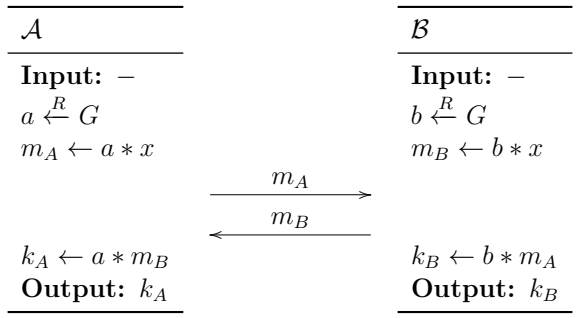

Figure 2.1: Key agreement protocol $\mathcal{KA}1$.

which satisfies the associativity property $(gh)*x = g*(h*x)$ for all $g, h \in G$, $x \in X$, and the property $e * x = x$ for the identity element $e \in G$ and all $x \in X$. The orbit of a set element $x \in X$ is the subset $G * x = \{g * x \colon g \in G\}$. The orbits of the elements of X are equivalence classes.

By $a \leftarrow b$ we denote the assignment of value b to a variable a. By $a \xleftarrow{R} G$ we mean that a is sampled from the uniform distribution on the set of elements of G. We write $\#S$ for the number of elements in S. By $\log n$ we denote the binary logarithm of n.

As a general rule, we assume that all algorithms take descriptions of G and X, and an element $x \in X$, as part of implied *system parameters*. By descriptions of G and X we mean the information needed, besides the input, to implement the operations involved in the algorithm, such as the random sampling from G, the action of G on X, the group operation etc.

2.2.2 Key Agreement Protocols Based on Group Action

We present three key agreement protocols based on the action of G on X. A key agreement protocol $\mathcal{KA}1$ pictured in Fig. 3.1 is a generalization of the Diffie-Hellman key agreement. The $\mathcal{KA}1$ protocol has been proposed by Monico [79]. By $\mathcal{A}$ and $\mathcal{B}$ we denote the algorithms run by the participants Alice and Bob, respectively.

Due to the commutativity of G and the associativity of the action, the following holds:

$$k_A = a * \beta = a * (b * x) = (ab) * x = (ba) * x = b * (a * x) = b * \alpha = k_B,$$

so $\mathcal{A}$ and $\mathcal{B}$ output the same session key k. The protocol $\mathcal{KA}1$ provides secrecy of k from passive adversaries.

The next two key agreement protocols make use of long-term public key pairs and thus require a one-time setup. Alice randomly chooses her *secret key* $\mathrm{sk}_A \in G$ and then computes the corresponding *public key* $\mathrm{pk}_A = \mathrm{sk}_A * x$. Alice then provides

$\mathcal{A}$		$\mathcal{B}$
Input: $\mathrm{sk}_A, \mathrm{pk}_B$		**Input:** $\mathrm{sk}_B, \mathrm{pk}_A$
$a \xleftarrow{R} G$		$b \xleftarrow{R} G$
$m_A \leftarrow (a\,\mathrm{sk}_A) * \mathrm{pk}_B$		$m_B \leftarrow (b\,\mathrm{sk}_B) * \mathrm{pk}_A$
	$\xrightarrow{m_A}$	
	$\xleftarrow{m_B}$	
$k_A \leftarrow a * m_B$		$k_B \leftarrow b * m_A$
Output: k_A		**Output:** k_B

Figure 2.2: Key agreement protocol $\mathcal{KA}2$.

$\mathcal{A}$		$\mathcal{B}$
Input: $\mathrm{sk}_A, \mathrm{pk}_B$		**Input:** $\mathrm{sk}_B, \mathrm{pk}_A$
$a \xleftarrow{R} G$		$b \xleftarrow{R} G$
$m_A \leftarrow a * \mathrm{pk}_B$		$m_B \leftarrow b * \mathrm{pk}_A$
	$\xrightarrow{m_A}$	
	$\xleftarrow{m_B}$	
$k_A \leftarrow (a\,\mathrm{sk}_A^{-1}) * m_B$		$k_B \leftarrow (b\,\mathrm{sk}_B^{-1}) * m_A$
Output: k_A		**Output:** k_B

Figure 2.3: Key agreement protocol $\mathcal{KA}3$.

her public key to Bob in an authentic manner. Bob does the same setup with his key pair $\mathrm{sk}_B, \mathrm{pk}_B$. Key Agreement 2 and 3 protocols are shown in Fig. 2.2 and 2.3, respectively.

$\mathcal{A}$ and $\mathcal{B}$ output the same session key k in the protocol $\mathcal{KA}2$ since

$$k_A = a * m_B = a * ((b\,\mathrm{sk}_B) * (\mathrm{sk}_A * x)) = (a\,b\,\mathrm{sk}_B\,\mathrm{sk}_A) * x = \\ = b * ((a\,\mathrm{sk}_A) * (\mathrm{sk}_B * x)) = b * m_A = k_B.$$

In the $\mathcal{KA}3$ protocol, $\mathcal{A}$ and $\mathcal{B}$ output the same session key k since

$$k_A = (a\,\mathrm{sk}_A^{-1}) * m_B = (a\,\mathrm{sk}_A^{-1}) * (b * (\mathrm{sk}_A * x)) = (ab) * x = \\ = (b\,\mathrm{sk}_B^{-1}) * (a * (\mathrm{sk}_B * x)) = (b\,\mathrm{sk}_B^{-1}) * m_A = k_B.$$

Whereas the $\mathcal{KA}1$ protocol does not provide authenticity, the protocols $\mathcal{KA}2$ and $\mathcal{KA}3$ are designed to provide mutual key authentication, i.e. Alice is assured that no other party aside from the one in possession of sk_B may gain access to k, and vice-versa. The protocols $\mathcal{KA}2$ and $\mathcal{KA}3$ are generalizations of the MTI/C1 and MTI/C0 protocols proposed by Matsumoto, Takashima and Imai [78, §12.6].

$\mathcal{K}$: Key generation
Input: -
$\mathrm{sk} \xleftarrow{R} G$
$y \leftarrow \mathrm{sk} * x$
$k \xleftarrow{R} K$
$\mathrm{pk} \leftarrow (y, k)$
Output: sk, pk

$\mathcal{E}$: Encryption
Input: pk, m
$a \xleftarrow{R} G$
$u \leftarrow a * y$
$h \leftarrow H_k(u)$
$z \leftarrow a * x$
$c \leftarrow h \oplus m$
$\mathrm{ct} \leftarrow (c, z)$
Output: ct

$\mathcal{D}$: Decryption
Input: sk, pk, ct
$u \leftarrow \mathrm{sk} * z$
$h \leftarrow H_k(u)$
$m \leftarrow h \oplus c$
Output: m

Figure 2.4: Public-key encryption scheme $\mathcal{PE}$.

2.2.3 Public-Key Encryption Based on Group Action

We now generalize the ElGamal public-key encryption scheme to the context of group action. An approach proposed by Monico [79] requires the set X to be a group in order to mask a message $m \in X$. In contrast with this, we use a "hashed" version of the ElGamal encryption scheme, which eliminates these restrictions on X and m through the use of a hash function family $\mathcal{H}$, which, however, introduces a need for a security assumption about $\mathcal{H}$ (see Theorem 2.2).

For a fixed message length w, the message space is the set of bit strings $\{0,1\}^w$, and thus we can write $m \in \{0,1\}^w$. We use a hash function family $\mathcal{H} = \{H_k \colon k \in K\}$ indexed by a finite set K, such that each H_k is a function

$$H_k \colon G * x \to \{0,1\}^w.$$

The public-key encryption scheme $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is pictured in Fig. 3.2. Some of the notation we use are: m is a message, $\mathrm{sk} \in G$ is a secret key, $\mathrm{pk} \in X \times K$ is a public key and $\mathrm{ct} \in \{0,1\}^w \times X$ is a ciphertext. We have that $x, y, z, u \in X$. The algorithm $\mathcal{K}$ also takes the description of K as part of implied system parameters.

The encryption scheme $\mathcal{PE}$ is sound, that is to say, for all pairs (sk, pk) which can be output by $\mathcal{K}()$ and for all $m \in \{0,1\}^w$ we have that $\mathcal{D}(\mathrm{sk}, \mathrm{pk}, \mathcal{E}(\mathrm{pk}, m)) = m$. Indeed, we can write $H_k(a * r) = H_k(\mathrm{sk} * y)$ since

$$a * r = a * (\mathrm{sk} * x) = (a\,\mathrm{sk}) * x = \mathrm{sk} * (a * x) = \mathrm{sk} * y.$$

2.2.4 Security of Cryptographic Schemes Based on Group Action

In this section we provide reductionist security arguments for the $\mathcal{KA}1$ protocol and the $\mathcal{PE}$ scheme, thus showing that breaching the security of these schemes is not easier than solving particular computational problems. A detailed discussion, together with proofs of Theorems 2.1 and 2.2, can be found in our earlier work [106].

For a finite abelian group G acting on a set X and a fixed element $x \in X$ we define the following computational problems:

Problem 2.1 (Group Action Inverse Problem (GAIP))**.** *Given a randomly chosen element $y \in G * x$, find a group element $g \in G$ such that $g * x = y$.*

A problem similar to GAIP was defined for semigroups by Maze et al. [77] as the semigroup action problem. However we prefer the name GAIP as the problem asks to invert the function $f(a) = a * x$.

Problem 2.2 (Decisional Diffie-Hellman Group Action Problem (DDHAP))**.** *Given a triple $(y, z, u) \in X^3$ sampled with probability $\frac{1}{2}$ from one of the two following probability distributions:*

- $(a * x,\ b * x,\ (ab) * x)$, *where a and b are randomly chosen from G,*
- $(a * x,\ b * x,\ c * x)$, *where a, b and c are randomly chosen from G,*

decide which distribution the triple is sampled from.

Using a GAIP solver it is straightforward to construct a solver for the DDHAP, thus the DDHAP is not harder than the GAIP.

For a DDHAP distinguisher $\mathcal{S}$, its probability of returning the correct solution will be denoted by $\Pr_{\mathcal{S}}^{\mathrm{DDH}}$. $\Pr_{\mathcal{S}}^{\mathrm{DDH}}$ is a function of a security parameter $s = \log \#\ G * x$. Since the distinguisher $\mathcal{S}$ can gain a success probability of $\frac{1}{2}$ by returning a random solution, the advantage of $\mathcal{S}$ is defined to be

$$\mathrm{Adv}_{\mathcal{S}}^{\mathrm{DDH}} = \left| \Pr_{\mathcal{S}}^{\mathrm{DDH}} - \frac{1}{2} \right|.$$

We can now define the following assumption about the computational complexity of the DDHAP:

Assumption 2.1 (DDHAP)**.** *For any polynomial-time DDHAP distinguisher $\mathcal{S}$, the advantage $\mathrm{Adv}_{\mathcal{S}}^{\mathrm{DDH}}$ is a negligible*[1] *function of s.*

To model the security of the $\mathcal{KA}1$ protocol we will use a notion of *session-key (SK) security in the authenticated-links adversarial model (AM)* proposed by Canetti and Krawczyk [21, 22]. In outline, this security notion asserts that any polynomial-time adversary $\mathcal{I}$ that cannot change the information transmitted between parties, does not learn anything about the value of the session key established between uncorrupted parties. This is formalized via the infeasibility for $\mathcal{I}$ to distinguish between the real value of the session key and an independent random value in a specially designed experiment. We refer to the papers of Canetti and Krawczyk for a

[1] A function $\mu(x)$ is negligible, if for every positive integer c there exists an $N_c > 0$ such that for all $x > N_c$, the following holds: $|\mu(x)| < 1/x^c$.

formal definition of the SK security in the AM. After a few implementation-specific modifications to the protocol $\mathcal{KA}1$, namely introducing party identifiers, session identifiers and requiring to erase variables a and b before returning the output, the following theorem can be proved:

Theorem 2.1. *If the DDHAP assumption holds for the finite abelian group G acting on the set X, then the $\mathcal{KA}1$ protocol is SK-secure in the AM.*

The classical goal of encryption is to preserve the privacy of messages: an adversary should not be able to learn from a ciphertext information about its plaintext beyond the length of that plaintext. This idea is captured via the notion of semantic security of an encryption scheme, proposed by Goldwasser and Micali [50], which asserts that any polynomial-time adversary cannot effectively distinguish between the encryption of two messages of his choosing. We will use an equivalent notion, indistinguishability of encryptions in a chosen-plaintext attack (IND-CPA) [8]. The equivalence of these two security notions has been shown by Goldreich [46].

In our security argument we will also use a property of a hash function family $\mathcal{H}$ to be entropy smoothing (ES). The smooth entropy denotes the number of almost uniform random bits in a random variable [19, 20]. The ES hash function should be able to produce almost uniformly distributed outputs by decreasing the output size, as compared to the size of the input. This is formalized via the requirement that any polynomial-time adversary cannot effectively distinguish between the values $(k, H_k(u))$ and (k, h), where $k \in K$, $u \in U$ and $h \in \{0,1\}^w$ are chosen at random, and U is the domain of the hash function. When applied to the $\mathcal{PE}$ scheme, U is the set of bit strings that represent the elements of $G * x$.

Assumption 2.2 (ES)**.** *The hash function family $\mathcal{H}$ is entropy smoothing.*

Theorem 2.2. *If the DDHAP assumption holds for the finite abelian group G acting on the set X, and the hash function family $\mathcal{H}$ is ES, then the public-key encryption scheme $\mathcal{PE}$ is secure in the sense of IND-CPA.*

We have shown that the security of the encryption scheme $\mathcal{PE}$ and the protocol $\mathcal{KA}1$ is based on the hardness of the DDHAP and the GAIP. The $\mathcal{PE}$ security is also subject to the ES assumption about the used hash function family.

2.3 Isogenous Elliptic Curves over Prime Fields

We show that elliptic curves provide an option for implementing the cryptographic schemes presented in Sect. 2.2. The aim of this section is to define mathematical structures that will be used as a set X and a finite abelian group G acting on X.

An elliptic curve E over a field F, $\mathrm{char}(F) \notin \{2, 3\}$, is an algebraic curve defined by an equation $y^2 = x^3 + Ax + B$, where $A, B \in F$ and $4A^3 + 27B^2 \neq 0$. The set of rational points on E over F, together with the "point at infinity" O, is an additive

group with the zero element O. The elliptic curve E over F is denoted E/F, and its group of rational points is denoted $E(F)$. For elliptic curves E_1/F and E_2/F, an isogeny between E_1 and E_2 is a morphism $\phi\colon E_1 \to E_2$ of varieties that satisfies $\phi(O) = O$. The elliptic curves E_1 and E_2 are called isogenous if there is a non-constant isogeny between them. Every isogeny $E_1 \to E_2$ induces a homomorphism of the groups $E_1(F)$ and $E_2(F)$. For an elliptic curve E/F, the set of all isogenies $E \to E$ defined over $\bar{F}$ is called the endomorphism ring of E and denoted $\mathrm{End}(E)$.

We review basic facts about elliptic curves over $\mathbb{C}$ to establish notation [101, Ch. II §1]. For a pair of complex numbers $\omega_1, \omega_2 \in \mathbb{C}$, such that $\omega_2/\omega_1 \notin \mathbb{R}$, the additive group $\Lambda = \omega_1\mathbb{Z} + \omega_2\mathbb{Z}$ is called a complex lattice. For a complex lattice Λ, the Weierstrass $\wp$-function gives rise to an elliptic curve

$$E_\Lambda/\mathbb{C}\colon y^2 = 4x^3 - g_2(\Lambda)x - g_3(\Lambda) \tag{2.1}$$

and a group isomorphism $\mathbb{C}/\Lambda \cong E_\Lambda(\mathbb{C})$. For any nonzero $\alpha \in \mathbb{C}$ the multiplication-by-α map induces the isomorphism $E_\Lambda(\mathbb{C}) \cong E_{\alpha\Lambda}(\mathbb{C})$, and for the endomorphism ring of E_Λ we have $\mathrm{End}(E_\Lambda) \cong \{\alpha \in \mathbb{C}\colon \alpha\Lambda \subset \Lambda\}$. When $\mathrm{End}(E_\Lambda)$ is larger than $\mathbb{Z}$, E_Λ is said to have complex multiplication, and $\mathrm{End}(E_\Lambda)$ is then isomorphic to an imaginary quadratic order.

Let $\mathcal{O}_K$ be the ring of integers of an imaginary quadratic field K. By $\mathcal{ELL}(\mathcal{O}_K)$ we denote the set of isomorphism classes of elliptic curves over $\mathbb{C}$ having the endomorphism ring $\mathcal{O}_K$:

$$\mathcal{ELL}(\mathcal{O}_K) = \frac{\{E/\mathbb{C}\colon \mathrm{End}(E) \cong \mathcal{O}_K\}}{\text{isomorphism over } \mathbb{C}}.$$

For convenience we will write $E \in \mathcal{ELL}(\mathcal{O}_K)$ meaning that E belongs to the corresponding isomorphism class $[E]$ in $\mathcal{ELL}(\mathcal{O}_K)$.

There is a well-defined action of the ideal class group $\mathcal{CL}(\mathcal{O}_K)$ on $\mathcal{ELL}(\mathcal{O}_K)$ given by

$$[\mathfrak{a}] * E_\Lambda = E_{\mathfrak{a}^{-1}\Lambda},$$

where $\mathfrak{a}$ is a fractional ideal, $[\mathfrak{a}]$ is the corresponding ideal class and Λ is a lattice with $E_\Lambda \in \mathcal{ELL}(\mathcal{O}_K)$.

For every ideal class $[\mathfrak{a}]$ there exists an integral ideal $\mathfrak{b} \subset \mathcal{O}_K$ for which $[\mathfrak{b}] = [\mathfrak{a}]$. Then $\Lambda \subset \mathfrak{b}^{-1}\Lambda$, and the homomorphism

$$\begin{array}{ccc} \mathbb{C}/\Lambda & \to & \mathbb{C}/\mathfrak{b}^{-1}\Lambda \\ z & \mapsto & z, \end{array}$$

induces a natural isogeny $\psi\colon E_\Lambda \to E_{\mathfrak{b}^{-1}\Lambda}$. The kernel of ψ is isomorphic to $\mathfrak{b}^{-1}\Lambda/\Lambda \cong \mathcal{O}_K/\mathfrak{b}$, and the degree of ψ equals the norm $N_{\mathbb{Q}}^K(\mathfrak{b})$.

We now reduce elliptic curves over $\mathbb{C}$ to the ones over a finite field[2] [31, §14.C]. Let H be the Hilbert class field of K, and $\mathcal{O}_H$ its ring of integers. Then the elliptic

[2]We use ordinary elliptic curves. With supersingular curves the endomorphism ring is noncommutative and so does not lead to an action by an abelian group. This is why our cryptographic schemes do not have the same efficiency as the hash function of Charles, Goren and Lauter [24].

curves in $\mathcal{ELL}(\mathcal{O}_K)$ are defined over H. Let $p > 3$ be a prime in $\mathbb{Z}$ which splits completely in H, and fix a prime $\mathfrak{P}$ of H lying above p, so that $\mathcal{O}_H/\mathfrak{P} \cong \mathbb{F}_p$. Finally, let $E \in \mathcal{ELL}(\mathcal{O}_K)$ be an elliptic curve which has good reduction at $\mathfrak{P}$. Hence g_2 and g_3 of (2.1) can be written in the form α/β where $\alpha, \beta \in \mathcal{O}_H$, $\beta \notin \mathfrak{P}$, so that $[g_2]$ and $[g_3]$ can be defined in $\mathcal{O}_H/\mathfrak{P}$. The elliptic curve $\bar{E}\colon y^2 = 4x^3 - [g_2]x - [g_3]$ over the finite field $\mathcal{O}_H/\mathfrak{P}$ is called the reduction of E modulo $\mathfrak{P}$. The reduction preserves the endomorphism ring, namely $\mathrm{End}_{\bar{\mathbb{F}}_p}(\bar{E}) \cong \mathcal{O}_K$, and every elliptic curve over $\mathbb{F}_p$ with the endomorphism ring $\mathcal{O}_K$ arises in this way. For two elliptic curves $E_1, E_2 \in \mathcal{ELL}(\mathcal{O}_K)$ that have good reduction at $\mathfrak{P}$, the natural reduction map $\mathrm{Hom}(E_1, E_2) \to \mathrm{Hom}(\bar{E}_1, \bar{E}_2)$ is injective and preserves degrees [101, Proposition II.4.4].

Let now $E_1/\mathbb{F}_p$ be an ordinary elliptic curve such that $\mathrm{End}(E_1) \cong \mathcal{O}_K$ for some imaginary quadratic field K. According to a theorem of Tate, E_1 is $\mathbb{F}_p$-isogenous to an elliptic curve $E_2/\mathbb{F}_p$ if and only if $\#E_1(\mathbb{F}_p) = \#E_2(\mathbb{F}_p)$ [110, §3 Theorem 1]. We denote $n = \#E_1(\mathbb{F}_p)$ and restrict ourselves to the isogeny class of elliptic curves with n points. We define a set of isomorphism classes of elliptic curves with the endomorphism ring $\mathcal{O}_K$ to be

$$\mathcal{ELL}_{p,n}(\mathcal{O}_K) = \frac{\{E/\mathbb{F}_p\colon \#E(\mathbb{F}_p) = n \text{ and } \mathrm{End}(E) \cong \mathcal{O}_K\}}{\text{isomorphism over } \bar{\mathbb{F}}_p}.$$

In terms of horizontal and vertical isogenies introduced by Kohel [71] this set corresponds to the surface, i.e. the topmost level, of the isogeny graph.

The reduction modulo $\mathfrak{P}$ maps $\mathcal{ELL}(\mathcal{O}_K)$ to $\mathcal{ELL}_{p,n}(\mathcal{O}_K)$ and induces the action of $\mathcal{CL}(\mathcal{O}_K)$ on $\mathcal{ELL}_{p,n}(\mathcal{O}_K)$. Since the reduction preserves isogeny degrees, for an action $[\mathfrak{a}] * E_1/\mathbb{F}_p = E_2/\mathbb{F}_p$ by an integral ideal $\mathfrak{a}$ there exists a separable $\mathbb{F}_p$-isogeny $\psi\colon E_1 \to E_2$ of degree $N^K_{\mathbb{Q}}(\mathfrak{a})$.

In order to shorten the notation, we will often write $\mathcal{ELL}_{p,n}$ or even $\mathcal{ELL}$ instead of $\mathcal{ELL}_{p,n}(\mathcal{O}_K)$, and $\mathcal{CL}$ instead of $\mathcal{CL}(\mathcal{O}_K)$, where we do not need to explicitly refer to $\mathcal{O}_K$, p and n.

Remarkably, the set $\mathcal{ELL}_{p,n}$ is a principal homogeneous space over the group $\mathcal{CL}$ [119, Theorem 4.5]. In particular, $\mathcal{CL}$ acts freely and transitively on $\mathcal{ELL}_{p,n}$, i.e. for any $x, y \in \mathcal{ELL}_{p,n}$ there exists precisely one $g \in \mathcal{CL}$ such that $g * x = y$. It follows that $\#\mathcal{CL} = \#\mathcal{ELL}_{p,n}$.

For the sake of a small example, consider the elliptic curve $E\colon y^2 = x^3 + x + 5$ over the field $\mathbb{F}_{47}$. E has 42 points, the Frobenius discriminant $\Delta_\pi = -152$ is fundamental and $j_E = 27$. Figure 2.5 shows the action of $\mathcal{CL}(\mathcal{O}_{-152})$ on $\mathcal{ELL}_{47,42}(\mathcal{O}_{-152})$. Nodes of the graph are the j-invariants of the elliptic curves in $\mathcal{ELL}_{47,42}(\mathcal{O}_{-152})$. Ideal classes are represented by the reduced binary quadratic forms. Permutations are written in the cyclic notation, that is (27 15 41) means that the ideal class $[(6, 4, \cdot)]$ maps the curves with j-invariant 27 to curves with j-invariant 15, 15 to 41 and 41 to 27. Since the ideal $7\mathbb{Z} + \frac{-4+\sqrt{-152}}{2}\mathbb{Z}$ belongs to the ideal class $[(6, 4, \cdot)]$ and has norm 7, isogenies of degree 7 form cycles (27 15 41) and (19 12 24).

Elements of $\mathcal{CL}(\mathcal{O}_{-152})$	Permutations on $\mathcal{ELL}_{47,42}(\mathcal{O}_{-152})$
$g = [(3, 2, \cdot)]$	(27 12 15 24 41 19)
$g^2 = [(6, 4, \cdot)]$	(27 15 41)(19 12 24)
$g^3 = [(2, 0, \cdot)]$	(27 24)(19 15)(41 12)
$g^4 = [(6, -4, \cdot)]$	(27 41 15)(19 24 12)
$g^5 = [(3, -2, \cdot)]$	(27 19 41 24 15 12)
$g^6 = [(1, 0, \cdot)]$	(27)(19)(41)(24)(15)(12)

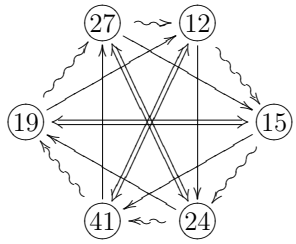

Figure 2.5: $\mathcal{CL}(\mathcal{O}_{-152})$ action on $\mathcal{ELL}_{47,42}(\mathcal{O}_{-152})$.

Thus for an abelian group G and a set X defined by

$$G = \mathcal{CL}(\mathcal{O}_K),$$
$$X = \mathcal{ELL}_{p,n}(\mathcal{O}_K)$$

it is possible to implement the cryptographic schemes proposed in Sect. 2.2. The next three sections describe the implementation in greater detail.

2.4 Elements of $\mathcal{ELL}_{p,n}$ and $\mathcal{CL}$

We show how one can store elements of X and G when implementing the cryptographic schemes of Sect. 2.2 on a set of isomorphism classes of elliptic curves.

2.4.1 Elements of $\mathcal{ELL}_{p,n}$

Elements of a set $\mathcal{ELL}_{p,n}$ are isomorphism classes of elliptic curves. Since two elliptic curves are isomorphic over $\bar{\mathbb{F}}_p$ if and only if they have the same j-invariant [100, Proposition III.1.4.b], we represent the elements of $\mathcal{ELL}_{p,n}$ by j-invariants. As j-invariants lie in $\mathbb{F}_p$ they can be stored as non-negative integers less than p.

However, for calculations described in Sect. 2.5 one will need an explicit equation of a curve $E \in \mathcal{ELL}_{p,n}$ with a given j-invariant. Here we see two ways for implementation: either to transmit the explicit curve equation, or to transmit the j-invariants and compute the equations when it is needed. The former approach saves computational resources while the latter one saves bandwidth. In the latter case, the elliptic curve equation can be obtained in the following way. At first one sets $c = j/(j - 1728)$ and considers the curve $E\colon y^2 = x^3 - 3cx + 2c$. E is either a desired elliptic curve or a twist, i.e. $\#E(\mathbb{F}_p)$ correspondingly equals n or $2p + 2 - n$. When these numbers are relatively prime, $\#E$ can be tested by taking a random point on E and multiplying it by n. If in the twisted case, one takes a quadratic non-residue $v \in \mathbb{F}_p^*$ and sets the desired curve to be $E'\colon y^2 = x^3 - 3cv^2x + 2cv^3$.

2.4.2 Elements of $\mathcal{CL}$

Elements of an ideal class group $\mathcal{CL}(\mathcal{O}_K)$ are classes of fractional ideals modulo principal fractional ideals of the imaginary quadratic order $\mathcal{O}_K$. A traditional approach is to use reduced ideals or, equivalently, reduced binary quadratic forms, as representatives of the ideal classes. However for storing elements of $\mathcal{CL}$ we will use a different approach, which is more suitable in the context of the action on $\mathcal{ELL}_{p,n}$.

We first recall an important result by Kohel [71]. For an ordinary elliptic curve $E/\mathbb{F}_p$ with endomorphism ring $\mathcal{O}_K$ of discriminant Δ, and a prime l, there are $\left(\frac{\Delta}{l}\right)+1$ isogenies of degree l to curves with endomorphism ring isomorphic to $\mathcal{O}_K$. We immediately observe that inert primes, namely those satisfying $\left(\frac{\Delta}{l}\right)=-1$, do not appear as degrees of isogenies inside $\mathcal{ELL}_{p,n}$, and therefore will not be further considered. Ramified primes, that is when $l \mid \Delta$, yield only one horizontal isogeny of degree l. Due to the existence of dual isogenies, isogenies of a ramified degree form loops of length 2. In other words, since $l\mathcal{O}_K = \mathfrak{l}^2$, the ideal $\mathfrak{l}$ has order 2 in $\mathcal{CL}$. As compared to split primes, a ramified isogeny degree l does not introduce much diversity when moving on the isogeny graph, because the number of hops by l-isogenies has to be considered modulo 2. When l is split, there are two horizontal isogenies of degree l, and l-isogeny cycles can be much longer. For this reason it is beneficial to select an elliptic curve with an endomorphism ring where most of the small primes are split, when choosing the system parameters. Our further attention will be concentrated on split primes.

For a fixed positive integer $l_{\max}$ let L be an indexed set

$$L = \left\{ \text{primes } l_i \colon \left(\frac{\Delta}{l_i}\right) = 1 \text{ and } l_i \le l_{\max} \right\}. \tag{2.2}$$

Let also $d = \#L$. Then for each i, $1 \le i \le d$, we define a prime ideal $\mathfrak{l}_i$ of $\mathcal{O}_K$ to be

$$\mathfrak{l}_i = l_i\mathbb{Z} + \frac{b_i + \sqrt{\Delta}}{2}\mathbb{Z}, \tag{2.3}$$

where $b_i = \min\{b \in \mathbb{N} \colon b^2 \equiv \Delta \pmod{4l_i}\}$. Now a homomorphism

$$\begin{aligned} \phi \colon \quad \mathbb{Z}^d \quad &\to \mathcal{CL} \\ (v_1, \ldots, v_d) \quad &\mapsto \prod_{i=1}^{d} [\mathfrak{l}_i]^{v_i} \end{aligned} \tag{2.4}$$

lets us use vectors in $\mathbb{Z}^d$ to store elements of $\mathcal{CL}$. Obviously, the addition of vectors corresponds to the multiplication in $\mathcal{CL}$, and the additive vector inverse is to be used as the ideal class inverse (used in the $\mathcal{KA}3$ protocol).

Remark 2.1. Whether the map ϕ is onto depends on the choice of L. It is proved that, if the generalized Riemann hypothesis (GRH) is true, then there exists a constant c_0 such that for $l_{\max} = c_0 \log^2 |\Delta|$ in the setting above, the ideal classes $[\mathfrak{l}_i]$,

$1 \leq i \leq d$, generate the group $\mathcal{CL}$ [52, Theorem 2]. In practice for a majority of ideal class groups it suffices to take $c_0 = 0.5$. Moreover, it is believed that the average minimum value of $l_{\max}$ needed to generate $\mathcal{CL}$ is $O(\log^{1+\epsilon}|\Delta|)$ for any $\epsilon > 0$ [5]. In our practical experiments we chose $l_{\max} \approx \log|\Delta|$. Note that when the structure of $\mathcal{CL}$ is pre-computed during the system parameters selection, the fact that the prime ideals $\mathfrak{l}_i$ of norms in L generate $\mathcal{CL}$ can be tested and nonconforming class groups can be discarded. Otherwise, when the structure of $\mathcal{CL}$ is not pre-computed, the fact that any element of $\mathcal{CL}$ can be represented as a product in (2.4) depends on the GRH and the choice of $l_{\max}$.

2.4.3 Generating the System Parameters

In order to choose the set $\mathcal{ELL}_{p,n}$ one first picks a prime p of sufficient size (see Sect. 2.7.1). One then tries arbitrary elliptic curves over $\mathbb{F}_p$ until an appropriate curve is found. For every curve E one computes $\#E(\mathbb{F}_p)$ using the SEA algorithm. The trace t of the Frobenius endomorphism $\pi\colon (x, y) \mapsto (x^p, y^p)$ is then obtained as $t = p + 1 - \#E(\mathbb{F}_p)$, and to test that E is ordinary one verifies that $t \neq 0$. The Frobenius discriminant Δ_π is then obtained by the formula

$$\Delta_\pi = t^2 - 4p. \tag{2.5}$$

The discriminant Δ of $\mathrm{End}(E)$ satisfies

$$\Delta = \Delta_\pi / g^2 \tag{2.6}$$

for some integer g. A straightforward way to ensure that $\mathrm{End}(E)$ is a maximal imaginary quadratic order is to check whether Δ_π is a fundamental discriminant. Besides, when the conductor of Δ_π is not divisible by a large prime, one may use an algorithm of Kohel [71] to move from E to a curve with the maximal endomorphism ring. In this case one should also check that $l_i \nmid g$ for all $l_i \in L$, so that there are no l_i-isogenies down.

Once Δ is known, one chooses $l_{\max}$ (see Remark 2.1) and examines how many of the primes less or equal to $l_{\max}$ are split. The more small primes are split the better performance the $\mathcal{CL}$ action will have. One can even do an "early abort" of the SEA algorithm when t is computed modulo small primes and too few of them satisfy $(\frac{t^2-4p}{l_i}) = 1$. When the curve is chosen, the structure of $\mathcal{CL}(\mathcal{O}_\Delta)$ should be computed, as we discuss later in Sect. 2.6.

The obtained elliptic curve $E \in \mathcal{ELL}_{p,n}$ is to be used as the set element x in the proposed cryptographic schemes.

2.5 Implementation of $\mathcal{CL}$ Action on $\mathcal{ELL}_{p,n}$

We show how to implement the group action $*$ used in the cryptographic schemes of Sect. 2.2. Let an elliptic curve $E \in \mathcal{ELL}_{p,n}$ be defined by $y^2 = x^3 + Ax + B$, and

an ideal class be given by a vector $\vec{v} \in \mathbb{Z}^d$ according to the notation of Sect. 2.4.2. Our aim is to compute $\phi(\vec{v}) * E \in \mathcal{ELL}_{p,n}$.

From (2.4) we have that

$$\phi(\vec{v}) * E = (\prod_{i=1}^{d} [\mathfrak{l}_i]^{v_i}) * E. \tag{2.7}$$

The associativity property of the group action implies that we can compute (2.7) by gradually acting by the factors $[\mathfrak{l}_i]$ or $[\mathfrak{l}_i]^{-1}$ in (2.7), depending on the sign of v_i. We call the operation $[\mathfrak{l}_i] * E = E_1$ a hop, this corresponds to an isogeny $E \to E_1$ of degree l_i. The computation of (2.7) consists of $\sum_{i=1}^{d} |v_i|$ hops between elliptic curves. We further explain the implementation of a single hop.

Throughout this section we use a notation

$$(l, b, \cdot) = l\mathbb{Z} + \frac{b + \sqrt{\Delta}}{2}\mathbb{Z}$$

for the prime ideals $\mathfrak{l}_i$ defined by (2.3), in order to explicitly refer to b. Since $\left(\frac{\Delta}{l}\right) = 1$, the prime l is split and we can write $l\mathcal{O}_K = (l, b, \cdot)(l, -b, \cdot)$. It follows that $[(l, b, \cdot)]^{-1} = [(l, -b, \cdot)]$, and for the negative coordinates in $\vec{v}$ we should use the action by $[(l, -b, \cdot)]$.

To compute the elliptic curve $E_1 = [(l, b, \cdot)] * E$ we apply ideas used in the SEA algorithm [29, 96]. For the action $[(l, b, \cdot)] * E = E_1$ there exists a separable $\mathbb{F}_p$-isogeny $\psi \colon E \to E_1$ of degree $N_{\mathbb{Q}}^{K}(l, b, \cdot) = l$. The same holds for $[(l, -b, \cdot)] * E = E_2$. The j-invariants of E_1 and E_2 are computed as roots of the equation

$$\Phi_l(x, j(E)) = 0 \pmod{p}, \tag{2.8}$$

where Φ_l is the modular polynomial of level l. When l does not divide the conductor g of Δ_π, the equation (2.8) has exactly 2 roots. We should determine which root is the j-invariant of the curve E_1. For that we take one of the roots $\hat{\jmath}$ and apply the algorithm of Elkies to compute the equation of an isogenous elliptic curve $\hat{E}$, $j(\hat{E}) = \hat{\jmath}$, and the polynomial $\hat{h}(x)$ that vanishes on the l-isogeny $E \to \hat{E}$ kernel.

The kernel of an l-degree isogeny is a subgroup of the l-torsion group, and the Frobenius endomorphism π on the kernel points satisfies the characteristic equation

$$\pi^2 - t\pi + p \equiv 0 \pmod{l}. \tag{2.9}$$

For split l the equation (2.9) has two different roots $\pi_1, \pi_2 \in \mathbb{Z}_l$ called Frobenius eigenvalues. These are related to the ideals $(l, b, \cdot)$ and $(l, -b, \cdot)$ by the following formula:

$$\pi_{1,2} \equiv \frac{t \pm gb}{2} \pmod{l},$$

where t is the trace of the Frobenius endomorphism. Indeed, we have that $b^2 \equiv \Delta \pmod{4l}$, thus $g^2b^2 \equiv \Delta_\pi \pmod{4l}$ and $(t+gb)/2$ satisfies (2.9):

$$\left(\frac{t+gb}{2}\right)^2 - t\frac{t+gb}{2} + p = \frac{g^2b^2 - \Delta_\pi}{4} \equiv 0 \pmod{l}.$$

The same holds for $(t-gb)/2$. The eigenvalue $\pi_1 \equiv (t-gb)/2 \pmod{l}$ corresponds to the action of π on the kernel of the isogeny associated with $(l, b, \cdot)$ and $\pi_2 \equiv (t+gb)/2 \pmod{l}$ corresponds to the isogeny associated with $(l, -b, \cdot)$.

We then check that the eigenvalue π_1 satisfies the relation

$$(x^p, y^p) \equiv [\pi_1](x, y) \pmod{y^2 - x^3 - Ax - B,\ \hat{h}(x)}, \tag{2.10}$$

where $[\pi_1](x, y)$ stands for the point multiplication by π_1. If (2.10) holds, we set the resulting elliptic curve E_1 to be $\hat{E}$. Otherwise E_1 is obtained from the second root of (2.8).

The computational complexity of a single hop between l-isogenous elliptic curves is dominated by solving the equation (2.8). The degree of the polynomial $f(x) = \Phi_l(x, j(E))$ is $l+1$. The roots are found by computing the $\gcd(x^p - x, f(x))$. The left-right binary exponentiation $x^p \pmod{f(x)}$ takes $\log p$ polynomial squarings, and multiplications by x are given "for free". Each polynomial multiplication through the number-theoretic transform (NTT) requires $O(l \log l)$ field multiplications. Also the division with remainder of the $2l$-degree product polynomial by $f(x)$ can be implemented in $O(l \log l)$ multiplications in $\mathbb{F}_p$ [12]. In practice for l less than approximately 70 these operations are faster implemented with the "schoolbook" and related algorithms, but for the asymptotic analysis we use the $O(l \log l)$ estimation anyway. This results in a total of $O(l \log l \log p)$ field multiplications needed to compute $x^p \pmod{f(x)}$. The GCD of l-degree polynomials is computed with $O(l \log^2 l)$ field multiplications [12], and since $\log l < \log p$, the resulting complexity of solving $f(x) = 0$ is $O(l \log l \log p)$.

The second most demanding operation in an l-isogenous hop is the verification of (2.10). A substitution $y^{p-1} = (x^3 + Ax + B)^{(p-1)/2}$ allows the exponentiation to be performed in the univariate polynomial ring $\mathbb{F}_p[x]/h_1(x)$. Since the degree of $h_1(x)$ is $(l-1)/2$, the binary exponentiation requires $O(l \log l \log p)$ field multiplications. When several consecutive hops are done along the same isogeny degree l, the verification of (2.10) is needed only on the first hop. This is because when moving along a cycle, at the second and the subsequent hops we know where we came from.

Thus the running time of one hop between l-isogenous elliptic curves is

$$O(l \log l \log p) \tag{2.11}$$

multiplications in $\mathbb{F}_p$.

To estimate the average running time of the action (2.7), we use the following approximations: $h \approx 0.46(-\Delta)^{1/2}$ for the class number $\#\,\mathcal{CL}(\mathcal{O}_\Delta)$ [28]; $l_{\max} \approx \log|\Delta|$

for the biggest prime in L (see Remark 2.1); $\Delta = cp$, where c is a constant (follows from (2.5), the Hasse's bound $|t| \leq 2\sqrt{p}$, (2.6) and the fact that the conductor g is chosen to be small during the system parameter generation); $l_i \approx 2i \ln 2i$ for the value of the i-th prime in L; and $d \approx \frac{1}{2} l_{\max} / \ln l_{\max}$ for the number of primes in L. In the last two approximations we use the prime number theorem and the fact that almost half of the primes are split. Let each v_i take values from an interval $[-\hat{v}, \hat{v}]$. The number of allowed values for each v_i approximates as $\log^{1/\log e} p$, since being raised to power d it gives $p^{1/2}$ possible vectors. The average running time of the group action (2.7) is then $O(\sum_{i=1}^{d} \log^{0.7} p \; l_i \log l_i \log p)$. We use the fact that $\sum_{i=1}^{d} i \ln^2 i$ is $O(d^2 \ln^2 d)$ and the above approximation for d. This gives

$$O\left(\log^{3.7} p\right)$$

multiplications in $\mathbb{F}_p$ needed for an average action of $\mathcal{CL}$ on $\mathcal{ELL}_{p,n}$.

2.6 Implementation of Sampling from $\mathcal{CL}$

We show how to implement the sampling operation $\cdot \xleftarrow{R} G$ used in the cryptographic schemes of Sect. 2.2.

2.6.1 Random Sampling from $\mathcal{CL}$

We use the notation for L, $\mathfrak{l}_i$, d, $\vec{v}$ and $\phi(\cdot)$ introduced in Sect. 2.4.2. A vector $\vec{v} \in \mathbb{Z}^d$ is said to be a relation if $\phi(\vec{v}) = [\mathcal{O}_K]$, i.e. $\vec{v}$ maps to the identity element of $\mathcal{CL}(\mathcal{O}_K)$. Let the set L be chosen such that the set of ideal classes $[\mathfrak{l}_i]$, $1 \leq i \leq d$, generates the group $\mathcal{CL}(\mathcal{O}_K)$. Following a class group computation algorithm of Jacobson [57], we compute a lattice $\Lambda = \ker \phi \subset \mathbb{Z}^d$ of relations among the ideal classes $[\mathfrak{l}_i]$, $1 \leq i \leq d$, so that $\mathcal{CL}(\mathcal{O}_K) \cong \mathbb{Z}^d / \Lambda$. Then we choose a minimal generating set of ideals $\mathfrak{l}_i$, $i \in J$, for a subset of indices $J \subset \{1, ..., d\}$, and find the orders $m_i = \mathrm{ord}[\mathfrak{l}_i]$, $i \in J$.

Now, using the Lenstra-Lenstra-Lovasz (LLL) lattice basis reduction algorithm, we compute a short basis of Λ and store it in a matrix B_Λ of column vectors. All the above described steps are needed only once and therefore can be done during the parameter choice or the pre-computation phase. This allows to reject elliptic curves that require $l_{\max}$ larger than approximately $\log|\Delta|$ (see Remark 2.1) or yield long vectors in B_Λ. Practical experiments for $\lceil \log p \rceil = 224$ and $l_{\max} \approx \log|\Delta|$ show that the coordinates of vectors in B_Λ are generally less than 50.

In order to implement the random sampling from $\mathcal{CL}$, we construct a vector $\vec{u} \in \mathbb{Z}^d$ by choosing the coordinates

$$u_i \leftarrow \begin{cases} \xleftarrow{R} \{0, 1, \ldots, m_i - 1\}, & i \in J; \\ \leftarrow 0, & i \notin J. \end{cases}$$

Now $\phi(\vec{u})$ is a random element of $\mathcal{CL}$, as the uniform distributions on the cyclic subgroups $\langle[\mathfrak{l}_i]\rangle$, $i \in J$, give the uniform distribution on $\mathcal{CL}$.

Some of the coordinates of $\vec{u}$ will be large. For instance, if the class group is generated by an ideal $\mathfrak{l}_i$, then u_i is a random number between 0 and $\#\mathcal{CL}-1$. The following optimisation steps are aimed at computing an equivalent vector $\vec{v} \equiv \vec{u} \pmod{\Lambda}$ which is faster than $\vec{u}$ in terms of its action on $\mathcal{ELL}$.

We first find a lattice vector $\vec{b} \in \Lambda$ close to $\vec{u}$ and set $\vec{w} = \vec{u} - \vec{b}$. The vector $\vec{b}$ can be found by Babai rounding [2] as $\vec{b} = B_\Lambda \lfloor B_\Lambda^{-1}\vec{u} \rceil$, where $\lfloor\cdot\rceil$ is the coordinate-wise floor function, and $\vec{u}$ is a column vector, or by any other algorithm for the closest lattice vector problem.

To further optimize $\vec{w}$, we will need a d-dimensional row vector $\vec{t} = (t_1, \ldots, t_d)$, where each coordinate t_i is the average time used to compute the action $[\mathfrak{l}_i] * E$. The row vector $\vec{t}$ can be obtained experimentally during the parameter choice phase. Now for a column vector $\vec{v} \in \mathbb{Z}^d$, the approximate time needed to compute the action $\phi(\vec{v}) * E$ is $\vec{t}\,|\,\vec{v}\,|$, where $|\cdot|$ is the coordinate-wise absolute value function. We thus need to solve the following (mixed) integer linear program (ILP):

$$\begin{aligned} &\text{minimize} \quad \vec{t}\,|\,\vec{v}\,| \\ &\text{subject to} \quad \vec{v} = \vec{w} + B_\Lambda \vec{k}, \quad \text{int } \vec{k}. \end{aligned} \tag{2.12}$$

The problem (2.12) can be solved by various ILP algorithms, e.g. the branch and bound algorithm, the simplex algorithm or a primitive search. Note that we do not require finding the optimal solution, as it can take a long time. Even a quick run of an ILP algorithm allows to significantly improve the value of the objective function in (2.12). Moreover, in some applications, for instance when $\vec{v}$ is the private key in the encryption scheme $\mathcal{PE}$ (Fig. 3.2), it is beneficial to spend more time on the optimization of $\vec{v}$ during the pre-computation phase.

The random sampling from $\mathcal{CL}$ proposed in this section requires the pre-computation of an ideal class group structure and therefore cannot be used with large class groups. However the group size threshold keeps increasing due to the development of computational resources and algorithms. The up-to-date class group computation record reported by Biasse [11] employs a 366-bit discriminant. This is enough for achieving the 112-bit security level, as discussed later in Sect. 2.7.

2.6.2 Pseudo-Random Sampling from Large $\mathcal{CL}$

We show how to implement the sampling operation without prior knowledge of the class group structure.

An algorithm for random sampling proposed by Srinivasan [103] outputs an ideal of a large norm. To further factor this ideal over the smooth factor base (2.2), techniques from index calculus algorithms for imaginary quadratic fields could be used as it is done by Galbraith et al. [45]. However this approach would have exponential running time and still yield an ideal representation which is slow in

terms of its action on $\mathcal{ELL}_{p,n}$, as compared to optimized representations discussed in Sect. 2.6.1.

We propose to use a non-uniform probability distribution S on the set of elements of $\mathcal{CL}$ instead of the uniform distribution R. Note that in order to complete the proofs of Theorems 2.1 and 2.2 we have to additionally assume that R and S are computationally indistinguishable. Several authors, including Galbraith [44], Jao et al. [61] and Teske [113], construct samplings from $\mathcal{CL}$ that employ ideals with small split norms, in order to emulate the random sampling from $\mathcal{CL}$. Below we propose a candidate probability distribution S that is constructed to optimize the speed of the $\mathcal{CL}$ action on $\mathcal{ELL}$. How plausible it is that S is computationally indistinguishable from R is a difficult question that requires further analysis.

The probability distribution S is constructed as follows. For a fixed $l_{\max}$ we use the notations Δ, d, $\mathfrak{l}_i$ and ϕ from Sect. 2.4.2. Let also $h \approx 0.46(-\Delta)^{1/2}$ be an approximation for the class number $\#\mathcal{CL}(\mathcal{O}_\Delta)$. We then choose a set $V \subset \mathbb{Z}^d$ such that $\#V = ch$ for a small $c > 1$, and the random sampling from V is easy to implement. For instance, if V is the set of vectors inside a d-dimensional box

$$V = \{\vec{v}\colon\ -\hat{v}_i \le v_i \le \hat{v}_i,\ 1 \le i \le d\} \tag{2.13}$$

defined by non-negative integers $\hat{v}_i$, $1 \le i \le d$, then the random sampling from V can be achieved through d random samplings of the coordinates. We define S to be the probability distribution on $\mathcal{CL}$ induced by the uniform probability distribution on V and the map (2.4).

To construct the box V we start with a d-cube. Since for smaller primes l_i the action $[\mathfrak{l}_i] * E$ can be computed faster, we stretch the box V along the faster dimensions and squeeze it along the slower ones, so that the average time used for the computation along the i-th axis is the same for all the dimensions $1 \le i \le d$. The values $\hat{v}_i$ in (2.13) can be computed using the timing vector $\vec{t}$ (see Sect. 2.6.1). This approach has an advantage against timing side-channel attacks, as the running time of the group action is almost the same for different randomly chosen vectors from V.

2.7 Security of $\mathcal{ELL}$-Based Cryptographic Schemes

In this section we discuss the plausibility of the DDHAP and the ES assumptions with respect to the cryptographic schemes based on the $\mathcal{CL}$ action on $\mathcal{ELL}_{p,n}$.

2.7.1 Plausibility of the $\mathcal{CL}$-DDHAP Assumption

The DDHAP formulated for the $\mathcal{CL}$ action on $\mathcal{ELL}_{p,n}$ ($\mathcal{CL}$-DDHAP) has not been considered in the literature. As far as we are concerned, the most efficient approach is to solve the corresponding $\mathcal{CL}$ group action inverse problem ($\mathcal{CL}$-GAIP).

Let us estimate the computational complexity of the $\mathcal{CL}$-GAIP. Galbraith et al. proposed an algorithm for constructing isogenies between elliptic curves [45]. Stages 2 and 3 of this algorithm particularly solve the $\mathcal{CL}$-GAIP, that is, find an ideal that maps a given elliptic curve to another given curve in the set $\mathcal{ELL}_{p,n}(\mathcal{O}_K)$. The algorithm is based on the Pollard's rho method [84] and requires approximately $(\pi h)^{1/2}$ hops between l_i-isogenous curves, $l_i \in L$, $h = \#\mathcal{CL}$. We have estimated the running time of one hop by (2.11), so the average running time of the algorithm equals $O(h^{1/2} \sum_{i=1}^{d} \frac{1}{d} l_i \log l_i \log p)$. After using the approximations for h, d and l_i as in Sect. 2.5 this becomes

$$O(p^{1/4}(\log^2 p) \log\log p) \tag{2.14}$$

multiplications in $\mathbb{F}_p$. We do not count the $O(p^{1/4+\epsilon})$ complexity of finding a short smooth representation of the resulting ideal.

In order to choose appropriate system parameters we use cryptographic security levels defined by the European Network of Excellence in Cryptology II (ECRYPT2) [40]. Computational complexities of 80, 96, 112 and 128 bits are assumed to be infeasible during the next 4, 10, 20 and 30 years, respectively. The size of p is chosen such that the number of bits in (2.14) equals the corresponding security level recommendation. The resulting values of $\log p$ are listed in Table 2.1.

2.7.2 Solving the $\mathcal{CL}$-GAIP with a Quantum Computer

Quantum computers allow to solve certain computational problems with a significantly greater efficiency than classical computers. An intriguing question is whether the $\mathcal{CL}$-GAIP can be efficiently solved on a quantum computer.

Since the problem involves the action of an ideal class group $\mathcal{CL}$ of an imaginary quadratic field, we will firstly review current advances in the computations in $\mathcal{CL}$. In the classical computation case, sub-exponential algorithms have been proposed for computing the structure of $\mathcal{CL}$ and solving the discrete logarithm problem (DLP) in a cyclic subgroup of $\mathcal{CL}$ (see, for example, Jacobson [57, 58]). Note however that these results do not imply sub-exponential complexity for the $\mathcal{CL}$-GAIP, which is still exponential-time (2.14).

In the quantum computation case, a polynomial-time algorithm for the structure of $\mathcal{CL}$ has been described by Hallgren [53]. The ability of quantum computer to solve the hidden subgroup problem is employed for computing the lattice of relations between generators of $\mathcal{CL}$. Schmidt [93] has carefully described an implementation of Shor's algorithm for solving the DLP in a cyclic subgroup of $\mathcal{CL}$ and estimated the necessary quantum register size.

These classical and quantum computation results for problems in the ideal class group do not apply to the $\mathcal{CL}$-GAIP. Indeed, the $\mathcal{CL}$-GAIP is defined for elements of the set $\mathcal{ELL}_{p,n}$ that the group $\mathcal{CL}$ acts on. One is given the j-invariants of two elliptic curves $E_x, E_y \in \mathcal{ELL}_{p,n}$, and the problem is to find an ideal $\mathfrak{r}$ such that $E_y = \mathfrak{r} * E_x$. In order to reduce the $\mathcal{CL}$-GAIP to a similar problem over $\mathbb{C}$ and look

at the relationship between corresponding complex lattices, one has to lift these elliptic curves to $\mathbb{C}$, namely to solve the following problem:

Problem 2.3 (Coherent Lifting Problem)**.** *For given ordinary elliptic curves $E_x/\mathbb{F}_p$, $E_y/\mathbb{F}_p$ with $\mathrm{End}(E_x) \cong \mathrm{End}(E_y) \cong \mathcal{O}_K$, find complex lattices Λ_x, Λ_y such that there is a prime $\mathfrak{P} \mid p$ of the Hilbert class field H of K for which*

$$\begin{aligned} E_{\Lambda_x} \pmod{\mathfrak{P}} &\cong E_x\,, \\ E_{\Lambda_y} \pmod{\mathfrak{P}} &\cong E_y\,. \end{aligned}$$

One can solve the coherent lifting problem by constructing H through the computation of the class polynomial $H_\Delta = \prod_{i=1}^{h}(X - j(\tau_i))$, where the complex numbers τ_i are obtained from the reduced representatives of the elements of $\mathcal{CL}$. One then finds a prime $\mathfrak{P}$ of H and reduces the values $j(\tau_i)$ modulo $\mathfrak{P}$. Since the degree of H_Δ is $h = \#\mathcal{CL}$, this approach requires time and space exponential in $\log h$. In fact, the best way to solve the coherent lifting problem seems to be to solve the $\mathcal{CL}$-GAIP first.

Now we will try to apply Shor's DLP algorithm [98] directly to the elements of $\mathcal{ELL}_{p,n}$. The original algorithm relies on the following idea. Let $y = x^r$ in a finite cyclic group, so that the DLP asks to compute r knowing x and y. A function $f(a,b) = x^a y^b$ has the value $(r,-1)$ as its period, since $f(a,b) = f(a+r, b-1)$. When $f(a,b)$ is implemented in quantum gates, one can efficiently find the period by means of the quantum Fourier transform, thus obtaining r. Quantum computers use reversible computation, and implementing a deterministic function on a quantum computer reversibly requires as much space as it does time. So it is essential for implementing the Shor's algorithm that the periodic function is polynomial-time. For solving the $\mathcal{CL}$-GAIP we try to construct a function $\hat{f}(\mathfrak{a}, \mathfrak{b})$ similar to $f(a,b)$, that takes imaginary quadratic ideals $\mathfrak{a}$ and $\mathfrak{b}$. Even though it is possible to compute $\mathfrak{a} * E_x$ and $\mathfrak{b} * E_y$, we do not know of any polynomial-time composition operation for the two obtained elliptic curves that would be suitable for the purpose. We leave it as an open question to find a polynomial-time periodic function on $\mathcal{ELL}_{p,n}$ with the period dependent on $\mathfrak{r}$, such that $E_y = \mathfrak{r} * E_x$.

It has been recently proposed to use quantum computers for solving the hidden shift problem. The problem asks, for a given finite group G, a finite set X and maps $f, g\colon G \to X$ such that $g(a) = f(a+r)$ for all $a \in G$ and a fixed shift $r \in G$, to find r. When applied to the $\mathcal{CL}$-GAIP, these functions can be defined as $g(\mathfrak{a}) = \mathfrak{a} * E_y$ and $f(\mathfrak{a}) = \mathfrak{a} * E_x$. Then $\mathfrak{r}$ is a (multiplicative) shift because $f(\mathfrak{a}\mathfrak{r}) = \mathfrak{a}\mathfrak{r} * E_x = \mathfrak{a} * E_y = g(\mathfrak{a})$. However polynomial-time quantum algorithms for the hidden shift problem have been described only for some special types of functions, namely the Legendre symbol [114] and several classes of bent functions, a type of boolean functions [91].

2.7.3 Plausibility of the ES Assumption

The ES assumption about a hash function family $\mathcal{H} = \{H_k \colon k \in K\}$ concerns the hash function's ability to extract entropy from a "partially random" source. In the $\mathcal{ELL}$-based $\mathcal{PE}$ scheme the hash function $H_k(u)$ is applied to the elliptic curve u represented by its j-invariant. Since there is $\log p$ bits in the representation of a j-invariant as an element of $\mathbb{F}_p$, but only $h \approx p^{1/2}$ elliptic curve isomorphism classes exist in the $\mathcal{ELL}_{p,n}$, the entropy of the random variable u is approximately $\frac{1}{2}\log p$ bits. Thus, on input u, the hash function H_k should output an almost uniformly distributed string of $\frac{1}{2}\log p$ bits. A similar problem appears, for example, when the Diffie-Hellman key agreement protocol is implemented in, say, a 160-bit multiplicative subgroup of a 1024-bit finite field. The shared secret output by the key agreement protocol needs to be transformed into a secret keying material. A function that implements this transformation is called a key derivation function. The National Institute of Standards and Technology (NIST) defines two approved key derivation functions [81]. The length of the output keying material in these functions is adjustable. The NIST key derivation functions are based on hash algorithms approved by NIST, namely the SHA-1 function and the SHA-2 family of hash functions. The applicability for key derivation is also a requirement for the upcoming SHA-3 family of hash functions. These facts suggest that it is indeed possible to construct an ES hash function family suitable for the $\mathcal{ELL}$-based $\mathcal{PE}$ scheme.

2.8 Numerical Experiments

In this section we report about a trial implementation of the arithmetic used in the proposed cryptographic schemes[3].

We have implemented the $\mathcal{CL}$ group action on $\mathcal{ELL}_{p,n}$ using the computer algebra system PARI/GP 2.4.3 created by Cohen, Belabas et al. The exponentiation in $\mathbb{F}_p[x]/f(x)$ for degrees of $f(x)$ higher than approximately 70 is more efficiently realized in the Number Theory Library (NTL) 5.5.2 by Shoup, so we make external calls to NTL when it is appropriate. PARI and NTL libraries were compiled with GNU Multiple Precision Arithmetic Library (GMP) 5.0.0. A database of Atkin modular polynomials of levels 3–499 was downloaded from the web site of the Elliptic Curves and Higher Dimensional Analogues (ECHIDNA) project by Kohel. For the class group structure computation we used the `quadratic_order` class implemented by Jacobson in the LiDIA 2.2.0 library.

The largest discriminant for which we could compute the class group structure using LiDIA was 226 bits long. This took approximately 3 hours and the process occupied 3 gigabytes of memory space. This size of discriminant corresponds to 75 bits of security and we have included it in our results. The class group structure

[3]The source code of the implementation is available at the author's personal web page, currently at `http://www.item.ntnu.no/people/personalpages/phd/anton/software`.

Security (bits)	$\lceil \log p \rceil$ (bits)	Time (seconds)
75	224	19
80	244	21
96	304	56
112	364	90
128	428	229

Table 2.1: Average running time of one $\mathcal{CL}$ action on $\mathcal{ELL}_{p,n}$.

was employed in the random sampling as described in Sect. 2.6.1. In the top row of Table 2.1, Time is the average time used for one random sampling of $\vec{v}$ followed by an optimization of $\vec{v}$ that lasts about 2 seconds and then the action $\phi(\vec{v}) * E$. Longer optimisation runs generally yield better group action times, which is relevant when the optimised vector can be precomputed off-line.

Table 2.1 shows the average time for one $\mathcal{CL}$ action on $\mathcal{ELL}_{p,n}$. For security levels 80–128, the Time column is the average time for a sampling of $\vec{v}$ from V, as described in Sect. 2.6.2, followed by the action $\phi(\vec{v}) * E$.

The system parameters which were used for time measurements in Table 2.1 are listed in Appendix 2.A.

We stress that the provided time estimations are valid for one class group action. For the $\mathcal{PE}$ scheme, an encryption requires two group actions that can be computed in parallel. Since modern processors often have several processing cores the "wall clock" encryption time can be treated as one group action time. The decryption in the $\mathcal{PE}$ scheme takes one group action. In the three proposed $\mathcal{KA}$ protocols each party has to perform two consecutive group actions. However the first action can be precomputed before the protocol starts.

Timings in Table 2.1 were obtained on a Ubuntu Linux 9.04 system with Intel Core i7 920 processor clocked at 3.6 GHz. The implementation is single-threaded, so only one of the four processor cores was used at a time. Note that whereas PARI, NTL and GMP are fast computation oriented libraries, a customized implementation of the arithmetic may result in better speeds. For instance the reduction modulo p can be implemented faster for $p = 2^n \pm a$ for small a, as compared to the generic long division algorithm used in GMP. Also the polynomial multiplication and division operations can be efficiently parallelized [12].

2.9 Concluding Remarks

Let us compare the proposed cryptographic schemes with the ones based on the ECDLP, namely the elliptic curve Diffie-Hellman key agreement scheme, the elliptic curve digital signature algorithm and others. For an elliptic curve over a field $\mathbb{F}_q$, the ECDLP is usually considered in a cyclic subgroup of order approximately q.

For the $\mathcal{CL}$-GAIP, the cardinality of the set $\mathcal{ELL}_{p,n}$ is the class number h, and the elliptic curves are defined over $\mathbb{F}_p$, where $p \approx h^2$. We shall consider the ECDLP and the $\mathcal{CL}$-GAIP of similar sizes q and h, respectively. First of all, we note that the system parameters for an ECDLP-based cryptographic scheme can be generated in polynomial in $\log q$ time, whereas a $\mathcal{CL}$-GAIP-based cryptographic scheme requires sub-exponential in $\log h$ time for computing the class group structure. The second aspect is the cryptosystem running time. The average running time of one scalar multiplication on an elliptic curve is $10 \log q$ [55], or $O(\log q)$ multiplications in $\mathbb{F}_q$. For the $\mathcal{CL}$ action on $\mathcal{ELL}$, the average running time is $O(\log^{3.7} h)$ multiplications in $\mathbb{F}_p$, which is much slower than for the ECDLP-based schemes. The third aspect we will consider is the computational complexity of the problems. The ECDLP complexity for a cryptographically strong elliptic curve is widely believed to be $O(q^{1/2} \log q)$ field operations. The computational complexity of the $\mathcal{CL}$-GAIP is $O(h^{1/2}(\log^2 h) \log \log h)$ multiplications in $\mathbb{F}_p$, i.e. the $\mathcal{CL}$-GAIP has higher complexity than the ECDLP.

It is not yet clear whether the $\mathcal{CL}$-GAIP can be efficiently solved on a quantum computer. Arguments against the applicability of some currently known quantum algorithms have been provided in Sect. 2.7.2. In case a quantum attack is discovered later, the proposed cryptographic schemes would seemingly become of theoretical interest only.

The encryption scheme and the key agreement protocols proposed in this paper use random sampling from the ideal class group $\mathcal{CL}$. Since the implementation of the $\mathcal{CL}$ action on $\mathcal{ELL}_{p,n}$ employs short smooth ideal representations, efficient random sampling is only possible for class groups with known structure. This provides security levels of up to 112 bits, as of 2009 class group computation records. Higher security levels are only achievable with a pseudo-random sampling which has to offer good randomness characteristics. It should be also noted that when the class group structure is not pre-computed, the fact that all elements of $\mathcal{CL}$ can be represented and used in the cryptographic scheme depends on the GRH.

Acknowledgements

The author would like to thank Prof. Alexander Rostovtsev, Prof. Alexei Rudakov and Prof. Stig F. Mjølsnes for their valuable support during the period of work on this topic. Very useful comments were also received from the Advances in Mathematics of Communications journal's anonymous reviewers. The author is grateful to Dr. Steven Galbraith for his feedback on this paper and for suggestion of the coherent lifting problem.

2.A System Parameters Used in Time Measurements

We use the notation param_s to refer to the s-bit security level specified in Table 2.1. The following parameters are listed in Table 2.2: p is the prime field characteristic; E is an elliptic curve from the set $\mathcal{ELL}_{p,n}$, where $n = p + 1 - t$; t is the Frobenius trace for the curves in $\mathcal{ELL}_{p,n}$; h is the class number (calculated only for the 75-bit security); $l_{\max}$ is the maximal isogeny degree used in the implementation; $\{(l_i, \hat{v}_i)\}$ is the list of tuples consisting of an isogeny degree l_i and the corresponding hop limit $\hat{v}_i$, as in (2.13) (listed only for the 128-bit security). The elliptic curves were chosen to have fundamental Frobenius discriminants $\Delta_\pi = t^2 - 4p$ in order to ensure that their endomorphism rings are maximal imaginary quadratic orders.

2.B More Schemes Based on Group Action

In addition to the cryptographic schemes proposed in Section 2.2 and schemes of other authors listed in Section 1.3, we propose more constructions based on the isogeny problem.

Throughout this section we let G be a finite abelian group acting on a set X. In the context of the isogeny problem we have that $G = \mathcal{CL}(\mathcal{O}_K)$ and $X = \mathcal{ELL}_{p,n}(\mathcal{O}_K)$ for suitably chosen elliptic curves over $\mathbb{F}_p$ having n rational points and the endomorphism ring $\mathcal{O}_K$. In all public-key protocols the private key sk is chosen to be a random element of G, and the corresponding public key is computed as $\text{pk} \leftarrow \text{sk} * x$, where $x \in X$ is a part of the common system parameters.

The digital signature scheme $\mathcal{DS}$ pictured in Fig. 2.6 is based on the idea of Fiat and Shamir [42] to obtain a signature scheme from an interactive authentication protocol by replacing the verifier's challenge with a hash value (this idea was later used by Schnorr to obtain a DLP-based signature scheme [94]). The scheme $\mathcal{DS}$ is constructed from the Σ-protocol of Couveignes shown in Fig. 1.2. It is assumed that a security parameter $t \in \mathbb{N}$ and a hash function $H : (G * x)^t \times M \to \{0,1\}^t$ are publicly known.

A signature generated by Algorithm 2.1 is always accepted by Algorithm 2.2 because, for each i, the following holds:

$$y_i = \begin{cases} c_i * \text{pk} = a_i * \text{pk}, & \text{when } b_i = 0; \\ c_i * x = (a_i \,\text{sk}) * x = a_i * (\text{sk} * x) = a_i * \text{pk}, & \text{when } b_i = 1. \end{cases}$$

To forge a $\mathcal{DS}$ signature without knowledge of the private key, a forger has to find a hash input $(y_1, \ldots, y_t, m)$ that yields a hash value $(b_1, \ldots, b_t)$ for which he knows values c_i satisfying: if $b_i = 0$ then $y_i = c_i * \text{pk}$, else $y_i = c_i * x$. To achieve this, the forger can choose values $(b_1, \ldots, b_t, c_1, \ldots, c_t)$, compute $(y_1, \ldots, y_t)$, and verify the condition $H(y_1, \ldots, y_t, m) = (b_1, \ldots, b_t)$. If the condition does not hold,

Parameter	Value
p_{75}	$2^{224} - 63$
E_{75}	$y^2 = x^3 + x + 5217$
t_{75}	706283819635943803784155145600859
h_{75}	367259891656247020496958525412808 1
$l_{\max,75}$	163
p_{80}	$2^{243} + 59$
E_{80}	$y^2 = x^3 + x + 20321$
t_{80}	697564694854065258432585807924187581
$l_{\max,80}$	263
p_{96}	$2^{303} + 101$
E_{96}	$y^2 = x^3 + x + 3704$
t_{96}	57847001694414882341709570340537328669512200 27
$l_{\max,96}$	443
p_{112}	$2^{363} + 309$
E_{112}	$y^2 = x^3 + x + 1919$
t_{112}	$-$59344974584391105805850406935841437299180143301389290 37
$l_{\max,112}$	487
p_{128}	$2^{427} + 69$
E_{128}	$y^2 = x^3 + x + 1025$
t_{128}	14933846636205862089595788320503564108095537034421076949568186265
$l_{\max,128}$	499
$\{(l_i, \hat{v}_i)\}_{128}$	$(3, 2669)$, $(7, 1094)$, $(13, 475)$, $(17, 317)$, $(19, 269)$, $(29, 121)$, $(31, 107)$, $(37, 74)$, $(41, 64)$, $(53, 41)$, $(59, 37)$, $(61, 35)$, $(71, 25)$, $(83, 20)$, $(89, 19)$, $(97, 17)$, $(101, 17)$, $(103, 16)$, $(107, 14)$, $(127, 13)$, $(131, 9)$, $(137, 9)$, $(149, 8)$, $(167, 7)$, $(181, 6)$, $(197, 6)$, $(199, 6)$, $(223, 5)$, $(229, 5)$, $(239, 5)$, $(307, 2)$, $(311, 3)$, $(313, 2)$, $(317, 2)$, $(331, 2)$, $(337, 2)$, $(347, 2)$, $(367, 2)$, $(379, 2)$, $(383, 2)$, $(389, 1)$, $(409, 2)$, $(421, 1)$, $(449, 1)$, $(457, 1)$, $(461, 1)$, $(499, 1)$

Table 2.2: System parameters used in the numerical experiments.

Alg. 2.1 Signature generation

Input: sk, pk, m
  for $i = 1$ to t **do**
    $a_i \xleftarrow{R} G$
    $y_i \leftarrow a_i * \mathrm{pk}$
  end for
  $(b_1, \ldots, b_t) \leftarrow H(y_1, \ldots, y_t, m)$
  for $i = 1$ to t **do**
    $c_i \leftarrow a_i(\mathrm{sk})^{b_i}$
  end for
  $s \leftarrow (b_1, \ldots, b_t, c_1, \ldots, c_t)$
Output: s

Alg. 2.2 Signature verification

Input: pk, m, s
  $(b_1, \ldots, b_t, c_1, \ldots, c_t) \leftarrow s$
  for $i = 1$ to t **do**
    if $b_i = 0$ **then**
      $y_i \leftarrow c_i * \mathrm{pk}$
    else
      $y_i \leftarrow c_i * x$
    end if
  end for
  $H(y_1, \ldots, y_t, m) \stackrel{?}{=} (b_1, \ldots, b_t)$
Output: 1

Figure 2.6: Digital signature scheme $\mathcal{DS}$.

the forger chooses another values $(b_1, \ldots, b_t, c_1, \ldots, c_t)$ or a message m, and tries again. The hash function $H()$ should be indistinguishable from a random function (i.e. entropy smoothing). Due to this requirement, forger's success probability at every trial is approximately $1/2^t$. The forger performs a sequence of Bernoulli trials, and the number of necessary trials is described by the geometric distribution with the mean value 2^t. Hence the parameter t should be larger than or equal to the security level s.

Another requirement on the function $H()$ is that it should be one-way with respect to m. Otherwise the forger can choose values $(b_1, \ldots, b_t, c_1, \ldots, c_t)$, compute $(y_1, \ldots, y_t)$, and solve $H(y_1, \ldots, y_t, m) = (b_1, \ldots, b_t)$ to find m. There is no need for the function $H()$ to be one-way with respect to $(y_1, \ldots, y_t)$ because, given $(y_1, \ldots, y_t)$, it is hard to compute the group action inverses $(c_1, \ldots, c_t)$.

When speaking about the $\mathcal{DS}$ implementation using the $\mathcal{CL}$ action, there is one operation that has not been discussed before: the multiplication in $\mathcal{CL}$. When $b_i = 1$, the value c_i is a product of a random element a_i and the private key sk. The value c_i is a part of the signature, and hence it should not leak any information about sk. Elements of $\mathcal{CL}$ are represented by integer vectors, as in (2.4). We note that it is unsecure to just add two vectors and write the result into c_i. However, when $\mathcal{CL}$ is cyclic with a small-norm generator of a known order h, elements of $\mathcal{CL}$ can be represented by integers modulo h. This ensures that no information about sk leaks from the representation of c_i.

Finally, we note that one signature generation or verification requires t group actions. Because of this, the $\mathcal{DS}$ scheme implemented on isogenous elliptic curves is currently slow.

A one-way authenticated key agreement protocol $\mathcal{KA}4$ (see Fig. 2.7), based on group action, can be obtained as a generalization of ElGamal's one-pass protocol,

$\mathcal{A}$		$\mathcal{B}$
Input: pk_B		**Input:** sk_B
$a \xleftarrow{R} G$		
$y \leftarrow a * x$		
	$\xrightarrow{y}$	
$k_A \leftarrow a * \mathrm{pk}_B$		$k_B \leftarrow \mathrm{sk}_B * y$
Output: k_A		**Output:** k_B

Figure 2.7: One-pass authenticated key agreement protocol $\mathcal{KA}4$.

Alg. 2.3 Key generation
Input: –
$\mathrm{ek} \xleftarrow{R} G$
$\mathrm{dk} \leftarrow \mathrm{ek}^{-1}$
Output: ek, dk

Alg. 2.4 Encryption
Input: $\mathrm{ek} \in G$, $m \in X$
$c \leftarrow \mathrm{ek} * m$
Output: c

Alg. 2.5 Decryption
Input: $\mathrm{dk} \in G$, $c \in X$
$m \leftarrow \mathrm{dk} * c$
Output: m

Figure 2.8: Encryption scheme $\mathcal{ES}1$.

also called a half-certified Diffie-Hellman protocol [78, Protocol 12.51]. Here entity A a-priory possesses B's authentic public key pk_B. The key authentication is only one-way because B has no assurance in who he shares k_B with.

A secret key encryption scheme $\mathcal{ES}1$ (see Fig. 2.8) on group action can be obtained by generalizing the Pohlig-Hellman encryption scheme [82]. In this scheme the decryption key dk can be easily computed from the encryption key ek, and vice-versa.

It is sometimes not convenient to have a message space X, as it is in the scheme $\mathcal{ES}1$. For instance, a set $X = \mathcal{ELL}_{p,n}(\mathcal{O}_K)$ is represented by approximately $p^{1/2}$ bit strings sparsely distributed among the bit strings of length $\lceil \log(p) \rceil$. To address this problem we construct a "hashed" variant of $\mathcal{ES}1$, called $\mathcal{ES}2$ (see Fig. 2.9). The key generation for $\mathcal{ES}2$ is done by Algorithm 2.3. The message space in $\mathcal{ES}2$ is the set of bit strings of length $w \leq \log(\#G * x)$. The scheme employs a hash function $H : G * x \mapsto \{0,1\}^w$ that is entropy smoothing. Note that a hybrid encryption scheme similar to $\mathcal{ES}2$ would make little sense because one can use a bare symmetric encryption instead.

It is also possible to send confidential messages without any pre-shared secret keys, given that the channel is authentic. The protocol $\mathcal{TP}$ pictured in Fig. 2.10 is a generalization of the three-pass protocol proposed by Shamir[4].

The protocol $\mathcal{TP}$ can be easily transformed into a "hashed" version with the

[4]According to Massey [74, §IV.C], this three-pass protocol was proposed by A. Shamir in an unpublished work. The protocol was patented by Massey and Omura [75].

Alg. 2.6 Encryption

Input: $m \in \{0,1\}^w$, ek

$a \xleftarrow{R} G$

$y \leftarrow a * x$

$k \leftarrow H(y)$

$e \leftarrow m \oplus k$

$c \leftarrow \mathrm{ek} * y$

Output: (e, c)

Alg. 2.7 Decryption

Input: (e, c), dk

$y \leftarrow \mathrm{dk} * c$

$k \leftarrow H(y)$

$m \leftarrow e \oplus k$

Output: m

Figure 2.9: Encryption scheme $\mathcal{ES}2$.

$\mathcal{A}$		$\mathcal{B}$
Input: $m \in X$		**Input:** $-$
$a \xleftarrow{R} G$		$b \xleftarrow{R} G$
$m_1 \leftarrow a * m$		
	$\xrightarrow{m_1}$	
		$m_2 \leftarrow b * m_1$
	$\xleftarrow{m_2}$	
$m_3 \leftarrow a^{-1} * m_2$		
	$\xrightarrow{m_3}$	
		$m \leftarrow b^{-1} * m_3$
Output: $-$		**Output:** m

Figure 2.10: Three-pass message transfer protocol $\mathcal{TP}$.

$\mathcal{P}$		$\mathcal{V}$
Input: $v \in \{0,1\}^w$, sk_P		**Input:** pk_P
$a \xleftarrow{R} G$		
$y \leftarrow a * x$		
$z \leftarrow \mathrm{sk}_P * y$		
$c \leftarrow H(z) \oplus v$		
	Commit: $y, c \rightarrow$	
	Reveal: $a \rightarrow$	
		$a * x \stackrel{?}{=} y$
		$z \leftarrow a * \mathrm{pk}_P$
		$v \leftarrow H(z) \oplus c$
Output: $-$		**Output:** v

Figure 2.11: Commitment scheme $\mathcal{CS}$.

message space $\{0,1\}^w$ or a "hybrid encryption" version with the message space $\{0,1\}^*$. A possible implementation is: Alice chooses a random hash preimage (or key material) $k \in G * x$ and securely transmits it to Bob using the protocol $\mathcal{TP}$. In her third protocol message, Alice also transmits $m \oplus H(k)$ (or m encrypted with a symmetric algorithm using a key derived from k). Bob then uses k to recover (or decrypt) the message m.

A commitment scheme $\mathcal{CS}$ (see Fig. 2.11) is constructed using the ideas from the ElGamal encryption. As before, the entity P possesses a private key $\mathrm{sk}_P \in G$ and a public key $\mathrm{pk}_P = \mathrm{sk}_P * x$. The entity P first commits to a value v and then reveals.

The commitment scheme $\mathcal{CS}$ is perfectly binding. Indeed, suppose that for a commitment (y, c) there exist two reveal values a and a' that can be accepted by the verifier with two different results v and v', respectively. From the checks performed by the verifier we have that $a*x = y$ and $a'*x = y$. Hence $z = a*\mathrm{pk}_P = a*(\mathrm{sk}_P*x) = \mathrm{sk}_P * (a * x) = \mathrm{sk}_P * y$ and similarly $z' = a' * \mathrm{pk}_P = \mathrm{sk}_P * (a' * x) = \mathrm{sk}_P * y$. We have shown that $z = z'$ and hence $v = v'$, a contradiction.

The scheme $\mathcal{CS}$ is computationally hiding: an attacker can compute v from x, pk_P, y and c by solving the Diffie-Hellman group action problem (x, y, pk_P) which gives the value z. The attacker then computes $v = H(z) \oplus c$.

Paper II

Chapter 3 is published in

Anton Stolbunov

Reductionist Security Arguments for Public-Key Cryptographic Schemes Based on Group Action

"Norwegian Information Security Conference (NISK 2009)" (ed. Stig F. Mjølsnes), Tapir Akademisk Forlag, (2009), 97–109

Chapter 3

Reductionist Security Arguments for Public-Key Cryptographic Schemes Based on Group Action

ABSTRACT. We provide reductionist security arguments for a key agreement protocol $\mathcal{KA}$, which is the Diffie-Hellman key agreement protocol generalized to the context of a group action on a set, and for a public-key encryption scheme $\mathcal{PE}$, which is the "hashed" ElGamal scheme generalized for a group action on a set. For the $\mathcal{KA}$ protocol we use the notion of session key security in the authenticated links model, proposed by Canetti and Krawczyk. For the $\mathcal{PE}$ scheme we use a version of the semantic security notion proposed by Goldwasser and Micali. We prove that the security of the $\mathcal{KA}$ protocol and the $\mathcal{PE}$ scheme is based on the decisional Diffie-Hellman group action problem, defined later in this paper. The $\mathcal{PE}$ scheme security also depends on the entropy smoothing property of the hash function family used in the scheme.

3.1 Introduction

The formulation of public-key cryptographic schemes in terms of a semigroup action on a set allows to use new mathematical structures for implementing the schemes. Several implementations have been proposed that use (semi)group actions different from the exponentiation in a cyclic group [67, 77, 79, 89]. A common feature of these cryptographic schemes is that their security goes beyond the conventional discrete logarithm problem and is based on some new computational problems. The new problems might prove themselves to be asymptotically harder than those used nowadays. For instance, some known discrete logarithm solvers are difficult to adapt for the computational problems discussed in Section 3.4.

To identify the computational problems constituting the basis for security of a cryptographic scheme, one should construct a reduction from a particular computa-

tional problem(s) to an attack against the cryptographic scheme. This reduction is often called a security proof, however Koblitz and Menezes have recently advocated a more accurate name: a reductionist security argument [69, 70].

Our paper provides a reductionist security argument for a key agreement protocol $\mathcal{KA}$ based on a group action on a set. For this we firstly define the protocol $\mathcal{KA}$ and the computational problem DDHAP (decisional Diffie-Hellman group action problem, defined in Section 3.4) that is used in the reduction. We then proceed with defining a security notion, i.e. a very general attack, and show that this attack, if successful, can be used to solve the DDHA problem. We thus conclude that attacking the $\mathcal{KA}$ protocol is not easier than solving the DDHA problem.

Formalizing the security of cryptographic protocols turned out to be more complicated than that of cryptographic primitives. Among the main reasons for this is the interactive nature of the protocols in the presence of two or more participants. This opens the possibility for many different types of attacks that use concurrent sessions, participant collusions, corrupt communication channels and so on. Bellare and Rogaway presented the first formal security notion for key agreement protocols [7]. Despite of its later extensions by various authors, the model appears to be rather difficult to work with. In our paper we will use a key agreement protocol security model proposed by Canetti and Krawczyk [21, 22]. The security in this model is formalized via the infeasibility for an adversary to distinguish between the real value of the session key and an independent random value. The adversary is modelled to have essentially the same characteristics as those specified by the Dolev-Yao threat model [37]. There exist more recent works on the the topic of key agreement protocol security, e.g. by Kudla and Paterson [72], but we consider the Canetti-Krawczyk model to be well-suited and sufficient for the scope of this paper.

Another important result of this paper is a reductionist security argument for a public-key encryption scheme $\mathcal{PE}$ based on a group action on a set. We use a version of the semantic security notion [50] called indistinguishability of encryptions in a chosen-plaintext attack (IND-CPA). The equivalence of the semantic security and the IND-CPA security notions has been shown by Goldreich [46]. We prove that the $\mathcal{PE}$ scheme is secure in the sense of IND-CPA if the DDHA problem is hard and the hash function family is entropy smoothing.

3.2 Notation

We start with the basic notation. Let G be a finite commutative semigroup. We will omit the (semi)group operation sign, writing gh for the product of elements $g, h \in G$. For a set X, a semigroup action of G on X is a map

$$\begin{aligned} G \times X &\to X \\ (g, x) &\mapsto g * x, \end{aligned}$$

Elements of G	Permutations on X
1	(3)(4)(5)(9)
2	(3 9 4 5)
3	(3 5 4 9)
4	(3 4)(5 9)

Table 3.1: Action of $G = \mathbb{Z}_5^*$ on $X = \{3, 4, 5, 9\} \subset \mathbb{Z}_{11}^*$ by exponentiation.

which satisfies the associativity property $(gh) * x = g * (h * x)$ for all $g, h \in G$ and $x \in X$. When G is a group, the group action of G on X also satisfies $e * x = x$ for the identity element $e \in G$ and all $x \in X$. An example of a group action is provided in Table 3.1. The orbit of a set element $x \in X$ is the subset $G * x = \{g * x \mid g \in G\}$. When G is a group, the orbits are equivalence classes on X [38, Proposition 4.1.2].

By $a \leftarrow b$ we denote the assignment of a value b to a variable a. By $a \xleftarrow{R} G$ we mean that a is sampled from the uniform distribution on the set of elements of G. We write $\#S$ for the number of elements in S. By $\oplus$ we denote the bitwise XOR operation.

As a general rule, we assume that all the discussed algorithms take descriptions of G and X, and a fixed element $x \in X$, as part of implied *system parameters*. By descriptions of G and X we mean the information needed to implement the operations involved in the algorithms, i.e. the random sampling from G, the action of G on X, the semigroup operation etc.

3.3 Cryptographic Schemes

Since the cryptographic schemes proposed in this section do not require inverses and the identity element in G, we let G be a finite commutative semigroup acting on a set X.

3.3.1 Key Agreement Protocol Based on Semigroup Action

We define a key agreement protocol $\mathcal{KA}$ depicted in Fig. 3.1. Here A and B are identifiers of the participants Alice and Bob, and i is a unique session identifier. In this protocol Alice is the initiator and Bob is the responder. By $\mathcal{A}$ and $\mathcal{B}$ we denote the algorithms run by Alice and Bob, respectively.

Due to the commutativity of G and the associativity of the action, the following holds:

$$k_A = a * \beta = a * (b * x) = (ab) * x = (ba) * x = b * (a * x) = b * \alpha = k_B, \qquad (3.1)$$

so $\mathcal{A}$ and $\mathcal{B}$ output the same session key.

$\mathcal{A}$		$\mathcal{B}$
Input: A, B, i		**Input:** B
$a \xleftarrow{R} G$		
$\alpha \leftarrow a * x$		
	$\xrightarrow{A, i, \alpha}$	
		$b \xleftarrow{R} G$
		$\beta \leftarrow b * x$
	$\xleftarrow{B, i, \beta}$	
$k_A \leftarrow a * \beta$		$k_B \leftarrow b * \alpha$
erase a		erase b
Output: k_A, i		**Output:** k_B, i

Figure 3.1: Key agreement protocol $\mathcal{KA}$.

If we choose G and X in a way as in Table 3.1, it becomes clear that the protocol $\mathcal{KA}$ is a generalization of the key agreement protocol proposed by Diffie and Hellman [36]. A simplified version of the $\mathcal{KA}$ protocol that did not contain session identifiers, party identifiers and "erase" statements, was proposed by Monico [79].

3.3.2 Public-Key Encryption Based on Semigroup Action

We now generalize the ElGamal public-key encryption scheme to the context of semigroup action. An approach proposed by Monico [79] requires the set X to be a group in order to mask a message $m \in X$. In contrast with this, we use a "hashed" version of the ElGamal encryption scheme, which eliminates these restrictions on X and m. For a fixed message length w, the message space is the set of bit strings $\{0,1\}^w$, and thus we can write $m \in \{0,1\}^w$. We use a hash function family $\mathcal{H} = \{H_k \colon k \in K\}$ indexed by a finite set K, such that each H_k is a function

$$H_k \colon G * x \to \{0,1\}^w.$$

We define a public-key encryption scheme $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in Fig. 3.2. Some of notations we use are: $\mathrm{sk} \in G$ is a secret key, $\mathrm{pk} \in X \times K$ is a public key and $\mathrm{ct} \in \{0,1\}^w \times X$ is a ciphertext. We also have that $x, y, z, u \in X$.

The encryption scheme $\mathcal{PE}$ is sound, that is to say, for all pairs (sk, pk) which can be output by $\mathcal{K}$ and for all $m \in \{0,1\}^w$ we have that $\mathcal{D}(\mathrm{sk}, \mathrm{pk}, \mathcal{E}(\mathrm{pk}, m)) = m$. Indeed, we can write $H_k(a * y) = H_k(\mathrm{sk} * z)$ since

$$a * y = a * (\mathrm{sk} * x) = (a\,\mathrm{sk}) * x = \mathrm{sk} * (a * x) = \mathrm{sk} * z.$$

$\mathcal{K}$: Key generation
Input: -
$\mathrm{sk} \xleftarrow{R} G$
$y \leftarrow \mathrm{sk} * x$
$k \xleftarrow{R} K$
$\mathrm{pk} \leftarrow (y, k)$
Output: sk, pk

$\mathcal{E}$: Encryption
Input: pk, m
$a \xleftarrow{R} G$
$u \leftarrow a * y$
$h \leftarrow H_k(u)$
$z \leftarrow a * x$
$c \leftarrow h \oplus m$
$\mathrm{ct} \leftarrow (c, z)$
Output: ct

$\mathcal{D}$: Decryption
Input: sk, pk, ct
$u \leftarrow \mathrm{sk} * z$
$h \leftarrow H_k(u)$
$m \leftarrow h \oplus c$
Output: m

Figure 3.2: Public-Key Encryption Scheme $\mathcal{PE}$.

3.4 Computational Problems

In this section we define computational problems that will serve as the basis for the security of the cryptographic schemes defined above.

We note that a generic semigroup action has some disadvantages, as compared to a group action, when viewed from the security perspective. Without inverses in G, an action by $a \in G$ on X might not be a permutation, and so $\alpha = a * x$ does not necessarily imply $x \in G * \alpha$. In other words, the orbits $G * x$ and $G * \alpha$ might be of different sizes. This may, for instance, reduce the session key space in the $\mathcal{KA}$ protocol. An easy example is the semigroup of the subsets of the set of two elements $\{p, q\}$ with the join composition $\cup$, acting on itself. The orbit of $\{\}$ has four elements, whereas the orbit of $\{p, q\}$ has only one element. Thus, semigroups which are not groups should be chosen carefully. On the other hand, a group action eliminates this kind of flaw, as the orbits are equivalence classes. We note that if G is a group acting on a set X, then due to the orbit-stabilizer theorem [38, Proposition 4.1.2], we have that

$$\# G * x = [G : G_x],$$

where $G_x = \{g \in G \mid g * x = x\}$ is the stabilizer of x. Since the cosets in G/G_x have the same cardinality, the random sampling $a \xleftarrow{R} G$ followed by the group action $u \leftarrow a*x$ produces the same result as the random sampling from the orbit $u \xleftarrow{R} G*x$.

For the rest of this paper we let G be a finite abelian group acting on a set X, and x a fixed element in X.

Problem 3.1. Group Action Inverse (GAI) Problem*: given a randomly chosen element $y \in G * x$, find a group element $g \in G$ such that $g * x = y$.*

Problem 3.1 is a generalization of the discrete logarithm problem. To show this, consider X to be a multiplicative group, x a generator of a cyclic subgroup and

$G = \mathbb{Z}^*_{\operatorname{ord} x}$. However there exist instances of G and X which are outside of this trivial scope[1]. In order to adapt conventional algorithms such as the Pollard rho, Pollard kangaroo, Pohlig-Hellman or index calculus for the generic GAI problem, one may try to define a group on $G * x$ which is isomorphic to G/G_x. This is achieved by choosing x to be the identity element and defining the product of any two elements $y, z \in G * x$ as

$$y \cdot z = (ab) * x, \tag{3.2}$$

where a and b are such that $y = a * x$ and $z = b * x$. But, when both a and b are not known, the multiplication (3.2) is equivalent to Problem 3.2 and is hard for some instances of G and X. Thus it is still a question how the named discrete logarithm solvers can be adapted for the GAI problem.

Problem 3.2. Computational Diffie-Hellman Group Action (CDHA) Problem*: given elements $y = a * x$ and $z = b * x$, where a and b are chosen at random from G, find $(ab) * x$.*

Problem 3.3. Decisional Diffie-Hellman Group Action (DDHA) Problem*: given a triple $(y, z, u) \in X^3$ sampled with probability $1/2$ from one of the two following probability distributions:*

- $(a * x,\ b * x,\ (ab) * x)$, *where a and b are randomly chosen from G,*
- $(a * x,\ b * x,\ c * x)$, *where a, b and c are randomly chosen from G,*

decide which distribution the triple is sampled from.

Problems 3.2 and 3.3 are generalizations of the computational Diffie-Hellman problem and the decisional Diffie-Hellman problem, respectively. Using a GAIP solver it is straightforward to construct a solver for the CDHA problem, thus the CDHA problem is not harder than the GAI problem. Similarly, the DDHA problem is not harder than the CDHA problem.

For a DDHAP distinguisher $\mathcal{S}$, its probability of returning the correct solution will be denoted by $\Pr^{\mathrm{DDH}}_{\mathcal{S}}$. $\Pr^{\mathrm{DDH}}_{\mathcal{S}}$ is a function of a security parameter $s = \log \#G{*}x$. Since the distinguisher $\mathcal{S}$ can gain a success probability of $1/2$ by returning a random solution, the advantage of $\mathcal{S}$ is defined to be

$$\operatorname{Adv}^{\mathrm{DDH}}_{\mathcal{S}} = \left| \Pr^{\mathrm{DDH}}_{\mathcal{S}} - \frac{1}{2} \right|.$$

We can now formulate the following assumption about the computational complexity of the DDHA problem:

[1] A good example is the action of the class group $\mathcal{CL}(\mathcal{O}_K)$ of an imaginary quadratic field K on the set $\mathcal{ELL}_{p,n}(\mathcal{O}_K)$ of isomorphism classes of elliptic curves over $\mathbb{F}_p$ with n points and the endomorphism ring $\mathcal{O}_K$ [89]

Assumption 3.1. DDHAP Assumption*: for any polynomial-time DDHAP distinguisher $\mathcal{S}$, the advantage* $\mathrm{Adv}_{\mathcal{S}}^{\mathrm{DDH}}$ *is a negligible*[2] *function of* s.

3.5 Reductionist Security Arguments

3.5.1 Session-Key Security of the $\mathcal{KA}$ Protocol

To model the security of a key agreement protocol we will use a notion of session-key (SK) security in the authenticated-links adversarial model (AM) proposed by Canetti and Krawczyk [21]. We refer to their paper for a formal definition of the SK security in the AM. Below we provide an outline of this security notion.

A protocol Π is modelled as a collection of interactive probabilistic polynomial-time (PPT) algorithms run by the parties. These algorithms are triggered by arriving messages. A session is an instantiation of Π run at a party. Note that there can be more parties than roles in Π, and any number of sessions can be run within each party.

The adversary $\mathcal{I}$ is a PPT algorithm that has full control over the communication links. In addition to this, $\mathcal{I}$ can:

- activate a session within some party by either sending it an action request message or a protocol message;
- corrupt a party, i.e. learn its current internal state;
- learn the current internal state of the specified session within a party;
- learn the session key output by the specified session;
- perform a test-session query (see below).

The only restriction the AM imposes on $\mathcal{I}$ is that it cannot inject or modify messages (except for messages from corrupted parties) and that any message can be delivered at most once.

The notion of SK security captures the idea that such an adversary $\mathcal{I}$ does not learn anything about the value of the key of an uncorrupted session. This is formalized via the infeasibility for $\mathcal{I}$ to distinguish between the real value of the session key and an independent random value. In particular, $\mathcal{I}$ participates in the following *SK experiment*. After some period of its regular actions described above, the adversary $\mathcal{I}$ chooses a session in which it wants to be tested, by issuing a test-session query. We then toss a coin and provide $\mathcal{I}$ with either the value of the session key for the queried session, or with a random value from the key space. $\mathcal{I}$ is then allowed to continue with the regular actions, but not allowed to expose the test

[2] A function $\mu(x)$ is negligible, if for every positive integer c there exists an $N_c > 0$ such that for all $x > N_c$, the following holds: $|\mu(x)| < 1/x^c$.

session. At the end, $\mathcal{I}$ outputs its guess. We will denote by $\Pr_{\mathcal{I}}^{\mathrm{SK}}$ the probability that $\mathcal{I}$ guesses correctly, and by

$$\mathrm{Adv}_{\mathcal{I}}^{\mathrm{SK}} = \left| \Pr_{\mathcal{I}}^{\mathrm{SK}} - \frac{1}{2} \right|$$

the $\mathcal{I}$'s advantage.

Definition 3.1. A key exchange protocol Π is called SK-secure if the following properties hold for any polynomial-time adversary $\mathcal{I}$ in the AM:

1. If two uncorrupted parties complete matching sessions then they both output the same key.
2. $\mathrm{Adv}_{\mathcal{I}}^{\mathrm{SK}}$ is a negligible function of the security parameter s.

Theorem 3.1. *If the DDHAP assumption holds for the finite abelian group G acting on the set X, then the $\mathcal{KA}$ protocol is SK-secure in the AM.*

Proof. Our proof will be a generalization of that proposed by Canetti and Krawczyk for the case of $X = \mathbb{Z}_q^*$ [21, §5.1].

It has been shown in (3.1) that two uncorrupted parties in matching sessions output the same session key, what satisfies the first requirement of Definition 3.1. To show that the second requirement holds for the $\mathcal{KA}$ as well, let us assume there is a polynomial-time adversary $\mathcal{I}$ with a non-negligible advantage ϵ. We now construct a DDHAP distinguisher $\mathcal{S}$ that employs the adversary $\mathcal{I}$ as depicted in Algorithm 3.1.

Alg. 3.1 DDHAP distinguisher $\mathcal{S}$

Input: $(y, z, u) \in X^3$

1: $r \xleftarrow{R} \{1, ..., l\}$, where l is an upper bound on the number of sessions activated by $\mathcal{I}$ in any interaction
2: invoke $\mathcal{I}$ and simulate the environment of the SK experiment in the AM, except for the r-th activated protocol session
3: upon the activation of the r-th session (say it is between Alice and Bob and has a session identifier i), let Alice send (A, i, y) to Bob, and let Bob send (B, i, z) to Alice
4: **if** the r-th session is chosen by $\mathcal{I}$ as the test session **then**
5: provide u as the answer to the test query
6: $d \leftarrow \mathcal{I}$'s output
7: **else**
8: $d \xleftarrow{R} \{0, 1\}$
9: **end if**

Output: d

Consider the case when the r-th session is not chosen by $\mathcal{I}$ as the test session. The distinguisher $\mathcal{S}$ outputs a random guess, and thus $\mathrm{Adv}_{\mathcal{S}}^{\mathrm{DDH}} = 0$. Now when the r-th session is the test session, the definition of Problem 3.3 ensures that the probability distributions observed by $\mathcal{I}$ are the same as in the SK experiment, and thus $\mathrm{Adv}_{\mathcal{S}}^{\mathrm{DDH}} = \epsilon$. Since this "lucky" case happens with $1/l$ probability, we have that in general

$$\mathrm{Adv}_{\mathcal{S}}^{\mathrm{DDH}} = \epsilon/l,$$

which is non-negligible. Since $\mathcal{S}$ is polynomial-time, we have a contradiction with the DDHAP assumption. □

3.5.2 IND-CPA Security of the $\mathcal{PE}$ Scheme

The classical goal of encryption is to preserve the privacy of messages: an adversary should not be able to learn from a ciphertext information about its plaintext beyond the length of that plaintext. This idea is captured via the notion of semantic security of an encryption scheme, which asserts that any polynomial-time adversary cannot effectively distinguish between the encryption of two messages of his choosing [50]. We will use an equivalent notion, indistinguishability of encryptions in a chosen-plaintext attack (IND-CPA) [8, §2.2].

For a public-key encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ the IND-CPA experiment is depicted in Algorithm 3.2.

Alg. 3.2 IND-CPA experiment

1: $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathcal{K}$
2: $(m_0, m_1, j) \leftarrow \mathcal{I}_1(\mathrm{pk})$
3: $d \xleftarrow{R} \{0, 1\}$
4: $\mathrm{ct} \leftarrow \mathcal{E}(\mathrm{pk}, m_d)$
5: $d' \leftarrow \mathcal{I}_2(j, \mathrm{ct})$

An adversary is viewed as a pair of algorithms $\mathcal{I} = (\mathcal{I}_1, \mathcal{I}_2)$. Algorithm $\mathcal{I}_1$ can output some state information j that is then passed to $\mathcal{I}_2$. The messages m_0 and m_1 output by $\mathcal{I}_1$ are required to have the same length. By $\mathrm{Pr}_{\mathcal{I}}^{\mathrm{IND}}$ we denote the probability that $d = d'$ in the IND-CPA experiment. The advantage of $\mathcal{I}$ is

$$\mathrm{Adv}_{\mathcal{I}}^{\mathrm{IND}} = \left| \mathrm{Pr}_{\mathcal{I}}^{\mathrm{IND}} - \frac{1}{2} \right|.$$

Definition 3.2. A public-key encryption scheme Π is said to be secure in the sense of IND-CPA if $\mathcal{I}$ being polynomial-time implies that the advantage $\mathrm{Adv}_{\mathcal{I}}^{\mathrm{IND}}$ is negligible.

In our security argument we will also use a property of a hash function family $\mathcal{H}$ to be entropy smoothing (ES). The smooth entropy denotes the number of almost

uniform random bits in a random variable [20]. The ES hash function should be able to produce almost uniformly distributed outputs by decreasing the output size, as compared to the size of the input. This is formalized via the requirement that any polynomial-time adversary cannot effectively distinguish between the values $(k, H_k(u))$ and (k, h), where $k \in K$, $u \in U$ and $h \in \{0,1\}^w$ are chosen at random, and U is the domain of the hash functions. The ES property is a reasonable assumption for modern ad hoc hash function families [99].

Definition 3.3. Let $\mathcal{H} = \{H_k \colon k \in K\}$ be an indexed family of hash functions, where each H_k is a function

$$H_k \colon U \to \{0,1\}^w$$

for a fixed w. Consider two probability spaces over $K \times \{0,1\}^w$, namely P_0 is the uniform probability space and P_1 is induced by the uniform distribution on $K \times U$ and a function

$$\begin{aligned} K \times U &\to K \times \{0,1\}^w \\ (k, u) &\mapsto (k,\ H_k(u))\,. \end{aligned}$$

$\mathcal{H}$ is said to be entropy smoothing, when the probability spaces P_0 and P_1 are computationally indistinguishable, i.e. the advantage $\mathrm{Adv}_{\mathcal{S}}^{\mathrm{ES}}$ of any polynomial-time distinguisher $\mathcal{S}$, that takes an element of $K \times \{0,1\}^w$ and outputs a bit, is negligible.

In the following theorem, the hash function's domain U is the orbit $G * x$.

Theorem 3.2. *If the DDHAP assumption holds for the finite abelian group G acting on the set X, and the hash function family $\mathcal{H}$ is entropy smoothing, then the public-key encryption scheme $\mathcal{PE}$ is secure in the sense of IND-CPA.*

Proof. We use a sequence-of-games technique described by Shoup [99] to construct our proof. Let an adversary $\mathcal{I}$ participate in Game 0 (see Algorithm 3.3), which is exactly the standard IND-CPA experiment.

Alg. 3.3 Game 0

1: $\mathrm{sk} \xleftarrow{R} G$, $y \leftarrow \mathrm{sk} * x$, $k \xleftarrow{R} K$
2: $(m_0, m_1, j) \leftarrow \mathcal{I}_1(y, k)$
3: $d \xleftarrow{R} \{0,1\}$
4: $a \xleftarrow{R} G$, $u \leftarrow a * y$, $h \leftarrow H_k(u)$, $z \leftarrow a * x$, $c \leftarrow h \oplus m_d$
5: $d' \leftarrow \mathcal{I}_2(j, c, z)$

Algorithm 3.4 defines Game 1. The only difference from Game 0 is that u is now chosen at random from the orbit $G * x$.

Let us define E_i to be the event when $d = d'$ in Game i.

Alg. 3.4 Game 1

1: $\mathrm{sk} \xleftarrow{R} G$, $y \leftarrow \mathrm{sk} * x$, $k \xleftarrow{R} K$
2: $(m_0, m_1, j) \leftarrow \mathcal{I}_1(y, k)$
3: $d \xleftarrow{R} \{0,1\}$
4: $a \xleftarrow{R} G$, $\boxed{u \xleftarrow{R} G * x}$, $h \leftarrow H_k(u)$, $z \leftarrow a * x$, $c \leftarrow h \oplus m_d$
5: $d' \leftarrow \mathcal{I}_2(j, c, z)$

Alg. 3.5 DDHAP distinguisher $\mathcal{S}_{01}$

Input: $(y, z, u) \in X^3$
1: $k \xleftarrow{R} K$
2: $(m_0, m_1, j) \leftarrow \mathcal{I}_1(y, k)$
3: $d \xleftarrow{R} \{0,1\}$
4: $h \leftarrow H_k(u)$, $c \leftarrow h \oplus m_d$
5: $d' \leftarrow \mathcal{I}_2(j, c, z)$
Output: $d \oplus d'$

We construct a DDHAP distinguisher $\mathcal{S}_{01}$ as shown in Algorithm 3.5. $\mathcal{S}_{01}$ outputs 0 when $\mathcal{I}$ guesses correctly, and 1 otherwise. When $\mathcal{S}_{01}$ receives a "right" triple (y, z, u) on input, i.e. when $y = a * x$, $z = b * x$ and $u = ab * x$, the probability distributions observed by $\mathcal{I}$ when run in $\mathcal{S}_{01}$ are equivalent to those in Game 0, and we have that

$$\Pr\left[\ \mathcal{S}_{01}(a*x, b*x, ab*x) = 0 \ \middle|\ (a,b) \xleftarrow{R} G^2\ \right] = \Pr[E_0].$$

Similarly we observe that

$$\Pr\left[\ \mathcal{S}_{01}(a*x, b*x, c*x) = 0 \ \middle|\ (a,b,c) \xleftarrow{R} G^3\ \right] = \Pr[E_1].$$

As a result,

$$\mathrm{Adv}^{\mathrm{DDH}}_{\mathcal{S}_{01}} = \left|\Pr^{\mathrm{DDH}}_{\mathcal{S}_{01}} - \frac{1}{2}\right| = \left|\frac{\Pr[E_0] + (1 - \Pr[E_1])}{2} - \frac{1}{2}\right| = \frac{1}{2}\left|\Pr[E_0] - \Pr[E_1]\right|. \tag{3.3}$$

It may seem that in Game 1 the adversary $\mathcal{I}$ already has no information about the bit d. But this is not exactly right. For instance, $\mathcal{I}$ can compute $h_0 \leftarrow c \oplus m_0$ and $h_1 \leftarrow c \oplus m_1$. If there is a way to check that one of the h_0, h_1 is not a hash image of an element from $G * x$, then $\mathcal{I}$ can conclude on the value of d. We see that the ability of the hash function to hide preimage is important, what is expressed by the entropy smoothing property.

We proceed with Game 2, where h is now a random bit string (see Algorithm 3.6).

Alg. 3.6 Game 2

1: $\mathrm{sk} \xleftarrow{R} G$, $y \leftarrow \mathrm{sk} * x$, $k \xleftarrow{R} K$
2: $(m_0, m_1, j) \leftarrow \mathcal{I}_1(y, k)$
3: $d \xleftarrow{R} \{0,1\}$
4: $a \xleftarrow{R} G$, $u \xleftarrow{R} G * x$, $\boxed{h \xleftarrow{R} \{0,1\}^w}$, $z \leftarrow a * x$, $c \leftarrow h \oplus m_d$
5: $d' \leftarrow \mathcal{I}_2(j, c, z)$

Alg. 3.7 ES distinguisher $\mathcal{S}_{12}$

Input: $(k, h) \in K \times \{0,1\}^w$
1: $\mathrm{sk} \xleftarrow{R} G$, $y \leftarrow \mathrm{sk} * x$
2: $(m_0, m_1, j) \leftarrow \mathcal{I}_1(y, k)$
3: $d \xleftarrow{R} \{0,1\}$
4: $a \xleftarrow{R} G$, $z \leftarrow a * x$, $c \leftarrow h \oplus m_d$
5: $d' \leftarrow \mathcal{I}_2(j, c, z)$
Output: $d \oplus d'$

Algorithm 3.7 illustrates an ES distinguisher $\mathcal{S}_{12}$.

When a tuple $(k, H_k(u))$ with random $k \in K$ and $u \in G * x$ is supplied to $\mathcal{S}_{12}$, the adversary $\mathcal{I}$ observes the same probability distributions as in Game 1. On the other hand, when (k, h) is supplied such that $k \in K$ and $h \in \{0,1\}^w$ are random, we get the setting of Game 2. Hence

$$\mathrm{Adv}^{\mathrm{ES}}_{\mathcal{S}_{12}} = \frac{1}{2} \left| \Pr[E_1] - \Pr[E_2] \right| . \tag{3.4}$$

Since in Game 2 h is chosen at random, the encryption $c = h \oplus m_d$ is equivalent to the one-time pad. So $\mathcal{I}$'s output is independent of the bit d, meaning that

$$\Pr[E_2] = \frac{1}{2}. \tag{3.5}$$

Getting together (3.3), (3.4) and (3.5) and applying the triangle inequality, we have

$$\begin{aligned} \left| \mathrm{Adv}^{\mathrm{IND}}_{\mathcal{I}} \right| = \left| \Pr[E_0] - \Pr[E_1] + \Pr[E_1] - \Pr[E_2] + \Pr[E_2] - \tfrac{1}{2} \right| \leq \\ \leq 2 \left| \mathrm{Adv}^{\mathrm{DDH}}_{\mathcal{S}_{01}} \right| + 2 \left| \mathrm{Adv}^{\mathrm{ES}}_{\mathcal{S}_{12}} \right| . \end{aligned}$$

Note that when $\mathcal{I}$ is polynomially bounded, so are the distinguishers $\mathcal{S}_{01}$ and $\mathcal{S}_{12}$. Hence if follows from the DDHAP assumption and the ES assumption that the advantage $\mathrm{Adv}^{\mathrm{IND}}_{\mathcal{I}}$ is negligible.

□

3.6 Concluding Remarks

As it has been noticed by Koblitz and Menezes [69], reductionist security arguments should be taken with care. To illustrate, the following considerations can be expressed about the security arguments provided in our paper.

To begin with, the computational problem used in the reductions is somewhat artificial. The DDHA problem is not harder than the corresponding CDHA problem, which is in turn not harder than the corresponding GAI problem. In practice, more research is often devoted to the latter two problems, leaving the former problem unexamined. However only well-studied problems can provide some sort of confidence about their hardness to the scientific community. Without a careful analysis of the DDHA problem complexity, nothing credible can be said about the security of cryptographic schemes based on this problem, and the security of the $\mathcal{KA}$ protocol and the $\mathcal{PE}$ scheme in particular.

Secondly, the reduction provided in Theorem 3.1 is not tight. Namely, having an adversary with an advantage ϵ, the advantage of the DDHAP distinguisher constructed during the proof is ϵ/l. Although l is polynomially bound, it might still be so big that the attack is not practical for solving the DDHAP instance, i.e. the advantage is too small to effectively obtain the solution. This inconsistency between asymptotic and practical results can damage one's assurance in security.

It should be also noted that our security argument for the $\mathcal{KA}$ protocol only concerns (concurrent sessions of) the $\mathcal{KA}$ protocol in isolation. We have not considered the question of composing the $\mathcal{KA}$ protocol with other cryptographic schemes, such as authentication protocols or encryption schemes. A trivial example of such a bad composition could be a weak symmetric cipher with the session key established by the $\mathcal{KA}$ protocol. If the cipher allows some kind of attack, then the whole composition is obviously insecure. Cremers has shown that many cryptographic protocols, proven to be correct in isolation, are vulnerable to multi-protocol attacks [32]. To prevent these attacks one may, for example, separate key material between different protocols, or tag protocol messages according to their context.

To summarize, in case when the DDHA problem is hard for a finite abelian group G acting on a set X, the result of this paper assures that the key agreement protocol $\mathcal{KA}$ based on G and X is secure when used over a channel providing authenticity, and that the public-key encryption scheme $\mathcal{PE}$ based on G and X is secure when the adversary does not have access to a decryption oracle and when the used hash function family has good pseudorandom generation capabilities.

Paper III

Chapter 4, excluding Appendices 4.A–4.C, is archived in

Steven Galbraith and Anton Stolbunov

Improved Algorithm for the Isogeny Problem for Ordinary Elliptic Curves

Preprint (2011), `http://arxiv.org/abs/1105.6331v1`

Chapter 4

Improved Algorithm for the Isogeny Problem for Ordinary Elliptic Curves

Co-authored with Steven Galbraith

ABSTRACT. A low storage algorithm for constructing isogenies between ordinary elliptic curves was proposed by Galbraith, Hess and Smart (GHS). We give an improvement of this algorithm by modifying the pseudorandom walk so that lower-degree isogenies are used more frequently. This is motivated by the fact that high degree isogenies are slower to compute than low degree ones. We analyse the running time of the parallel collision search algorithm when the partitioning is uneven. We also give experimental results. We conclude that our algorithm is around 14 times faster than the GHS algorithm when constructing horizontal isogenies between random isogenous elliptic curves over a 160-bit prime field.

The results apply to generic adding walks and the more general group action inverse problem; a speed-up is obtained whenever the cost of computing edges in the graph varies significantly.

4.1 Introduction

Let E_1 and E_2 be elliptic curves over a finite field $\mathbb{F}_q$. If $\#E_1(\mathbb{F}_q) = \#E_2(\mathbb{F}_q)$ then there is an isogeny $\phi : E_1 \to E_2$ over $\mathbb{F}_q$ [110, Theorem 1]. The *isogeny problem* is to compute such an isogeny.

Problem 4.1 (Isogeny Problem)**.** *Let $E_1/\mathbb{F}_q$ and $E_2/\mathbb{F}_q$ be ordinary elliptic curves satisfying $\#E_1(\mathbb{F}_q) = \#E_2(\mathbb{F}_q)$. Compute an $\mathbb{F}_q$-isogeny $\phi : E_1 \to E_2$.*

The isogeny problem for ordinary elliptic curves (we do not consider the supersingular case in this paper, though it is also interesting) over finite fields is a natural problem, which has at least two important applications in cryptography.

First, it allows to understand whether the difficulty of the discrete logarithm problem (DLP) is equal for all elliptic curves with the same number of points over $\mathbb{F}_q$. If E_1 and E_2 are ordinary then $\mathcal{O}_1 = \mathrm{End}_{\overline{\mathbb{F}}_q}(E_1)$ and $\mathcal{O}_2 = \mathrm{End}_{\overline{\mathbb{F}}_q}(E_2)$ are orders in a quadratic imaginary field K. Let $\mathcal{O}_K$ be the ring of integers of K and define the *conductor* $c(E_i) = [\mathcal{O}_K : \mathcal{O}_i]$ for $i = 1, 2$. If there is a large prime ℓ such that $\ell \mid c(E_1)$ and $\ell \nmid c(E_2)$ (or vice versa) then it seems to require at least ℓ^2 operations in $\mathbb{F}_q$ to compute an isogeny between E_1 and E_2, as explained in Section 4.6.1. However, if this does not happen (in which case we say that the curves have *comparable conductors*) then it can be feasible to compute an isogeny from E_1 to E_2 using the algorithms due to Galbraith [44] or Galbraith, Hess and Smart [45] (GHS); the heuristic complexity is $\tilde{O}(q^{1/4+o(1)})$ bit operations. As has been observed by Jao, Miller and Venkatesan [61], and further discussed by Koblitz, Koblitz and Menezes [68, §11], it follows that the DLP is random self-reducible among curves with the same number of points and comparable conductors.

Second, the problem of constructing isogenies between ordinary elliptic curves is the basis of security of some recently proposed cryptographic schemes [30, 35, 89, 107, 113]. Cryptographic key sizes for these schemes should be chosen based on the complexity of the isogeny problem.

Galbraith, Hess and Smart [45] gave an algorithm, based on pseudorandom walks in the isogeny graph, to solve the problem. At each step in the GHS algorithm an isogeny of relatively small degree ℓ is computed. The starting point of our work is the observation that the cost of computing an isogeny depends on ℓ (see Fig. 4.3), and so it makes sense to choose a pseudorandom walk which "prefers" to use the fastest possible isogenies. Similar ideas have also been used previously by authors: Bisson and Sutherland [13] in their algorithm for computing the endomorphism ring of ordinary elliptic curves; Stolbunov [107] in a family of cryptographic schemes based on isogenies.

The main problem is that making the pseudorandom walks "uneven" means that the walks are "less random", and so the number of steps in the algorithm to solve the isogeny problem increases. However, this increase in cost is offset by the saving in the cost of computing isogenies. We analyse the effect of "uneven" partitions and suggest some good choices of parameters for the algorithm. We also give experimental results to support our analysis.

The paper is organised as follows. In Section 4.2 we introduce a generalisation of the isogeny problem called the *group action inverse problem* (GAIP). We then explain why the isogeny problem is the same as GAIP in the case of an ideal class group; we call this the $\mathcal{CL}$-GAIP. In Section 4.3 we re-formulate (a variant of) the GHS algorithm as a generic algorithm for solving the GAIP and describe how it applies to the $\mathcal{CL}$-GAIP. In Section 4.4 we provide a theoretical analysis of the expected

running time of the idealised algorithm. Section 4.5 discusses how the idealised algorithm and the real implementation differ, and gives some experimental results. Section 4.6 then makes some predictions about how the algorithm will perform for isogeny computations, and determines the speedup of our ideas compared with the algorithm described by Galbraith, Hess and Smart. The main consequence of our work is that the isogeny problem can be solved in less than one tenth of the time of the GHS algorithm.

4.2 Definitions and Notation

4.2.1 The Group Action Inverse Problem

Let G be a finite abelian group, and X a non-empty set. A (left) action of G on X is a map

$$\begin{aligned} G \times X &\to X \\ (g, x) &\mapsto g * x, \end{aligned}$$

which satisfies the associativity property $(gh)*x = g*(h*x)$ for all $g, h \in G$, $x \in X$, and the property $e * x = x$ for the identity element $e \in G$ and all $x \in X$. The *orbit* of a set element $x \in X$ is the subset $G * x = \{g * x \mid g \in G\}$. The orbits of the elements of X are equivalence classes. The *stabilizer* of x is the set of all elements in G that fix x: $G_x = \{g \in G \mid g * x = x\}$.

Problem 4.2 (Group Action Inverse Problem)**.** *Let G be a finite abelian group acting on a non-empty set X. Given elements $x, y \in X$, find a group element $g \in G$ such that $g * x = y$.*

When the action of G on X is *transitive*, that is, X is finite and there is only one orbit, then the GAIP has at least one solution. When the action is *free*, i.e. the stabilizer of any set element is trivial, then the GAIP has at most one solution. In the case of a free and transitive action, the set X is called a *principal homogeneous space* for the group G, and the GAIP has exactly one solution. This last type of GAIP will be considered in the rest of the paper.

4.2.2 The Isogeny Problem and the Class Group Action Inverse Problem

Recall from the introduction that E_1 and E_2 are ordinary elliptic curves over $\mathbb{F}_q$ with $\#E_1(\mathbb{F}_q) = \#E_2(\mathbb{F}_q)$, $\mathcal{O}_i = \mathrm{End}_{\overline{\mathbb{F}}_q}(E_i)$ and $c(E_i) = [\mathcal{O}_K : \mathcal{O}_i]$ for $i = 1, 2$. As noted by Galbraith [44] (building on work of Kohel [71]), a natural approach to compute an isogeny from E_1 to E_2 is to first take "vertical" isogenies to elliptic curves E_1' and E_2' such that $\mathrm{End}_{\overline{\mathbb{F}}_q}(E_i') = \mathcal{O}_K$, and the isogeny problem is reduced to computing

a "horizontal" isogeny from E_1' to E_2'. Alternatively, if $\mathcal{O}_1$ and $\mathcal{O}_2$ are comparable, but both $c(E_1)$ and $c(E_2)$ have a large prime factor, one can use horizontal and/or vertical isogenies from E_1 to a curve E_1' such that $\mathrm{End}_{\overline{\mathbb{F}}_q}(E_1') = \mathcal{O}_2$ and the problem is again reduced to computing a horizontal isogeny.

So, without loss of generality, we assume for the remainder of the paper that $\mathrm{End}_{\overline{\mathbb{F}}_q}(E_1) = \mathrm{End}_{\overline{\mathbb{F}}_q}(E_2)$. Define $\mathcal{O}$ to be the order $\mathrm{End}_{\overline{\mathbb{F}}_q}(E_1)$. Write $\mathcal{CL}(\mathcal{O})$ for the group of invertible $\mathcal{O}$-ideals modulo principal $\mathcal{O}$-ideals and $h(\mathcal{O})$ for the order of $\mathcal{CL}(\mathcal{O})$.

The theory of complex multiplication (CM) implies that there are $h(\mathcal{O})$ isomorphism classes of elliptic curves E over $\mathbb{F}_q$ with $\mathrm{End}_{\overline{\mathbb{F}}_q}(E) = \mathcal{O}$ and a fixed number of points $\#E(\mathbb{F}_q)$. There is a (non-canonical) one-to-one-correspondence between isomorphism classes of elliptic curves E over $\mathbb{F}_q$ with $\mathrm{End}_{\overline{\mathbb{F}}_q}(E) = \mathcal{O}$ and ideal classes in $\mathcal{CL}(\mathcal{O})$ [119]. There is a (canonical) one-to-one correspondence between invertible $\mathcal{O}$-ideals $\mathfrak{l}$ and isogenies, such that if $\mathfrak{l}$ is an ideal of norm ℓ and E is an elliptic curve corresponding to the ideal $\mathfrak{a}$ then there is an ℓ-isogeny from E to E' where E' corresponds to the ideal $\mathfrak{a}\mathfrak{l}^{-1}$. Galbraith, Hess and Smart [45] show how, given an elliptic curve E and an ideal $\mathfrak{b}$, one can efficiently compute an explicit isogeny $\phi : E \to E'$ corresponding to $\mathfrak{b}$ via the above correspondence.

Let X be the set of isomorphism classes of elliptic curves over $\mathbb{F}_q$ with $\mathrm{End}_{\overline{\mathbb{F}}_q}(E) = \mathcal{O}$ and a fixed $\#E(\mathbb{F}_q)$. It follows that $\mathcal{CL}(\mathcal{O})$ acts on X and so we can define $\mathfrak{b} * E$ to be the isomorphism class of the image curve for the isogeny corresponding to $\mathfrak{b}$. The horizontal isogeny problem is a special case of the GAIP, which we call the class group action inverse problem ($\mathcal{CL}$-GAIP).

Problem 4.3 (Class Group Action Inverse Problem)**.** *Let $E_1/\mathbb{F}_q$ and $E_2/\mathbb{F}_q$ be ordinary elliptic curves satisfying $\#E_1(\mathbb{F}_q) = \#E_2(\mathbb{F}_q)$ and $\mathrm{End}_{\overline{\mathbb{F}}_q}(E_1) = \mathrm{End}_{\overline{\mathbb{F}}_q}(E_2) = \mathcal{O}$. Find the ideal class $[\mathfrak{b}] \in \mathcal{CL}(\mathcal{O})$ such that the curves $\mathfrak{b} * E_1$ and E_2 are isomorphic.*

Hence, for the rest of the paper we study the GAIP, keeping in mind this specific application.

Let $H = \{\mathfrak{l}_1, \ldots, \mathfrak{l}_r\}$ be a set of distinct prime ideals. We define the *ideal class graph* to be the graph with vertex set $\mathcal{CL}(\mathcal{O})$ and, for each $\mathfrak{l} \in H$, an edge $(\mathfrak{a}, \mathfrak{a}\mathfrak{l}^{-1})$ for all $\mathfrak{a} \in \mathcal{CL}(\mathcal{O})$. Similarly, we define the *isogeny graph* to have vertex set being isomorphism classes of elliptic curves with endomorphism ring $\mathcal{O}$ and an edge between two isomorphism classes if there is an isogeny between them corresponding to an ideal $\mathfrak{l} \in H$.

4.2.3 Other Notation

By $a \leftarrow b$ we denote the assignment of value b to a variable a. By $a \xleftarrow{R} G$ we mean that a is sampled from the uniform distribution on the set of elements of G. We write $\#S$ for the number of elements in S. By $\log(n)$ we denote the binary logarithm

of n. All equalities of the form $f(x) = O(g(x))$ are one-way equalities that should be read as "$f(x)$ is $O(g(x))$".

4.3 Algorithm for Solving the GAIP and the $\mathcal{CL}$-GAIP

4.3.1 Previous Isogeny Problem Algorithms

The first algorithm for solving the isogeny problem (equivalently, the $\mathcal{CL}$-GAIP) was proposed by Galbraith [44]. Let E_1 and E_2 be elliptic curves over $\mathbb{F}_q$ with $\mathrm{End}(E_i) = \mathcal{O}$ (alternatively, let x an y be $\mathcal{O}$-ideal classes). The idea was to construct two graphs of elliptic curves (subgraphs of the isogeny graph), one rooted at E_1 and the other at E_2 (equivalently, two subgraphs of the ideal class graph rooted at x and y respectively). Edges in the graph correspond to small-degree ideals. By the birthday paradox, when the graphs have total size approximately $\sqrt{\pi h(\mathcal{O})}$ one expects them to have a vertex in common, in which case we have a path of isogenies from E_1 to E_2. Indeed, under the assumption that the graphs behave like random subgraphs from the point of view of their intersection, it is natural to conjecture that the algorithm halts when the total number of vertices visited is, on average, $\sqrt{\pi h(\mathcal{O})}$. Note that this algorithm requires an exponential amount of time and memory.

The second, and previously the best, algorithm was due to Galbraith, Hess and Smart [45] (in particular the stage 1 of the algorithm described in that paper). The major improvement was to use pseudorandom walks and parallel collision search in the isogeny graph, rather than storing entire subgraphs. We give a generic description of this method in the next section. The advantage of the GHS method is that it only requires a polynomial amount of memory, and can be easily parallelised or distributed.

Although this paper considers the classical computational model, we note that a subexponential-time quantum algorithm for the isogeny problem has been proposed by Childs, Jao and Soukharev [26].

4.3.2 Generic Description of the GAIP Solving Algorithm

Let the GAIP (x, y) be defined for a group G acting on a set X, and let r be a positive integer greater than or equal to the rank of G. Choose a generating set $H = \{g_1, \ldots, g_r\} \subset G$ and consider a graph Γ with vertices the elements of X, and edges $(z, g_i * z)$, for all $1 \leq i \leq r$. In the special case $G = \mathcal{CL}(\mathcal{O})$, X the set of isomorphism classes of elliptic curves with the endomorphism ring $\mathcal{O}$, and $H = \{\mathfrak{l}_1, \ldots, \mathfrak{l}_r\}$, we obtain the isogeny graph defined in Section 4.2.2.

To solve the GAIP it suffices to find an (undirected) path in Γ between x and y.

A natural way to do this is to use (pseudo)random walks in Γ, starting from x and y. For instance one can use a random function $v\colon X \to \{1, \ldots, r\}$ and the map

$$\begin{aligned} \psi\colon X &\to X \\ z &\mapsto g_{v(z)} * z. \end{aligned}$$

The following language will be used throughout the paper: the function $v(z)$ is a *partitioning function*, because it defines a partition P on the set X. By an abuse of notation we will call parts in P *partitions*. Note that we do not require all partitions to be of the same size. *Partitioning probabilities* $p_1, \ldots, p_r$ are defined as

$$p_i = \Pr\left[v(z) = i \mid z \xleftarrow{R} X\right] \qquad \text{for all } 1 \leq i \leq r.$$

A *walk* on Γ is a sequence of nodes computed as

$$z_{j+1} = \psi(z_j).$$

A *hop* is one edge in the graph (i.e., one step of the walk). The set H is called *the supporting set* for walks on Γ. The above walk is a generalization of the *adding walk* proposed by Teske for groups [111].

One can apply the parallel collision search concept, as proposed by van Oorschot and Wiener [115]. To do this, define a subset X_D of *distinguished elements* in X, such that it is easy to verify that $z \in X_D$. Pseudorandom walks in Γ are formed by taking a random initial vertex[1], moving along edges with a certain probability, and halting when the current vertex is a distinguished element. This framework was used by Galbraith, Hess and Smart [45]. Figure 4.1 presents Algorithm $\mathcal{A}$, which is an algorithm to solve the GAIP following this approach.

Algorithm $\mathcal{A}$ uses $2t$ client threads, where $t \geq 1$, and one server thread. The algorithm takes as input a GAIP instance (x_0, x_1) and an integer t. The server starts t clients, each performing a walk starting from a randomized node $h_{0,i} * x_0$ for $1 \leq i \leq t$. The server starts another t clients, each performing a walk starting from a randomized node $h_{1,i} * x_1$. Each client continues the deterministic pseudorandom walk until it hits a distinguished node. Once a thread hits a distinguished node $z = a * x_s$, it puts the triple (z, a, s) on the shared queue and terminates. The server stores all received triples in a database D and restarts clients from new randomized starting nodes.

A *collision* is an event when some node is visited by client threads twice, while the preceding nodes visited by the threads are different. Since the walks are deterministic, after a collision the two threads follow the same route unless they hit a distinguished node. Thus every collision results in two triples of the form $(z, \cdot, \cdot)$

[1]The GHS algorithm [45] does not specify how to sample random vertices in the isogeny graph. We use an algorithm from Stolbunov [107, §6.1], which will be briefly explained at the end of Section 4.3.4.

Alg. 4.1 Server

Input: $(x_0, x_1, t) \in X \times X \times \mathbb{N}$

1: **for** $i = 1$ to t **do**
2: $(h_0, h_1) \xleftarrow{R} G \times G$
3: start client$(h_0 * x_0, h_0, 0)$
4: start client$(h_1 * x_1, h_1, 1)$
5: **end for**
6: $D \leftarrow \{\}$
7: **while** true **do**
8: fetch (z, a, s) from queue
9: **if** $(z, b, 1 - s) \in D$ for some b **then**
10: break loop
11: **end if**
12: $D \leftarrow D \cup \{(z, a, s)\}$
13: $h \xleftarrow{R} G$
14: start client$(h * x_s, h, s)$
15: **end while**
16: stop all clients

Output: $a^{1-2s} b^{2s-1}$

Alg. 4.2 Client

Input: $(z, a, s) \in X \times G \times \{0, 1\}$

1: $c \leftarrow 0$
2: **while** $z \notin X_D$ **do**
3: $i \leftarrow v(z)$
4: $z \leftarrow g_i * z$
5: $a \leftarrow a g_i$
6: $c \leftarrow c + 1$
7: **if** $c > c_{\max}$ **then**
8: $(z, a, s) \leftarrow \perp$
9: break loop
10: **end if**
11: **end while**

Output: (z, a, s)

Figure 4.1: Algorithm $\mathcal{A}$ for solving the GAIP.

being submitted to the server. A collision of walks, one of which was started from x_0 and the other one from x_1, is called a *good collision*. After a good collision the server detects two triples $(z, a, 0)$ and $(z, b, 1)$. It then halts all clients and outputs the solution $b^{-1}a$.

Since a walk might loop before it hits a distinguished node, clients use a simple loop detection mechanism that checks whether the walk remains shorter than a fixed maximum length $c_{\max}$. The value $c_{\max}$ is usually chosen to be a function of θ, e.g. $c_{\max} = 30/\theta$, which means that walks 30 times longer than expected are abandoned[2].

Denote by α the number of nodes visited by Algorithm $\mathcal{A}$, counted with repetition. If nodes were sampled uniformly at random then the expected value $\mathrm{E}(\alpha)$ would be close to $\sqrt{\pi \# G}$ by a variant of the birthday paradox (see Section 4.4.1). The expected total (serial) running time of Algorithm $\mathcal{A}$ approximately equals the product of $\mathrm{E}(\alpha)$ with the average cost of computing $g_i * z$ in line 4 of the client algorithm[3]. Our main observation is that the cost of computing $g_i * z$ is not the same for all g_i. Hence, one can speed up the algorithm by favoring the g_i which are faster to compute.

In the $\mathcal{CL}$-GAIP, the supporting set H is usually chosen to consist of prime ideals above the smallest integer primes which split in $\mathcal{O}$. In some rare cases it may be necessary to add one or more prime ideals of larger norm to ensure that H generates $\mathcal{CL}(\mathcal{O})$. Ramified primes can also be used, but since their order equals two in $\mathcal{CL}(\mathcal{O})$ they suffer from the defect mentioned in the next section.

4.3.3 A Remark on the GHS Algorithm

The GHS paper [45] states that "it is usually enough that H contains about 16 distinct split primes", and the partitioning function should "have a distribution close to uniform". In other words, it was advised to use about $r = 16$ partitions of approximately equal size. We will compare our algorithm against those suggested parameters in the remainder of the paper.

We note a potentially serious problem[4] with the algorithm of Galbraith, Hess and Smart [45]. On every hop the algorithm chooses a small prime ℓ and a bit b uniformly at random. Typically, ℓ is split and the algorithm chooses one of the two ℓ-isogenous elliptic curves deterministically using the bit b. Hence for a fixed ℓ, every hop where ℓ is chosen produces an action by, equally likely, the ideal $\mathfrak{l}$ or $\mathfrak{l}^{-1}$ (where $(\ell) = \mathfrak{l}\mathfrak{l}^{-1}$). Thus, since the ideal class group is abelian, the expected power of the ideal $\mathfrak{l}$ that has acted on the starting elliptic curve after any number of hops equals

[2] Van Oorschot and Wiener [115] suggest $c_{\max} = 20/\theta$. Our value is larger in order to preserve more non-looped walks.

[3] We do not count database access times and expected $L\theta\sqrt{n}$ random samplings of a group element.

[4] This remark also applies to the isogeny walk given by Teske [113, Algorithm 1]. Interestingly, another isogeny walk is given in Algorithm 3 of the same paper, which is not affected by this problem.

0. Such a walk is far from random, as it tends to remain "close" to its initial node. Hence, most likely the method of Galbraith, Hess and Smart does not perform as well in practice as the heuristic predictions stated in [45]. To avoid this problem, our algorithm always acts by the same ideal $\mathfrak{l}$ when the prime ℓ is chosen (i.e., the set H never contains both $\mathfrak{l}$ and $\mathfrak{l}^{-1}$; unless $\mathfrak{l}$ is ramified). We stress that the speed improvement of our algorithm is not due to the correction of the named flaw but because of the use of an uneven partitioning.

4.3.4 Better Choices for Solving the $\mathcal{CL}$-GAIP

We now discuss the main idea of the paper, which is to make the pseudorandom walks faster by using smaller degree prime ideals more often than larger degree ones.

Recall that α denotes the number of nodes visited by Algorithm $\mathcal{A}$, counted with repetition, and that $\mathrm{E}(\alpha)$ is close to $\sqrt{\pi n}$, where $n = \#G$. Therefore it is more convenient to consider the variable

$$L = \frac{\alpha}{\sqrt{n}}.$$

The value of L is fully determined by the group, the problem instance (x_0, x_1), the supporting set H, the partitioning function $v()$, the subset X_D of distinguished nodes, the loop detection value $c_{\max}$ and the random choices made by the algorithm. We define $\mathrm{E}(\, L \mid r, \vec{p}, m, \theta, c_{\max}\,)$ to be the expected value of L, taken over random choices of all the above parameters, conditioned on the values of the parameters:

r the number of partitions;
$\vec{p} = (p_1, \ldots, p_r)$ the partitioning probabilities;
$m = \lceil \log(n) \rceil$ the ceiling function of the binary logarithm of $\#G$;
θ the probability of distinguished nodes;
$c_{\max}$ the loop detection value.

To shorten the notation we will write $\mathrm{E}(L)$ instead of $\mathrm{E}(\, L \mid r, \vec{p}, m, \theta, c_{\max}\,)$.

The average running time of a step in the algorithm (equivalently, hop) is $\vec{p}\,\vec{t} = \sum_{i=1}^{r} p_i t_i$, where $\vec{t}$ is a column vector of timings of actions by the r chosen primes (see Fig. 4.3 for such timings). Hence, the expected serial running time of Algorithm $\mathcal{A}$ is approximately

$$\mathrm{E}(L)\,\sqrt{n}\,\vec{p}\,\vec{t}. \tag{4.1}$$

Ideally, the number of partitions r and the probability distribution $\vec{p}$ should be chosen by solving the optimization problem: given $n, \theta, \vec{t}$, choose r and $\vec{p}$ to minimize the expected running time $\mathrm{E}(L)\,\sqrt{n}\,\vec{p}\,\vec{t}$. We do not claim in this paper a complete solution to this optimization problem. But we do discuss how $\mathrm{E}(L)$ depends on r and $\vec{p}$, and we suggest some choices for these parameters.

For simplicity, and because they seem to give good results in practice, we restrict our attention to vectors $\vec{p} = (p_1, \ldots, p_r)$ such that the probabilities are in geometric

progression $p_{i+1}/p_i = w$ for $1 \leq i < r$. For example, taking $r = 4$ and $w = 1/2$ means probabilities $(p_1, \frac{1}{2}p_1, \frac{1}{4}p_1, \frac{1}{8}p_1)$ which add up to 1 (and so $p_1 = 8/15 \approx 0.53$). In our practical analysis we restrict to $3 \leq r \leq 16$ and $\vec{p}$ is the geometric progression of ratio $w \in \{1, 3/4, 1/2, 1/3, 1/4\}$. This choice is probably not the best solution to the optimization problem, but it seems to work well in practice.

To implement the starting randomization of the walks we use a method proposed by Stolbunov [107, §6.1]. We briefly describe the method. Since the class group structure computation is much faster [11] than Algorithm $\mathcal{A}$, one first computes the class group structure. For an imaginary quadratic order $\mathcal{O}$ of discriminant Δ, the class group $\mathcal{CL}(\mathcal{O})$ is generated (assuming GRH) by the set $\mathcal{L}$ of prime ideals of split norms less than or equal to $\ell_{\max} = c_1 \log^2|\Delta|$, for an effectively computable constant c_1 [95, Corollary 6.2]. Note that the set $\mathcal{L}$ used for the random sampling can be larger than the supporting set H. Knowing the class group generators and their orders, one obtains a random group element in a smooth form by raising generators to random exponents, each chosen between zero and the corresponding order. To shorten the representation one reduces it modulo the lattice of relations among the elements of $\mathcal{L}$. Indeed, it is possible to write any element of $\mathcal{CL}(\mathcal{O})$ as an $O(\log|\Delta|)$-term product of elements in $\mathcal{L}$. Jao, Miller and Venkatesan have shown (assuming GRH) that the ideal class graph $(\mathcal{CL}(\mathcal{O}), \mathcal{L})$ is an expander graph [62, Theorem 1.5]. Since the diameter of an expander graph is less than or equal to $2\log(h)/\log(1+c)$ for the expansion coefficient c and the number of vertices h [47, Theorem 9.9], the diameter of the ideal class graph $(\mathcal{CL}(\mathcal{O}), \mathcal{L})$ is $O(\log(h))$, where $h \approx |\Delta|^{1/2}$.

4.4 Theoretical Analysis of the Algorithm

4.4.1 Previous Results

A tremendous amount of research on the running time analysis of the Pollard rho algorithm has been carried out by various authors. We give a brief overview of some of the results relevant to our work.

First we consider random mappings on a set X of n elements. Rapoport [88, §II] and Harris [54, §3] obtained an approximation for the expected value of the number ρ of distinct elements in a random walk on X:

$$\mathrm{E}(\rho) \approx \sqrt{\frac{\pi n}{2}}.$$

For a more precise statement see Knuth [66, Exercise 3.1.12]. These results were subsequently used to approximate the expected length of the rho-shaped walk in the Pollard's algorithm [83].

Van Oorschot and Wiener [115, §4.1] proposed a parallel version of the Pollard's rho algorithm. When more than one walk is run in parallel, several collisions can occur, and only some of them may be useful (we call these collisions good). Let

$\mathfrak{p}$ be the probability that a random collision is good. They obtained the following approximation for the expected value of the number λ of distinct visited nodes, when the number of collisions is small:

$$\mathrm{E}(\lambda) \approx \sqrt{\frac{\pi n}{2\mathfrak{p}}}. \tag{4.2}$$

The iteration function proposed by Pollard [84] for the DLP involved three partitions of approximately equal size: two corresponding to multiplication and one to squaring hops. Teske proposed a different type of iteration function which she called an adding walk [111]. Adding walks allowed more partitions, but it was still preferable to have equally-sized partitions because the costs of iterations were approximately equal. Brent and Pollard [17] and Blackburn and Murphy [14] provided a heuristic argument where they assumed that the restrictions of the iterating function to r equally-sized partitions were random mappings:

$$\mathrm{E}(\rho) \approx \sqrt{\frac{\pi r n}{2(r-1)}}. \tag{4.3}$$

More recently, Bailey et al. [3, Appendix B] employed an uneven partitioning with probabilities p_i, $1 \leq i \leq r$, for the Pollard rho method. Again under the assumption about the randomness of the restrictions of the iterating function, they provided the following heuristic result:

$$\mathrm{E}(\rho) \approx \sqrt{\frac{\pi n}{2(1-\sum_{i=1}^{r} p_i^2)}}, \tag{4.4}$$

which agrees with (4.3) when all p_i are equal. Combining equations (4.2) and (4.4), since the probability that a collision is good is $\mathfrak{p} = 1/2$, would lead to a conjectured expected value of α of $\sqrt{\pi n/(1-\sum_{i=1}^{r} p_i^2)}$. Theorem 4.1 proves this result.

4.4.2 Issues Caused by Uneven Partitioning

When some partitions are used more often than others, walks become less likely to collide. Indeed, a collision involves two edges coming from two different partitions into the same node. Since every node has exactly one outgoing edge, uneven partitioning implies uneven distribution of edges among their types, and hence it becomes less likely to pick two edges of different types. This aspect is studied in the theoretical analysis below.

Another issue caused by uneven partitioning is that walks lose their mixing property, namely they behave less like random mappings than with even partitioning. This aspect is not accounted by our theoretical model, but it is discussed in Section 4.5.1.

4.4.3 Theoretical Model of the Algorithm

We now define an algorithm $\mathcal{A}_\pi$ that closely resembles $\mathcal{A}$. The only differences between $\mathcal{A}_\pi$ and $\mathcal{A}$ are that the walk is implemented using random permutations, and that there is no loop detection (to simplify the proof in the next section we assume that walks never loop before they hit a distinguished node). Walks for $\mathcal{A}_\pi$ are defined as follows. Let $h_1, \ldots, h_r$ be random permutations on X such that $h_i(z) \neq z$ and $h_i(z) \neq h_j(z)$ for all $z \in X$ and $i \neq j$. Walks are now defined using the map

$$\begin{aligned} \psi_\pi \colon X &\to X \\ z &\mapsto h_{v(z)}(z). \end{aligned}$$

Algorithm $\mathcal{A}_\pi$ is obtained from $\mathcal{A}$ by replacing line 4 of the client Algorithm 4.2 with $z \leftarrow h_i(z)$ and deleting lines 7–10. Because of the nature of the walks, Algorithm $\mathcal{A}_\pi$ does not solve the GAIP.

4.4.4 Running Time of the Theoretical Model

We now state the expected running time of Algorithm $\mathcal{A}_\pi$. This is essentially the same result as given in Appendix B of Bailey et al. [3], although their work is for the Pollard rho discrete logarithm problem, whereas we are considering a slightly different situation. We also give a Heuristic 4.2, for the standard deviation of the running time.

Theorem 4.1. *Let n be the cardinality of the set X, θ the probability of a node being distinguished and $p_1, \ldots, p_r$ the probabilities of choosing among r random permutations on X. Then the number α_π of nodes visited, with repetition, before Algorithm $\mathcal{A}_\pi$ terminates, has the following expected value:*

$$\mathrm{E}(\alpha_\pi) = \sqrt{\frac{\pi n}{d}} + \frac{2}{\theta} + O\left(\ln^4(n)\right),$$

where d is the expected in-degree of a visited node excluding the edge used to arrive at this node[5]*:*

$$d = 1 - (1-\theta)\sum_{i=1}^{r} p_i^2. \tag{4.5}$$

Proof. We sketch an outline of the proof and refer to Stolbunov [108] for the details. The proof uses the approach of Blackburn and Murphy [14]. The main task is to determine the expected number of elements sampled before the first good collision.

[5]The term in-degree refers to a graph with the set of vertices X and the edges $(z, \psi_\pi(z))$. For a visited vertex, the number of used incoming edges equals zero if it is a randomized starting vertex, or one otherwise.

It is then standard that $1/\theta$ further steps are required to detect a collision. Note that two collisions are expected in total.

Let $\Lambda \subset X$ denote the set of elements already visited at some stage during the execution of Algorithm $\mathcal{A}_\pi$. For each element $z \in \Lambda$ (except for the starting point) let $z_0 \in \Lambda$ be the previous element in the walk, and suppose z_0 lies in partition i, so that $z = h_i(z_0)$. Let $j \in \{1, \ldots, r\} \setminus \{i\}$. There is an incoming edge to z corresponding to partition j if and only if $h_j^{-1}(z)$ lies in partition j. Under the assumption that the partitions are random, this occurs with probability p_j. Hence, the expected number of edges into z coming from partition j is p_j. Now, since all the permutations are random and independent, the expected number of incoming edges to z is the sum of the expectations for each individual permutation:

$$\sum_{\substack{j=1 \\ j \neq i}}^{r} p_j.$$

Now, summing over all possible choices for i (given that each arises with probability p_i) gives

$$\sum_{i=1}^{r} p_i \sum_{\substack{j=1 \\ j \neq i}}^{r} p_j = \sum_{\substack{1 \leq i,j \leq r \\ i \neq j}} p_i p_j = 1 - \sum_{i=1}^{r} p_i^2.$$

This is the expected number of external incoming hops, for a random non-initial element of Λ. Since the proportion of initial elements equals θ, hence equation (4.5).

The expected number of elements sampled to get a collision is $\sqrt{\pi n/(2d)}$ by the same arguments as used by Brent-Pollard and Blackburn-Murphy. However, a collision is only a good collision with probability $1/2$ so, using the logic behind equation (4.2), one gets the formula $\sqrt{\pi n/d}$. □

Note that the value d in Theorem 4.1 can easily be computed for small r and known p_i. When all $p_i = 1/r$ and θ tends to zero, then d tends to

$$1 - r\frac{1}{r^2} = \frac{r-1}{r}.$$

Hence, Theorem 4.1 agrees with previous results on the Pollard rho algorithm when using r partitions all of the same size, cf. (4.3).

Heuristic 4.2. *Let α_π, n, θ and d be as in Theorem 4.1. Then the variance of the random variable α_π approximates as:*

$$\mathrm{Var}(\alpha_\pi) \approx \frac{(4-\pi)n}{d} + \frac{4-2\theta}{\theta^2} + \frac{1}{\theta}\sqrt{\frac{\pi n}{d}}. \tag{4.6}$$

We provide a brief argument for Heuristic 4.2 below and refer to Stolbunov [108] for the details.

The total number of visited nodes α_π is the sum of the number of unique visited nodes λ_π and the number δ_π of nodes visited twice or more. Hence

$$\mathrm{Var}(\alpha_\pi) = \mathrm{Var}(\lambda_\pi) + \mathrm{Var}(\delta_\pi) + 2\,\mathrm{Cov}(\lambda_\pi, \delta_\pi),$$

where the summands correspond to the ones in (4.6). The probability distribution of λ_π can be approximated by the (continuous) Rayleigh distribution [109] with the following probability density function and variance:

$$f_{\lambda_\pi}(x) \approx \frac{xd}{2n} e^{-\frac{x^2 d}{4n}}, \qquad \mathrm{Var}(\lambda_\pi) \approx \frac{(4-\pi)n}{d}.$$

When it comes to the duplicate visited nodes, chasing the good-collision distinguished node can be described as a sequence of Bernoulli trials with success probability $\theta/2$, because only half of the collisions are good. The number of trials δ_π needed to get one success conforms to the geometric distribution [102, §6.1.2]. Hence the probability mass function and the variance of δ_π are

$$f_{\delta_\pi}(x) = \frac{\theta}{2}\left(1 - \frac{\theta}{2}\right)^{x-1}, \qquad \mathrm{Var}(\delta_\pi) = \frac{4-2\theta}{\theta^2}.$$

The covariance of λ_π and δ_π is computed using the formula (see [108])

$$\mathrm{Cov}(\lambda_\pi, \delta_\pi) = \mathrm{E}(\lambda_\pi \delta_\pi) - \mathrm{E}(\lambda_\pi)\,\mathrm{E}(\delta_\pi).$$

4.4.5 Running Time Calculations

Let the partitioning probabilities $p_1, \ldots, p_r$ be chosen from a geometric progression with common ratio w (cf. Section 4.3.4). Table 4.1 lists the values d, the expected values and the standard deviations of L_π for $n = 2^{80}$ and $\theta = n^{-1/4}$. Mantissas are rounded to four decimal digits.

The values of d in the first column of Table 4.1 agree with $(r-1)/r$ as expected. Note also that the values of $E(L_\pi)$ in the first column converge to the expected asymptotic value of $\sqrt{\pi} \approx 1.7724$. The values in the $w = 1/4$ column do not change significantly when r is large; this is because the higher primes are used with such extremely low probability that they have no effect on the algorithm. The values in Table 4.1 will be used later to give an estimate of the running time of our improved variant of the algorithm.

4.5 Comparing Theory and Practice

There are many reasons why we do not expect the practical Algorithm $\mathcal{A}$ to behave as well as the theoretical Algorithm $\mathcal{A}_\pi$. The aim of this section is to briefly mention one of these issues, and to develop a plausible set of heuristics for the running time of Algorithm $\mathcal{A}$.

	$w = 1$			$w = 1/2$			$w = 1/4$		
r	d	$\mathrm{E}(L_\pi)$	Stdev	d	$\mathrm{E}(L_\pi)$	Stdev	d	$\mathrm{E}(L_\pi)$	Stdev
3	0.6667	2.1708	1.1347	0.5714	2.3447	1.2256	0.3810	2.8717	1.5011
4	0.7500	2.0467	1.0698	0.6222	2.2470	1.1746	0.3953	2.8191	1.4736
5	0.8000	1.9817	1.0359	0.6452	2.2067	1.1535	0.3988	2.8066	1.4671
6	0.8333	1.9416	1.0149	0.6561	2.1882	1.1438	0.3997	2.8035	1.4655
10	0.9000	1.8683	0.9766	0.6660	2.1719	1.1353	0.4000	2.8025	1.4649
16	0.9375	1.8306	0.9569	0.6667	2.1708	1.1347	0.4000	2.8025	1.4649

Table 4.1: The values d, $\mathrm{E}(L_\pi)$ and $\mathrm{Stdev}(L_\pi)$, when r partitions are used and partitioning probabilities decrease with ratio w. $n = 2^{80}$ and $\theta = 2^{-20}$.

4.5.1 Mixing of Adding Walks

As is standard, the theoretical analysis assumes truly random walks. However, we are using adding walks in a group, and such walks are not close to uniformly distributed if they are short. The mixing time is a measure of how long a walk runs before its values start to appear uniformly distributed. It is beyond the scope of this paper to analyse such issues in detail. We mention that Dai and Hildebrand [33] have studied the mixing time of adding walks. They show that adding walks on r partitions need a slack of $O(n^{(2/(r-1))+\epsilon})$ hops before they converge to the uniform distribution.

However, it is worth noting that Algorithm $\mathcal{A}$ does not necessarily need walks to be uniformly distributed after a certain number of hops. Instead it needs walks to collide. Just because walks have not yet reached uniform sampling does not prevent collisions from occurring.

4.5.2 Experiments

To get a better idea of how the algorithm works in practice, we have performed a suite of experiments. We report one of them in this paper and refer to Stolbunov [108] for more details.

Our numerical experiments are for $X = G$ (i.e., G acting on itself) being an abstract group of the form $\mathbb{Z}_{n_1} \oplus \cdots \oplus \mathbb{Z}_{n_s}$, where $n_{i+1} \mid n_i$ and $n_i \geq 2$ for all i. The integer s is the *rank* of G. The supporting set is randomly chosen, though it is checked that it generates the group.

For calculations we use a Linux cluster of 32 quad-core Intel X5550 processors clocked at 2.67 GHz. The code is written in C++. We use a single-threaded implementation of Algorithm $\mathcal{A}$, such that one thread alternates between x_0- and x_1-walks. The same experiment is run on all CPU cores in parallel but with different random generator seeds.

Group elements are represented by arrays of 64-bit integers. We make use of a

hash function $H : G \to \{0,1\}^{32}$ implemented using the 64 to 32 bit hash function of Wang [116]. The partitioning function $v(z)$ is computed by reducing $H(z)$ modulo a sufficiently large integer whose residues can be partitioned with the correct proportions. Wang's hash function uses bit shifts, negations, additions and XOR operations. This helps to make sure that $v(z)$ and $v(\psi(z))$ look like independent random variables, which is important because correlations between the functions $\psi(z)$ and $v(z)$ can result in undesirable loops in the walk.

Let θ be the desired distinguished point probability. We declare an element z to be distinguished iff $H(z) \equiv 0 \mod \lfloor 1/\theta \rceil$, where $\lfloor \cdot \rceil$ is the rounding to the nearest integer. Although Algorithm $\mathcal{A}$ has polynomial memory requirements, we find it practical to use an $O(n^{1/4})$ amount of storage[6], namely to choose

$$\theta = n^{-\frac{1}{4}}.$$

This is compatible with the work of Schulte-Geers [97]. The database of distinguished nodes is implemented as a binary tree.

For the starting randomization of walks we use the 64-bit Mersenne twister pseudorandom generator [76]. A pseudorandom element $g_r \in G$ acts on the initial node to create the starting point of the new walk.

4.5.3 Choosing the Number of Experiments

Let k be the number of experiments and L_k the average value of L over k experiments. According to the central limit theorem [102, §7.2.1], the probability distribution of the random variable L_k approaches the normal distribution with the mean $\mathrm{E}(L)$ and the variance $\mathrm{Var}(L)/k$ as k approaches infinity. For the normal distribution, over 99.7 % of the values lie within three standard deviations away from the mean. Thus, assuming k is big enough, we have that

$$\Pr\left[L_k - 3\frac{\mathrm{Stdev}(L)}{\sqrt{k}} \le E(L) \le L_k + 3\frac{\mathrm{Stdev}(L)}{\sqrt{k}}\right] > 0.997.$$

When measuring $\mathrm{E}(L)$, we use two levels of accuracy: the result lies within ± 0.1 % of the true value for the experiments satisfying $\log(n) \le 44$, and within ± 0.5 % of the true value otherwise. Thus we can use the inequalities

$$k_1 \ge \left(\frac{3\,\mathrm{Stdev}(L)}{0.001\,\mathrm{E}(L)}\right)^2, \qquad k_2 \ge \left(\frac{3\,\mathrm{Stdev}(L)}{0.005\,\mathrm{E}(L)}\right)^2 \tag{4.7}$$

[6]Let us justify the suitability of this choice by an example. Suppose one tries to solve a $\mathcal{CL}$-GAIP over a 244-bit field, a problem size proposed for isogeny-based cryptosystems [107]. Since the group size (i.e., class number) $n \approx 2^{122}$, the database of distinguished nodes should store $L\theta\sqrt{n}$ nodes, which is less than 2^{33} on average. Since the class number is approximately 122 bits long, one entry of the database (binary tree) of distinguished nodes would occupy 48 bytes, of which 16 bytes are used by a hashed j-invariant, 16 bytes by a compressed class group element and 16 bytes by two pointers. The whole database would occupy not more than 384 gigabytes of disk space, which we find to be quite moderate.

to find the sufficient number of experiments for the two accuracy levels. For a preliminary estimation of the number of experiments we use the formulae for $\mathrm{E}(L)$ and $\mathrm{Stdev}(L)$ obtained in Section 4.4.4. This gives us the values

$$k_1 = 2459137, \qquad k_2 = 98368\ ,$$

computed as maximums over all possible parameters in Experiment 4.1.

Our experiments have shown that, in most cases, both the sample mean and the sample standard deviation differ from the results of Theorem 4.1 by approximately the same factor, which cancels out in (4.7). This means that the obtained numbers k_1 and k_2 fit for the probability distributions under observation.

4.5.4 Experimental Measurement of L

In this section we measure $\mathrm{E}(L)$ by means of experimentation and assemble results in a table so that they can be used for arbitrary GAIP instances in the future.

Experiment 4.1 (Measuring L in Arbitrary Groups)**.** For each of the values[7] $\lceil \log(n) \rceil \in \{28, 32, 36, \ldots, 56\}$, $r \in \{3, 4, \ldots, 16\}$ and $w \in \{1,\ 3/4,\ 1/2,\ 1/3,\ 1/4\}$ conduct a set of k_1 (k_2 for $n > 2^{44}$) experiments. In each experiment choose a random[8] group G and a random subset of r elements that generates G. Use $\theta = n^{-1/4}$ and the partitioning probabilities decreasing with ratio w.

A subset of results is listed in Table 4.2, where mantissas are rounded to four decimal digits. Full data for $3 \le r \le 16$ and $w \in \{1, 3/4, 1/2, 1/3, 1/4\}$ are available in [108]. The entire experiment took 51 days of parallel processing on 128 cores.

When $w = 1$ and $r = 16$ one sees good agreement between Table 4.2 and Table 4.1, which suggests that our implementation is working well. In other cases we see that L is significantly larger than L_π, which shows that the theoretical analysis is over-optimistic about the behaviour of these pseudorandom walks. The results also confirm that $r = 3$ is not a good choice in practice.

Figure 4.2 graphs some values of the practice-to-theory ratio

$$\sigma = \frac{\mathrm{E}(L)}{\mathrm{E}(L_\pi)}.$$

Round dots depict our experimental results, and lines are their approximating functions (solid lines are $w = 1$, short-dashed lines are $w = 1/2$ and long-dashed lines are $w = 1/4$). For a fixed w, values of σ for $5 < r < 16$ lie between $r = 5$ and $r = 16$. One can observe an increased roughness of experimental results for $n > 2^{44}$

[7] We use $n > 2^{27}$ because otherwise L is highly affected by looped walks: every loop increases the number of visited nodes by $30n^{1/4}$.

[8] For each $m \in \{28, 32, 36, \ldots, 56\}$ we sample uniformly from the set of isomorphism classes of abelian groups of order n and rank at most r, where $2^{m-1} + 1 \le n \le 2^m$.

		$\lceil \log(n) \rceil$							
w	r	28	32	36	40	44	48	52	56
1	3	2.8547	2.9982	3.1380	3.2735	3.4079	3.5355	3.6681	3.7812
	4	2.2923	2.3101	2.3247	2.3371	2.3484	2.3518	2.3661	2.3661
	5	2.1039	2.0975	2.0968	2.0984	2.0978	2.1007	2.1009	2.1004
	6	2.0178	2.0099	2.0052	2.0032	2.0026	2.0038	2.0022	2.0023
	10	1.9021	1.8932	1.8862	1.8849	1.8831	1.8816	1.8769	1.8879
	16	1.8575	1.8455	1.8407	1.8384	1.8361	1.8302	1.8369	1.8357
$\frac{1}{2}$	3	3.1089	3.2761	3.4406	3.5985	3.7500	3.9086	4.0331	4.1785
	4	2.6071	2.6436	2.6723	2.6938	2.7101	2.7307	2.7315	2.7406
	5	2.4586	2.4665	2.4723	2.4782	2.4802	2.4875	2.4821	2.4776
	6	2.4000	2.4022	2.4063	2.4068	2.4086	2.4079	2.4069	2.4128
	10	2.3529	2.3536	2.3527	2.3553	2.3563	2.3486	2.3533	2.3557
	16	2.3516	2.3523	2.3519	2.3519	2.3536	2.3524	2.3465	2.3576
$\frac{1}{4}$	3	3.8596	4.1194	4.3652	4.5978	4.8213	5.0395	5.2484	5.4338
	4	3.5425	3.6694	3.7582	3.8280	3.8771	3.9015	3.9372	3.9517
	5	3.4753	3.5732	3.6423	3.6830	3.7103	3.7322	3.7295	3.7407
	6	3.4608	3.5566	3.6145	3.6526	3.6743	3.6985	3.6845	3.6845
	10	3.4578	3.5486	3.6064	3.6454	3.6658	3.6672	3.6853	3.6833
	16	3.4607	3.5498	3.6070	3.6427	3.6639	3.6747	3.6808	3.6880

Table 4.2: Expected values of L obtained experimentally for certain choices of r and w.

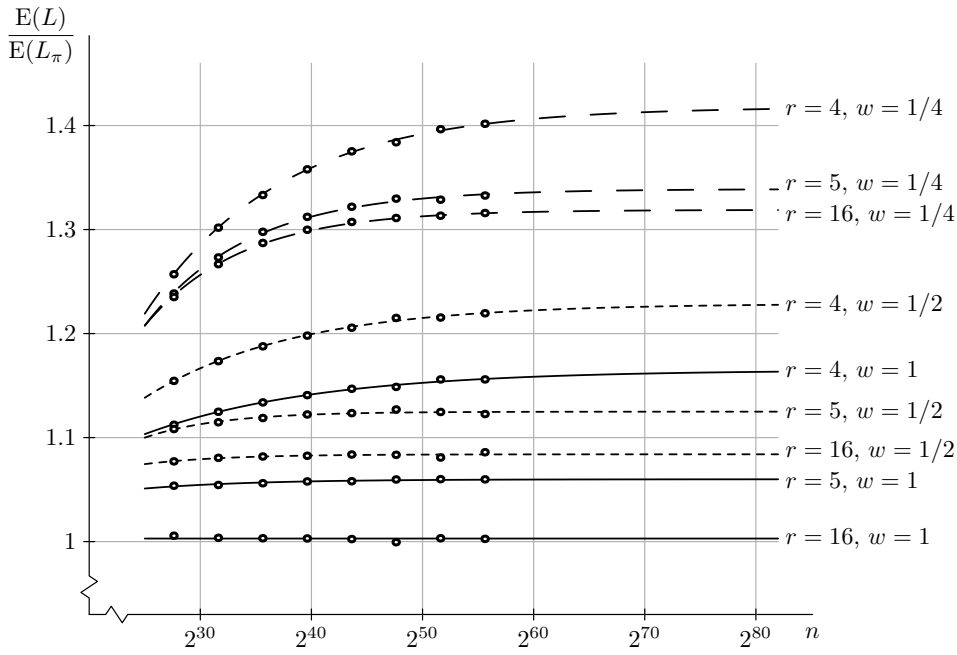

Figure 4.2: Values of $\sigma = \mathrm{E}(L)/\,\mathrm{E}(L_\pi)$ obtained experimentally and their approximations extended to $n = 2^{80}$.

due to the increased confidence interval. The graphs suggest that, for $r > 3$, the difference between $\mathrm{E}(L)$ and $\mathrm{E}(L_\pi)$ is fairly stable as n grows. Hence, when $r > 3$ we feel confident extrapolating actual values for $\mathrm{E}(L)$ from our formulae for $\mathrm{E}(L_\pi)$ and the experimentally determined correction factors σ.

Remark 4.1. Recently Montenegro proposed a heuristic for estimating the number of hops in birthday attacks [80]. His idea is to estimate the probability of short cycles, i.e. if two walks (with independent partitioning functions) are started from the same position, then what is the probability that they intersect soon? The lower this probability is, the sooner the algorithm will terminate. Applied to adding walks in an abelian group, this means that if two walks include short subsequences of edges which are equivalent up to the order of edges, these subsequences do not change the relative position of these walks. Although Montenegro only gives examples for Pollard's and Teske's walks, his heuristic also applies to walks with uneven partitioning. The probability P_1 that two independent walks started from $x_0 = y_0$ have a collision after one hop equals

$$P_1 = \Pr\left[x_1 = y_1\right] = \sum_{i=1}^{r} p_i^2.$$

If we only consider collisions after one hop, then Montenegro's heuristic gives an approximation similar to what we obtained in Theorem 4.1:

$$\mathrm{E}(\lambda) \approx \sqrt{\frac{\pi n}{1 - P_1}}.$$

The probability P_2 that a collision occurs on the second hop is

$$P_2 = \Pr\left[(x_1 \neq y_1) \wedge (x_2 = y_2)\right] = (1 - P_1)P_1^2,$$

and Montenegro's heuristic gives

$$\mathrm{E}(\lambda) \approx \sqrt{\frac{\pi n}{1 - (P_1 + P_2)}} = \sqrt{\frac{\pi n}{1 - P_1 - P_1^2 + P_1^3}}. \tag{4.8}$$

The calculation can be continued to more hops, but since probabilities of collisions become much smaller than P_2, this will result in very small numerical changes.

We have calculated the expected values of L using (4.8) and found that for $r \geq 6$ the heuristic agrees pretty well with our practical results, giving only up to 3.4 % error for $w = 1/2$ and up to 5.6 % error for $w = 1/4$.

4.6 The Algorithm in Practice

We now discuss how the isogeny algorithm performs in practice. We focus on the case of ideal class groups of maximal orders in CM fields coming from $\mathrm{End}(E)$ where E is a randomly chosen elliptic curve over $\mathbb{F}_p$ and p is a randomly chosen 160-bit prime. We also speculate on how the algorithm will perform for larger fields at the end of this section.

We have already obtained a good theoretical and experimental understanding of the algorithm for the group action problem. It is necessary now to include the cost of computing isogenies. The next section gives some estimates for the running time of computing isogenies of prime degree.

4.6.1 Cost of Computing Isogenies

Consider the cost of computing the action by a prime ideal in the isogeny graph. One has an elliptic curve and an ideal of norm ℓ. One must factor the modular polynomial to determine the possible j-invariants of ℓ-isogenous curves, one must perform Elkies' algorithm to determine the kernel polynomials for these isogenies, and then one must use the technique from [45] to determine which is the correct kernel and hence which is the correct isogeny[9]. It is not necessary to apply Vélu's formulae at this stage.

[9]If two or more consecutive hops are made by the same split isogeny degree ℓ, and there are no vertical ℓ-isogenies, then it is sufficient to choose the correct isogeny only at the first hop. On each subsequent hop one simply checks that the j-invariant does not match the previous one. This provides extra saving, especially when the partitioning is uneven. This extra saving is not accounted in Table 4.3.

ℓ	Time, s	ℓ	Time, s
3	0.002870	61	0.168179
5	0.004799	67	0.225253
7	0.006898	71	0.225503
11	0.012113	73	0.249365
13	0.015261	79	0.270537
17	0.022376	83	0.284242
19	0.026499	89	0.305786
23	0.036209	97	0.341988
29	0.052346	101	0.353268
31	0.058230	103	0.362195
37	0.084434	107	0.375111
41	0.092049	109	0.384550
43	0.106742	113	0.403007
47	0.116152	127	0.467993
53	0.143732	131	0.579427
59	0.150925	137	0.624039

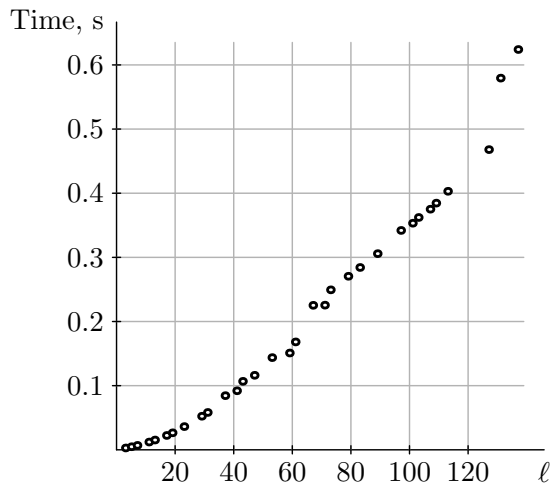

Figure 4.3: Average running time of one ℓ-isogeny (i.e., action by a prime ideal of norm ℓ) for elliptic curves over 160-bit prime fields.

We assume the modular polynomials have been precomputed and reduced to the finite field $\mathbb{F}_q$. Since the modular polynomial has $O(\ell^2)$ coefficients one performs $O(\ell^2)$ field operations to evaluate the modular polynomial at the target j-invariant. An expected $O(\ell \log(\ell) \log(q))$ field operations are performed to find the roots of the polynomial, employing fast polynomial arithmetic. Finally, $O(\ell^2)$ field operations are used by Elkies' algorithm. Hence one expects the time of one ℓ-hop to grow like

$$O\left(\ell^2 + \ell \log(\ell) \log(q)\right) \tag{4.9}$$

field operations.

We computed average timings using the ClassEll package by Stolbunov [104]. The package implements the ideal class group action on sets of ordinary elliptic curves. The experiment was run on Intel X5550 processors clocked at 2.67 GHz, the code executed at approximately 6799 millions instructions per second (MIPS). The data was gathered by repeatedly (20000 times) generating a random 160-bit prime p and a random ordinary elliptic curve over $\mathbb{F}_p$ with a fundamental Frobenius discriminant. The time spent on one action by a prime ideal, for prime ideals of all split norms less than or equal to 137, was recorded. To increase the accuracy, we performed more hops for smaller primes. Results are given in Fig. 4.3. We can observe bumps when ℓ moves over degrees of two which is typical for the polynomial multiplication by number-theoretic transform.

4.6.2 Ideal Class Groups

In Experiment 4.1 we used the uniform distribution of finite abelian groups. However, the structure of ideal class groups is not that random; the following observations are known as Cohen-Lenstra heuristics [27]: the odd part of the class group of an imaginary quadratic field is quite rarely non-cyclic; if p is a small odd prime, the proportion of imaginary quadratic fields whose class number is divisible by p is close to $1/p + 1/p^2$. The distribution of group structures in the isogeny problem is further affected by the fact that the imaginary quadratic orders are chosen as endomorphism rings of random elliptic curves. Nevertheless, our experiments show that the difference between values $\mathrm{E}(L)$ for random isogeny problem instances[10] and for random GAIP instances lies within the margin of error 0.2 %. The same holds for the standard deviation of L.

Due to the numerical results of Jacobson, Ramachandran and Williams [59] we know that the average maximum norm of the prime ideals required to generate the class group of $\mathbb{Q}(\sqrt{\Delta})$ for $-10^{11} < \Delta < 0$ approximately equals $0.60191 \ln|\Delta|$, and the number of prime ideals required to generate these class groups averages at approximately 3.3136. We assume that these results apply to our problem size as well. Hence for a random ideal class group of a 162-bit discriminant, it is very likely that a generating set of four prime ideals with the maximum norm 67 can be found. This observation is used in the next section where we model the choice of primes.

We make an assumption that walks with a supporting set that consists of ideals of small prime norm behave similar to walks when the supporting set consists of random group elements.

4.6.3 Predicted Results

In this section we estimate the time needed for solving a random instance of the isogeny problem over a 160-bit finite field using various numbers of partitions r and partitioning probabilities $\vec{p}$. The expected serial running time is computed using equation (4.1), which can be written as

$$\sigma \ \mathrm{E}(L_\pi) \sqrt{n}\, \vec{p}\vec{t}.$$

The values $\mathrm{E}(L_\pi)$ are computed using Theorem 4.1 and approximations for σ are based on our experimental data (partially displayed in Fig. 4.2). We take $n = 2^{80}$. What remains is to compute the average running time $\vec{p}\vec{t}$ of one hop.

For the isogeny problem, the supporting set H should be chosen to consist of prime ideals above the smallest integer primes which split in $\mathcal{O}$. If necessary, one or more prime ideals of larger norm are included in H to ensure that H generates $\mathcal{CL}(\mathcal{O})$. To compute the average product $\vec{p}\vec{t}$ for given r and w, we enumerate all subsets H of r primes larger or equal to 3 with the $r-4$ smallest primes in H being

[10] Parameters: $\lceil \log(p) \rceil = 90$, $4 \le r \le 16$; w, θ, $c_{\max}$ and k_1 are as in Experiment 4.1.

r\w	1	3/4	1/2	1/3	1/4
4	8708	6940	5429	4727	4690
5	6455	4495	2758	1925	1652
6	5514	3396	1755	1130	988
7	5068	2827	1334	904	858
8	4891	2530	1154	847	848
9	4930	2415	1093	**842**	856
10	5549	2548	1110	858	870
11	6391	2723	1132	874	885
12	7409	2915	1157	891	903
13	8485	3095	1180	906	919
14	9519	3255	1205	923	932
15	10636	3396	1225	937	944
16	**12200**	3541	1242	949	955

Table 4.3: Expected serial time (years) needed to solve a random $\mathcal{CL}$-GAIP over a 160-bit field.

less than or equal to[11] prime_{2r-7}, and the largest prime in H lying between 67 and $\max(67, \text{prime}_{2r+1})$. For every set H, a timing vector $\vec{t}$ is constructed using the data in Fig. 4.3. Hence we compute the average $\vec{p}\,\vec{t}$ over all H.

In Table 4.3 we give estimated times for solving a random instance of the isogeny problem over a 160-bit finite field (equivalently, the $\mathcal{CL}$-GAIP problem in $\mathcal{CL}(\mathcal{O})$ where $\mathcal{O} = \text{End}(E)$ for an elliptic curve over a 160-bit finite field). The time is provided in years of serial execution on one Intel X5550 2.67 GHz CPU core. On a cluster with hundreds of thousands of cores the problem can be solved in a matter of hours.

We see from Table 4.3 that the best combination $r = 9$ and $w = 1/3$ is approximately 14 times faster than 16 equally-sized partitions (both timings are in bold). In fact all values within $7 \leq r \leq 16$, $w \in \{1/3, 1/4\}$ provide good speeds.

For the rest of the section we briefly consider the question of how much faster our algorithm is than the GHS algorithm as $q \to \infty$. Both algorithms require $\tilde{O}(\sqrt{n})$ bit operations, but it is not immediately clear that the ratio of running times is bounded as $q \to \infty$. Let us compare $r = 16$, $w = 1$ with $r = 9$, $w = 1/3$. First we make a simplifying assumption: for any problem instance, a supporting set H consisting of the $r-1$ smallest split primes and one prime close to $\ln(q)$, generates the class group. Using the prime number theorem we approximate primes in H by $\ell_i \approx 2i\ln(2i)$, for $1 \leq i \leq r-1$. We also approximate $\ell_r \approx \ln(q)$. Since $\ell_i < \log(q)$ for sufficiently large q, the complexity (4.9) of one ℓ-hop is $O(\ell \ln(\ell) \ln(q))$ field operations, which we further approximate by $c\,\ell \ln(q)$ for some constant c. The improvement ratio

[11] Because approximately half of primes are split.

(i.e., speedup) is

$$\frac{\big(\mathrm{E}(L)\,\vec{p}\,\vec{t}\big)|_{\substack{r=16\\w=1}}}{\big(\mathrm{E}(L)\,\vec{p}\,\vec{t}\big)|_{\substack{r=9\\w=1/3}}} \approx \frac{1.836}{3.023}\,\frac{\frac{1}{16}\sum_{i=1}^{15} 2i\ln(2i)\ln(q)+\frac{1}{16}\ln^2(q)}{\frac{6561}{9841}\sum_{i=1}^{8}\left(\frac{1}{3}\right)^{i-1} 2i\ln(2i)\ln(q)+\frac{1}{9841}\ln^2(q)} \approx$$

$$0.607\,\frac{44.046+\frac{1}{16}\ln(q)}{3.682+\frac{1}{9841}\ln(q)} \to 0.607\,\frac{9841}{16} \approx 373 \qquad \text{as } q\to\infty.$$

Hence the improvement ratio slowly grows with q and stabilizes at few hundreds for a very large q (at $\ln(q) > 2^{25}$ in the example above). Sure, problems of that size are not feasible, and 9 primes are probably not sufficient to generate a class group that big. The growth of the improvement ratio is hard to predict, but we see no reasons for it to overcome $O(1)$ as $q \to \infty$.

4.7 Conclusion

In this paper we have improved the GHS algorithm for constructing isogenies between ordinary elliptic curves. Our improvement is by an $O(1)$ factor, which was estimated to be approximately 14 for random 160-bit elliptic curves with comparable conductors. This is a significant acceleration. Nevertheless, the asymptotic complexity of the $\mathbb{F}_q$-isogeny problem for curves with comparable conductors is $O(q^{1/4+o(1)}\log^2(q)\log(\log(q)))$ field operations, as before.

Acknowledgements

The paper was created through a collaboration of two authors whose names are listed alphabetically. The work was initiated during a two-month research visit of Anton Stolbunov to Steven Galbraith. Stolbunov would like to thank Department of Telematics, Norwegian University of Science and Technology, for the financial support of his research and that visit. We thank Gaetan Bisson and Edlyn Teske for their valuable comments on this paper.

4.A Proof of Theorem 4.1

4.A.1 Random Variables α, λ and δ

Let us introduce the following notation for a *completed run* of the algorithm $\mathcal{A}_\pi$. We avoid using indices π inside the proof, keeping in mind that all results apply to the algorithm $\mathcal{A}_\pi$:

α is the number of visited nodes, counted with repetition;

λ is the number of distinct visited nodes minus one[12];

δ is the number of nodes visited more than once, such that a node visited twice is counted once, a node visited three times is counted twice, and so on.

All distinct nodes visited by the algorithm, except for the problem instance nodes x_0 and x_1, are counted in the variable λ. After every collision the thread continues its walk unless it hits a distinguished node. Since the walk is deterministic, the sequence of nodes starting from the collision node coincides with some previous walk. These nodes are counted in the variable δ.

We have that

$$\begin{aligned} \alpha &= \lambda + \delta + 1, \\ \mathrm{E}(\alpha) &= \mathrm{E}(\lambda) + \mathrm{E}(\delta) + 1, \end{aligned} \tag{4.10}$$

which is a standard result in statistics [102, §4.4]. To complete the proof we will calculate the summands in the latter equation.

4.A.2 Expected Number of Duplicate Visited Nodes δ

Chasing the good-collision distinguished node can be described as a sequence of Bernoulli trials (random experiments with two possible outcomes: success or failure) with success probability $q\theta$, where $q = \frac{1}{2}$ is the probability that a randomly chosen collision is good[13]. The number of trials δ needed to get one success is described by the geometric distribution [102, §6.1.2]. Hence the probability mass function (PMF) of δ is

$$f_\delta(x) = \frac{\theta}{2}\left(1 - \frac{\theta}{2}\right)^{x-1}, \quad \text{where } x \geq 1, \tag{4.11}$$

and the expected value is

$$\mathrm{E}(\delta) = \frac{2}{\theta}.$$

4.A.3 Expected Number of Distinct Visited Nodes λ

We consider a graph with the set of vertices X and the edges $(z, \psi_\pi(z))$.

Lemma 4.3. *For a node visited by the algorithm $\mathcal{A}_\pi$, the expected in-degree minus the number of used incoming edges equals*

$$d = 1 - (1 - \theta)\sum_{i=1}^{r} p_i^2.$$

[12]This simplifies formulae in Lemma 4.4.

[13]In the algorithm $\mathcal{A}_\pi$ the number of walks that start from x_0 equals the number of walks that start from x_1. Hence half of the collisions are good, namely $q = \frac{1}{2}$.

Proof. We consider two types of nodes: randomized walk-starting nodes and nodes obtained by one or more hops. First we compute the expected in-degree of the starting nodes.

Let z be a random node on the graph. We show that the expected in-degree of z equals one. Indeed, every node has exactly one outgoing edge, therefore the graph has n edges. Hence

$$\mathrm{E}\left(\operatorname{indeg} z \mid z \xleftarrow{R} X\right) = \frac{1}{n}\sum_{i=1}^{n} \operatorname{indeg} z_i = 1.$$

The probability that a visited node is a walk-starting node equals θ.

Now we compute the expected in-degree of nodes obtained by one or more hops. Let z_0 be a random node such that $v(z_0) = i$ for some partition number i. By making a hop from z_0 we obtain a node z_1. The expected number of edges coming into z_1 from $X \setminus \{z_0\}$ equals

$$\mathrm{E}\left(\operatorname{indeg} z_1 - 1 \mid z_1 = \psi_\pi(z_0),\, z_0 \xleftarrow{R} X,\, v(z_0) = i\right) = \sum_{\substack{1\le j\le r\\ j\ne i}} p_j.$$

When $v(z_0)$ is not given, the expected number of edges coming into z_1 from $X \setminus \{z_0\}$ equals

$$\mathrm{E}\left(\operatorname{indeg} z_1 - 1 \mid z_1 = \psi_\pi(z_0),\, z_0 \xleftarrow{R} X\right) = \sum_{i=1}^{r} p_i \sum_{\substack{1\le j\le r\\ j\ne i}} p_j = \sum_{\substack{1\le i,j\le r\\ i\ne j}} p_i p_j = 1 - \sum_{i=1}^{r} p_i^2.$$

Finally, by assembling results for the two types of nodes, we compute

$$d = \theta + (1-\theta)\left(1 - \sum_{i=1}^{r} p_i^2\right) = 1 - (1-\theta)\sum_{i=1}^{r} p_i^2.$$

□

Example values of d are given in Table 4.1. In the rest of our proof we will use the facts that $d = O(1)$ and $\frac{1}{d} = O(1)$.

Recall that λ is the number of distinct visited nodes minus one. After a number of walks have been made without collisions, all nodes visited by these walks are counted in λ. When a new walk hits a collision, the part of this walk before the collision node is counted in λ, and the part after the collision node is counted in δ.

Lemma 4.4. *The PMF of λ equals*

$$f_\lambda(x) = \left(\frac{xqd}{n} + \frac{x^2q^2d(d+2)}{2n^2} - \frac{x^4q^3d^2(d+2)}{6n^3}\right) e^{-\frac{x^2qd}{2n}} + O\left(\frac{\ln^{7/2} n}{n^{3/2}}\right), \qquad (4.12)$$

where $0 \le x = O(\sqrt{n \ln n})$.

Proof. We note that when the walk is a random mapping on the set X and all collisions are good, an approximation for $f_\lambda(x)$ has been obtained by Harris [54]. For a random mapping with the proportion of good collisions q, an approximation for $f_\lambda(x)$ has been obtained by van Oorschot and Wiener [115].

Denote by E_i the event that a good collision occurs when $\lambda = i$. We want to compute the probability that no good collisions occur while $1 \le \lambda < x$, and a good collision occurs at $\lambda = x$, thus we can write

$$\begin{aligned} f_\lambda(x) = \Pr\left[\, E_x \wedge \overline{E_{x-1}} \wedge \cdots \wedge \overline{E_1} \,\right] = \\ \Pr\left[\, E_x \mid \overline{E_{x-1}} \wedge \cdots \wedge \overline{E_1} \,\right] \cdot \Pr\left[\, \overline{E_{x-1}} \wedge \cdots \wedge \overline{E_1} \,\right] = \\ \Pr\left[\, E_x \mid \overline{E_{x-1}} \wedge \cdots \wedge \overline{E_1} \,\right] \cdot \Pr\left[\, \overline{E_{x-1}} \mid \overline{E_{x-2}} \wedge \cdots \wedge \overline{E_1} \,\right] \cdot \Pr\left[\, \overline{E_{x-2}} \wedge \cdots \wedge \overline{E_1} \,\right] = \\ \cdots = \Pr\left[\, E_x \mid \overline{E_{x-1}} \wedge \cdots \wedge \overline{E_1} \,\right] \cdot \Pr\left[\, \overline{E_1} \,\right] \cdot \prod_{i=2}^{x-1} \Pr\left[\, \overline{E_i} \mid \overline{E_{i-1}} \wedge \cdots \wedge \overline{E_1} \,\right]. \end{aligned}$$

Although several walks run in parallel, at any moment there is a certain set Λ of visited nodes and a particular walk which makes the next hop. Denote by K the subset of nodes in $X \setminus \Lambda$ out-edges from which lead into Λ producing a good collision. We are interested in the probability that the walk lands in K, because the next hop of this walk is a good collision. Given the events $\overline{E_{i-1}} \wedge \cdots \wedge \overline{E_1}$, we know that all incoming good-collision edges originate outside of Λ, hence the expected size of K equals $qd\#\Lambda$. Given $\overline{E_{i-1}} \wedge \cdots \wedge \overline{E_1}$, we also know that the previous hop of the walk does not yield a good collision, i.e. the walk does not land in a part of Λ of the expected size qi. Hence[14] $\Pr[\, E_i \mid \overline{E_{i-1}} \wedge \cdots \wedge \overline{E_1} \,] = \frac{iqd}{n-qi}$. For $2 \le i < x = O(\sqrt{n \ln n})$ we have that

$$\begin{aligned} \Pr\left[\, \overline{E_i} \mid \overline{E_{i-1}} \wedge \cdots \wedge \overline{E_1} \,\right] = 1 - \frac{iqd}{n-qi} = \left(1 - \frac{iqd}{n}\right) - \frac{i^2q^2d}{n^2} - \frac{i^3q^3d}{n^2(n-qi)} = \\ \left(e^{-\frac{iqd}{n}} - \frac{(-iqd)^2}{2!n^2} - \frac{(-iqd)^3}{3!n^3} - \cdots\right) - \frac{i^2q^2d}{n^2} + O\left(\frac{\ln^{3/2} n}{n^{3/2}}\right) = \\ e^{-\frac{iqd}{n}} - \frac{i^2q^2d(d+2)}{2n^2} + O\left(\frac{\ln^{3/2} n}{n^{3/2}}\right). \end{aligned}$$

Computing the product

$$\prod_{i=2}^{x-1} \Pr\left[\, \overline{E_i} \mid \overline{E_{i-1}} \wedge \cdots \wedge \overline{E_1} \,\right] = \prod_{i=2}^{x-1} \left[e^{-\frac{iqd}{n}} - \frac{i^2q^2d(d+2)}{2n^2} + O\left(\frac{\ln^{3/2} n}{n^{3/2}}\right)\right] \quad (4.13)$$

[14]The node in K is not counted in $\lambda = i$, because λ is the number of distinct visited nodes minus one. After the walk has landed in K, hops made by all other walks are not counted, in order to simplify the proof.

Number of factors of the form				
$e^{-\frac{iqd}{n}}$	$\frac{i^2q^2d(d+2)}{2n^2}$	$O\left(\frac{\ln^{3/2} n}{n^{3/2}}\right)$	Number of summands	Sum
$x-2-a-b$	a	b	$\binom{x-2}{a+b}$	$O\left(\left(\frac{\ln^{3a+4b} n}{n^{a+2b}}\right)^{1/2}\right)$
$x-2$	0	0	1	$\left(1+\frac{xqd}{2n}\right)e^{-\frac{x^2qd}{2n}}+O\left(\frac{\ln n}{n}\right)$
$x-3$	1	0	$x-2$	$-\frac{x^3q^2d(d+2)}{6n^2}e^{-\frac{x^2qd}{2n}}+O\left(\frac{\ln^2 n}{n}\right)$
$x-3$	0	1	$x-2$	$O\left(\frac{\ln^2 n}{n}\right)$
$x-4$	2	0	$\binom{x-2}{2}$	$O\left(\frac{\ln^3 n}{n}\right)$
$x-4$	1	1	$\binom{x-2}{2}$	$O\left(\frac{\ln^{7/2} n}{n^{3/2}}\right)$
$x-5$	3	0	$\binom{x-2}{3}$	$O\left(\frac{\ln^{9/2} n}{n^{3/2}}\right)$
$x-4$	0	2	$\binom{x-2}{2}$	$O\left(\frac{\ln^4 n}{n^2}\right)$
$x-5$	2	1	$\binom{x-2}{3}$	$O\left(\frac{\ln^5 n}{n^2}\right)$
$x-6$	4	0	$\binom{x-2}{4}$	$O\left(\frac{\ln^6 n}{n^2}\right)$
each of the other categories				$O\left(\frac{\ln^{15/2} n}{n^{5/2}}\right)$

Table 4.4: Categories of summands in (4.13).

will require some bookkeeping. Each factor has three summands, and hence the result is a sum of 3^{x-2} terms, each term being a product of $x-2$ factors. We will separate the summands depending on the number of factors of the form $\frac{i^2q^2d(d+2)}{2n^2}$ and of the form $O\left(\frac{\ln^{3/2} n}{n^{3/2}}\right)$ they contain. Table 4.4 categorises the summands according to this property.

The row $(x-2-a-b, a, b)$ in Table 4.4 shows a general formula for the growth rate of the sum of a particular category. The sum contains $\binom{x-2}{a+b} = O(x^{a+b})$ terms, and hence it is

$$O\left(x^{a+b}\left(\frac{x}{n}\right)^{2a}\left(\frac{\ln n}{n}\right)^{\frac{3b}{2}}\right) = O\left(\left(\frac{\ln^{3a+4b} n}{n^{a+2b}}\right)^{1/2}\right),$$

because $\frac{i^2q^2d(d+2)}{2n^2} = O(\frac{\ln n}{n})$ and $0 < e^a \leq 1$ for any $a \leq 0$. Observe that the growth rate exceeds $O\left(\frac{1}{n^2}\right)$ only for those categories where $a+2b \leq 4$. These nine categories are listed in the table.

In the row $(x-2,0,0)$ of Table 4.4 the product is computed as

$$\prod_{i=2}^{x-1} e^{-\frac{iqd}{n}} = e^{-\sum_{i=2}^{x-1}\frac{iqd}{n}} = e^{-\frac{(x-2)(x+1)qd}{2n}} = e^{-\frac{(x^2-x)qd}{2n}}\left(1+O\left(\frac{1}{n}\right)\right) = e^{-\frac{(x^2-x)qd}{2n}} + O\left(\frac{1}{n}\right) = e^{-\frac{x^2qd}{2n}}\left(1+\frac{xqd}{2n}+O\left(\frac{x^2}{n^2}\right)\right)+O\left(\frac{1}{n}\right) = \left(1+\frac{xqd}{2n}\right)e^{-\frac{x^2qd}{2n}}+O\left(\frac{\ln n}{n}\right).$$

In the row $(x-3,1,0)$ of Table 4.4 the sum is computed as

$$\sum_{i=2}^{x-1}\left(\frac{i^2q^2d(d+2)}{2n^2}\prod_{j\in\{2,\dots,x-1\}\setminus\{i\}} e^{-\frac{jqd}{n}}\right) = \frac{q^2d(d+2)}{2n^2}\sum_{i=2}^{x-1}\left(i^2e^{\frac{iqd}{n}}\prod_{j=2}^{x-1}e^{-\frac{jqd}{n}}\right) = \frac{q^2d(d+2)}{2n^2}e^{-\frac{(x-2)(x+1)qd}{2n}}\sum_{i=2}^{x-1}\left(i^2\left(1+O\left(\frac{i}{n}\right)\right)\right) = \frac{q^2d(d+2)}{2n^2}e^{-\frac{(x^2-x-2)qd}{2n}}\left(1+O\left(\frac{x}{n}\right)\right)\sum_{i=2}^{x-1}i^2 = \frac{x(x-1)(2x-1)q^2d(d+2)}{12n^2}e^{-\frac{(x^2-x-2)qd}{2n}}+O\left(\frac{\ln^2 n}{n}\right) = \frac{x^3q^2d(d+2)}{6n^2}e^{-\frac{(x^2-x-2)qd}{2n}}+O\left(\frac{\ln^2 n}{n}\right) = \frac{x^3q^2d(d+2)}{6n^2}e^{-\frac{x^2qd}{2n}}\left(1+O\left(\frac{x}{n}\right)\right)+O\left(\frac{\ln^2 n}{n}\right) = \frac{x^3q^2d(d+2)}{6n^2}e^{-\frac{x^2qd}{2n}}+O\left(\frac{\ln^2 n}{n}\right).$$

For the sum of squares we have used the formula $\sum_{i=1}^{a} i^2 = \frac{1}{6}a(a+1)(2a+1)$ [87, Eq. 4.1.1.8].

There are $\sum_{i=0}^{x-2}(i+1) = \frac{1}{2}x(x-1) = O(x^2)$ categories of summands in the product (4.13). Hence the categories contained in the last row of Table 4.4 sum altogether to $O\left(x^2\frac{\ln^{15/2}n}{n^{5/2}}\right) = O\left(\frac{\ln^{17/2}n}{n^{3/2}}\right)$. We have shown that

$$\prod_{i=2}^{x-1}\Pr\left[\,\overline{E_i}\mid\overline{E_{i-1}}\wedge\cdots\wedge\overline{E_1}\,\right] = \left(1+\frac{xqd}{2n}-\frac{x^3q^2d(d+2)}{6n^2}\right)e^{-\frac{x^2qd}{2n}}+O\left(\frac{\ln^3 n}{n}\right).$$

Multiplying this value by $\Pr[\,\overline{E_1}\,] = 1+O\left(\frac{1}{n}\right)$ does not change the formula.

Thus the conditional probability of E_x is

$$\Pr\left[\, E_x \mid \overline{E_{x-1}} \wedge \cdots \wedge \overline{E_1} \,\right] = \frac{xqd}{n-qx} = \frac{xqd}{n} + \frac{x^2q^2d}{n^2} + \frac{x^3q^3d}{n^2(n-qx)} = \frac{xqd}{n} + \frac{x^2q^2d}{n^2} + O\left(\frac{\ln^{3/2} n}{n^{3/2}}\right).$$

Now we have all the factors to compute the PMF $f_\lambda(x)$:

$$\left[\frac{xqd}{n} + \frac{x^2q^2d}{n^2} + O\left(\frac{\ln^{3/2} n}{n^{3/2}}\right)\right]\left[\left(1 + \frac{xqd}{2n} - \frac{x^3q^2d(d+2)}{6n^2}\right)e^{-\frac{x^2qd}{2n}} + O\left(\frac{\ln^3 n}{n}\right)\right] = \left(\frac{xqd}{n} + \frac{x^2q^2d(d+2)}{2n^2} - \frac{x^4q^3d^2(d+2)}{6n^3}\right)e^{-\frac{x^2qd}{2n}} + O\left(\frac{\ln^{7/2} n}{n^{3/2}}\right).$$

□

We choose the value $x_0 = \sqrt{\frac{2n\ln n}{qd}}$ to be a border after which (i.e. on the interval $[x_0, n]$) the PMF of λ will be regarded as unknown. The expected value of λ is then the sum of an "essential part" $\sum_{x=0}^{\lceil x_0\rceil - 1} x f_\lambda(x)$ and an "error part" $\sum_{x=\lceil x_0\rceil}^{n} x f_\lambda(x)$. We first calculate the essential part of the expected value of λ.

Lemma 4.5. *For $x_0 = \sqrt{\frac{2n\ln n}{qd}}$ the following is true:*

$$\sum_{x=0}^{\lceil x_0\rceil - 1} x f_\lambda(x) = \sqrt{\frac{\pi n}{2qd}} - \frac{1}{3} - \frac{2}{3d} + O\left(\frac{\ln^{9/2} n}{n^{1/2}}\right).$$

Proof. Since $x_0 = O(\sqrt{n\ln n})$, we use Lemma 4.4 to calculate the sum

$$\sum_{x=0}^{\lceil x_0\rceil - 1} x f_\lambda(x) = \sum_{x=0}^{\lceil x_0\rceil - 1}\left(\frac{x^2qd}{n} + \frac{x^3q^2d(d+2)}{2n^2} - \frac{x^5q^3d^2(d+2)}{6n^3}\right)e^{-\frac{x^2qd}{2n}} + \sum_{x=0}^{\lceil x_0\rceil - 1} xO\left(\frac{\ln^{7/2} n}{n^{3/2}}\right) = \frac{qd}{n}\int_0^{x_0} x^2 e^{-\frac{x^2qd}{2n}}\partial x + \frac{q^2d(d+2)}{2n^2}\int_0^{x_0} x^3 e^{-\frac{x^2qd}{2n}}\partial x - \frac{q^3d^2(d+2)}{6n^3}\int_0^{x_0} x^5 e^{-\frac{x^2qd}{2n}}\partial x + O\left(\frac{\ln n}{n}\right) + \frac{\lceil x_0\rceil(\lceil x_0\rceil - 1)}{2}O\left(\frac{\ln^{7/2} n}{n^{3/2}}\right). \quad (4.14)$$

When computing sums by integration we have used the Euler-Maclaurin summation

formula [1, §3.6.28], which implies that

$$\sum_{x=0}^{\lceil x_0\rceil-1} x^b e^{-\frac{x^2qd}{2n}} = \int_0^{x_0} x^b e^{-\frac{x^2qd}{2n}}\partial x + \int_{x_0}^{\lceil x_0\rceil} x^b e^{-\frac{x^2qd}{2n}}\partial x + \\ + O\left(\max\left(\frac{\partial^i}{\partial x} x^b e^{-\frac{x^2qd}{2n}}\Big|_{\substack{i\geq 0\\ x\in\{0,\lceil x_0\rceil\}}}\right)\right) = \int_0^{x_0} x^b e^{-\frac{x^2qd}{2n}}\partial x + O(n^{\frac{b-2}{2}}\ln^{\frac{b}{2}} n) + O(1), \tag{4.15}$$

where $b\in\mathbb{N}$ and x_0 is defined above. Observe also that $e^{-\frac{x_0^2qd}{2n}} = \frac{1}{n}$.

Let us compute the exponential integrals in (4.14) separately. To the first integral we apply the formula $\int x^2 e^{-a^2x^2}\partial x = -\frac{x}{2a^2}e^{-a^2x^2} + \frac{\sqrt{\pi}}{4a^3}\operatorname{erf}(ax)$ [87, Eq. 1.3.3.8], where erf() is known as the error function: $\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}\partial t = 1 - \frac{2}{\sqrt{\pi}}\int_z^\infty e^{-t^2}\partial t$ [1, Eq. 7.1.2]. Note that $\operatorname{erf}(0) = 0$.

$$\frac{qd}{n}\int_0^{x_0} x^2 e^{-\frac{x^2qd}{2n}}\partial x = \frac{qd}{n}\left[-\frac{nx_0}{qd}e^{-\frac{x_0^2qd}{2n}} + \sqrt{\frac{\pi}{2}}\frac{n^{3/2}}{q^{3/2}d^{3/2}}\operatorname{erf}(\sqrt{\frac{qd}{2n}}x_0)\right] = \\ -\sqrt{\frac{2\ln n}{nqd}} + \sqrt{\frac{\pi n}{2qd}}\operatorname{erf}(\sqrt{\ln n}) = \sqrt{\frac{\pi n}{2qd}} - \sqrt{\frac{2n}{qd}}\int_{\sqrt{\ln n}}^\infty e^{-t^2}\partial t + O\left(\sqrt{\frac{\ln n}{n}}\right) = \\ \sqrt{\frac{\pi n}{2qd}} + O\left(\sqrt{\frac{\ln n}{n}}\right).$$

We have bounded the infinite integral using the inequality $e^{z^2}\int_z^\infty e^{-t^2}\partial t < \frac{1}{z}$ for $z\geq 0$ [1, Eq. 7.1.13].

To the second and third integrals in (4.14) we apply the formula $\int x^{2b+1}e^{-ax^2}\partial x = -\frac{x^{2b}}{2a}e^{-ax^2}\sum_{k=0}^b \frac{b!}{(b-k)!\,a^k x^{2k}}$ [87, §1.3.3]:

$$\frac{q^2d(d+2)}{2n^2}\int x^3 e^{-\frac{x^2qd}{2n}}\partial x = -\frac{q^2d(d+2)}{2n^2}\frac{x^2n}{qd}e^{-\frac{x^2qd}{2n}}\left(1+\frac{2n}{x^2qd}\right) = \\ -\left(\frac{x^2q(d+2)}{2n} + \frac{d+2}{d}\right)e^{-\frac{x^2qd}{2n}},$$

$$-\frac{q^3d^2(d+2)}{6n^3}\int x^5 e^{-\frac{x^2qd}{2n}} = \frac{q^3d^2(d+2)}{6n^3}\frac{x^4n}{qd}e^{-\frac{x^2qd}{2n}}\left(1+\frac{4n}{x^2qd}+\frac{8n^2}{x^4q^2d^2}\right) = \\ \left(\frac{x^4q^2d(d+2)}{6n^2} + \frac{2x^2q(d+2)}{3n} + \frac{4(d+2)}{3d}\right)e^{-\frac{x^2qd}{2n}}.$$

The corresponding definite integrals sum to

$$\frac{q^2d(d+2)}{2n^2}\int_0^{x_0} x^3e^{-\frac{x^2qd}{2n}}\partial x - \frac{q^3d^2(d+2)}{6n^3}\int_0^{x_0} x^5e^{-\frac{x^2qd}{2n}} =$$
$$\left(\frac{x_0^4q^2d(d+2)}{6n^2} + \frac{x_0^2q(d+2)}{6n} + \frac{d+2}{3d}\right)e^{-\frac{x_0^2qd}{2n}} - \frac{d+2}{3d} = -\frac{1}{3} - \frac{2}{3d} + O\left(\frac{\ln^2 n}{n}\right).$$

Assembling the calculated summands of (4.14) completes the proof. □

We now calculate an upper bound on the error part of $\mathrm{E}(\lambda)$.

Lemma 4.6. *For* $x_0 = \sqrt{\frac{2n\ln n}{qd}}$ *the following is true:*

$$\sum_{x=\lceil x_0\rceil}^{n} xf_\lambda(x) = O(\ln^4 n).$$

Proof. Using Lemma 4.4 and the facts that $x_0 = O(\sqrt{n\ln n})$ and $e^{-\frac{x_0^2qd}{2n}} = \frac{1}{n}$ we calculate the sum

$$\sum_{x=0}^{\lceil x_0\rceil-1} f_\lambda(x) = \sum_{x=0}^{\lceil x_0\rceil-1}\left(\frac{xqd}{n} + \frac{x^2q^2d(d+2)}{2n^2} - \frac{x^4q^3d^2(d+2)}{6n^3}\right)e^{-\frac{x^2qd}{2n}} +$$
$$+ \sum_{x=0}^{\lceil x_0\rceil-1} O\left(\frac{\ln^{7/2} n}{n^{3/2}}\right) = \frac{qd}{n}\int_0^{x_0} xe^{-\frac{x^2qd}{2n}}\partial x + \frac{q^2d(d+2)}{2n^2}\int_0^{x_0} x^2e^{-\frac{x^2qd}{2n}}\partial x -$$
$$- \frac{q^3d^2(d+2)}{6n^3}\int_0^{x_0} x^4e^{-\frac{x^2qd}{2n}}\partial x + O\left(\frac{\ln^4 n}{n}\right) =$$
$$\frac{qd}{n}\int_0^{x_0} xe^{-\frac{x^2qd}{2n}}\partial x + \frac{q^2d(d+2)}{2n^2}x_0^3e^{-\frac{x_0^2qd}{2n}} + O\left(\frac{\ln^4 n}{n}\right) =$$
$$1 - e^{-\frac{x_0^2qd}{2n}} + O\left(\frac{\ln^4 n}{n}\right) = 1 + O\left(\frac{\ln^4 n}{n}\right). \quad (4.16)$$

When computing sums by integration we have used (4.15). For the exponential integrals in (4.16) we have applied the formulae $\int x^be^{-ax^2}\partial x = -\frac{x^{b-1}}{2a}e^{-ax^2} + \frac{b-1}{2a}\int x^{b-2}e^{-ax^2}\partial x$ and $\int xe^{-ax^2}\partial x = -\frac{1}{2a}e^{-ax^2}$ [87, §1.3.3].

Equation (4.16) implies that

$$\sum_{x=\lceil x_0\rceil}^{n} f_\lambda(x) = 1 - \sum_{x=0}^{\lceil x_0\rceil-1} f_\lambda(x) = O\left(\frac{\ln^4 n}{n}\right). \quad (4.17)$$

Since $f_\lambda(x) \geq 0$ for any $x \geq 0$, the value (4.17) is the maximum among all possible subsums of the sequence $(f_\lambda(x_0), f_\lambda(x_0+1), \ldots, f_\lambda(n))$. Hence, by Abel's inequality [121, §II.2.301], the following is true:

$$\sum_{x=\lceil x_0 \rceil}^{n} x f_\lambda(x) \leq nO\left(\frac{\ln^4 n}{n}\right) = O(\ln^4 n).$$

□

Lemmas 4.5 and 4.6 let us calculate the expected value of λ:

$$\mathrm{E}(\lambda) = \sum_{x=0}^{n} x f_\lambda(x) = \sqrt{\frac{\pi n}{2qd}} + O(\ln^4 n). \tag{4.18}$$

4.A.4 Expected Number of Visited Nodes α

Substituting $q = \frac{1}{2}$ in (4.18), we find that the expected value of α equals

$$\mathrm{E}(\alpha) = \mathrm{E}(\lambda) + \mathrm{E}(\delta) + 1 = \sqrt{\frac{\pi n}{d}} + \frac{2}{\theta} + O(\ln^4 n).$$

This completes the proof of Theorem 4.1.

4.B Argument for Heuristic 4.2

We keep the notation from Section 4.A. Recall from (4.10) that α is a sum of two random variables. Hence [102, §4.4]

$$\mathrm{Var}(\alpha) \approx \mathrm{Var}(\lambda) + \mathrm{Var}(\delta) + 2\,\mathrm{Cov}(\lambda, \delta).$$

Using (4.12) and (4.18) with $q = \frac{1}{2}$, the variance of λ can be approximated as

$$\mathrm{Var}(\lambda) = \int_0^\infty f_\lambda(x)\,(x - \mathrm{E}(x))^2\,\partial x \approx \frac{d}{2n}\int_0^\infty x\left(x - \sqrt{\frac{\pi n}{d}}\right)^2 e^{-\frac{x^2 d}{4n}}\partial x = \frac{(4-\pi)n}{d}.$$

We have applied (4.21) and the formula $\int_0^\infty x^{2b+1} e^{-ax^2}\partial x = \frac{b!}{2a^{b+1}}$ for $a > 0$ [51, Eq. 3.461.3].

The random variable δ is described by the geometric distribution (4.11), hence its variance is

$$\mathrm{Var}(\delta) = \frac{4 - 2\theta}{\theta^2}.$$

What remains is to find the covariance of λ and δ. Note that both λ and δ tend to increase with the number of collisions, therefore λ and δ are not independent. Their covariance can be computed as [102, §4.3.1]

$$\mathrm{Cov}(\lambda, \delta) = \mathrm{E}(\lambda\delta) - \mathrm{E}(\lambda)\,\mathrm{E}(\delta).$$

Denote by M the number of collisions up to and including the first good collision. Using the fact that for a fixed M, the random variables λ and δ are independent, the term $\mathrm{E}(\lambda\delta)$ will be computed as

$$\mathrm{E}(\lambda\delta) = \sum_{m=1}^{\infty} f_M(m)\,\mathrm{E}(\,\lambda\delta \mid M = m\,) = \sum_{m=1}^{\infty} f_M(m)\,\mathrm{E}(\,\lambda \mid M = m\,)\,\mathrm{E}(\,\delta \mid M = m\,).$$

The process of trying collisions until a good one is found is described by the geometric distribution with the PMF

$$f_M(m) = (1-q)^{m-1}q = \frac{1}{2^m}, \quad \text{where } m \geq 1.$$

Given $M = m$, the number of duplicate visited nodes δ represents the number of Bernoulli trials with success probability θ necessary for the m-th success to occur. This is known as the negative binomial distribution [102, §6.1.3]. Hence the conditional PMF and the conditional expected value of δ are

$$f_\delta(\,x \mid M = m\,) = \binom{x-1}{m-1}(1-\theta)^{x-m}\theta^m, \text{ where } m \geq 1 \text{ and } x \geq 1.$$
$$\mathrm{E}(\,\delta \mid M = m\,) = \frac{m}{\theta}.$$

Lemma 4.7. *The PMF of λ given that $M = m$ is*

$$f_\lambda(\,x \mid M = m\,) = \frac{x^{2m-1}d^m}{(m-1)!\,2^{m-1}\,n^m}\,e^{-\frac{x^2 d}{2n}} + O\left(\frac{1}{n}\right), \tag{4.19}$$

where $0 \leq x = O(\sqrt{n})$.

Proof. We use the induction on M. Lemma 4.4 shows that the result is valid for $M = 1$, which is easy to see by substituting $q = 1$ in (4.12).

Assume the result is valid for $M = m-1$ for some $m \geq 2$, and consider the case when $M = m$. Let us express λ as the sum

$$\lambda = y + \lambda',$$

where y is the number of distinct nodes visited before the $(m-1)$-th collision minus one and λ' is the number of distinct nodes visited between the $(m-1)$-th and m-th collisions. The PMF of λ' can be obtained analogously to Lemma 4.4:

$$f_{\lambda'}(z) = \frac{(y+z)d}{n-(y+z)}\left(1-\frac{y+1}{n}\right)\prod_{i=y+2}^{y+z-1}\left(1-\frac{id}{n-i}\right) = \frac{(y+z)d}{n}e^{-\frac{(2y+z)zd}{2n}} + O\left(\frac{1}{n}\right).$$

Thus the conditional PMF of λ is the discrete convolution

$$f_\lambda(\, x \mid M = m \,) = \sum_{y=0}^{x} f_\lambda(\, y \mid M = m-1 \,)\, f_{\lambda'}(x-y) =$$
$$\sum_{y=0}^{x} \left(\frac{y^{2m-3} d^{m-1}}{(m-2)!\, 2^{m-2}\, n^{m-1}}\, e^{-\frac{y^2 d}{2n}} + O\left(\frac{1}{n}\right)\right) \left(\frac{xd}{n} e^{-\frac{(x+y)(x-y)d}{2n}} + O\left(\frac{1}{n}\right)\right) =$$
$$\frac{xd^m}{(m-2)!\, 2^{m-2}\, n^m}\, e^{-\frac{x^2 d}{2n}} \sum_{y=0}^{x} y^{2m-3} + O\left(\frac{1}{n}\right) =$$
$$\frac{xd^m}{(m-2)!\, 2^{m-2}\, n^m}\, e^{-\frac{x^2 d}{2n}} \left(\frac{x^{2m-2}}{2(m-1)} + O(x^{2m-3}) \right) + O\left(\frac{1}{n}\right) =$$
$$\frac{x^{2m-1} d^m}{(m-1)!\, 2^{m-1}\, n^m}\, e^{-\frac{x^2 d}{2n}} + O\left(\frac{1}{n}\right).$$

For computing the sum $\sum_{y=0}^{x} y^{2m-3}$ we have used the Euler-Maclaurin summation formula [1, §3.6.28], which implies that

$$\sum_{y=0}^{x} y^a = \int_0^x y^a\, \partial y + O\left(\max\left(\frac{\partial^i}{\partial y} y^a \Big|_{\substack{i \geq 0 \\ y=x}} \right) \right) = \frac{x^{a+1}}{a+1} + O(x^a).$$

□

We approximate the conditional PMF of λ at the interval $0 \leq x = O(\sqrt{n})$ by

$$f_\lambda(\, x \mid M = m \,) \approx \frac{x^{2m-1} d^m}{(m-1)!\, 2^{m-1}\, n^m}\, e^{-\frac{x^2 d}{2n}}.$$

Note that for such chosen approximation, for any $m \geq 1$ we have that $\int_0^\infty f_\lambda(\, x \mid M = m \,) = 1$. It is also possible to show that for any $0 < \epsilon < 1$ there exists a constant $c_{m,\epsilon}$ such that $1 - \int_0^{c_{m,\epsilon}\sqrt{n}} f_\lambda(\, x \mid M = m \,) \leq \epsilon$. Hence an approximation for the conditional expected value of λ can be computed as

$$\mathrm{E}(\, \lambda \mid M = m \,) \approx \int_0^\infty x f_\lambda(\, x \mid M = m \,) \partial x = \frac{d^m}{(m-1)!\, 2^{m-1}\, n^m} \int_0^\infty x^{2m} e^{-\frac{x^2 d}{2n}}\, \partial x =$$
$$\frac{d^m}{(m-1)!\, 2^{m-1}\, n^m}\, \frac{(2m)!\, n^m}{m!\, 2^{m+1}\, d^m} \sqrt{\frac{2\pi n}{d}} = \frac{(2m)!}{2^{2m}\, m!\, (m-1)!} \sqrt{\frac{2\pi n}{d}}. \quad (4.20)$$

For computing the integral we have used the formula

$$\int_0^\infty x^{2b} e^{-ax^2} \partial x = \frac{(2b-1)!!}{2^{b+1} a^b} \sqrt{\frac{\pi}{a}}, \quad (4.21)$$

where $a > 0$ and $b \in \mathbb{N}$ [51, Eq. 3.461.2]. The double factorial equals $(2b-1)!! = \frac{(2b)!}{b!\, 2^b}$.

An interesting observation is that in order to obtain exactly two collisions one should expect the algorithm to visit $\mathrm{E}(\,\lambda \mid M = 2\,)$ distinct nodes. But $\frac{\mathrm{E}(\,\lambda|M=2\,)}{\mathrm{E}(\lambda)} = 3\sqrt{2}/4 \approx 1.06$, which means that, by our approximated formula (4.20), on average the algorithm terminates earlier than it is expected to find two collisions.

We now compute

$$\mathrm{E}(\lambda\delta) = \sum_{m=1}^{\infty} f_M(m)\,\mathrm{E}(\,\lambda \mid M = m\,)\,\mathrm{E}(\,\delta \mid M = m\,) \approx \frac{1}{\theta}\sqrt{\frac{2\pi n}{d}}\sum_{m=1}^{\infty}\frac{(2m)!}{2^{3m}\,(m-1)!\,(m-1)!} = \frac{1}{8\theta}\sqrt{\frac{2\pi n}{d}}\sum_{k=0}^{\infty}\frac{(2k+2)!}{(k!)^2\,8^k} = \frac{5}{2\theta}\sqrt{\frac{\pi n}{d}}.$$

For the series we have used the formulae $\sum_{k=0}^{\infty}\frac{(2k+3)!}{k!\,(k+1)!}a^k = 6(1-4a)^{-5/2}$ and $\sum_{k=0}^{\infty}\frac{(2k+1)!}{(k!)^2}a^k = (1-4a)^{-3/2}$ when $|a| < \frac{1}{4}$ [87, §5.2.13].

Finally we assemble the summands to find that

$$\mathrm{Cov}(\lambda,\delta) \approx \frac{5}{2\theta}\sqrt{\frac{\pi n}{d}} - \frac{2}{\theta}\sqrt{\frac{\pi n}{d}} = \frac{1}{2\theta}\sqrt{\frac{\pi n}{d}},$$
$$\mathrm{Var}(\alpha) \approx \frac{(4-\pi)n}{d} + \frac{4-2\theta}{\theta^2} + \frac{1}{\theta}\sqrt{\frac{\pi n}{d}}.$$

4.C Numerical Experiments

In this section we provide more details about Experiment 4.1 (measurement of L in arbitrary groups) and list its full set of results. We report about other experiments with adding walks as well.

4.C.1 Details and Results of Experiment 4.1

When choosing the supporting set H, we have to check that it generates the group G. Let $G \cong \mathbb{Z}_{n_1} \oplus \cdots \oplus \mathbb{Z}_{n_s}$. We write elements of G in the vector form $\vec{g} = (g_1, \ldots, g_s) \in \mathbb{Z}_{n_1} \oplus \cdots \oplus \mathbb{Z}_{n_s}$. Denote also by $\vec{e_i}$ the vector with 1 in position i and zeros elsewhere. The subset $\{\vec{e_1}, \ldots, \vec{e_s}\}$ generates G. Let now a subset $H = \{\vec{h_1}, \ldots, \vec{h_t}\} \subset G$ be chosen, where $t \geq s$. The subset H generates G if and only if each of the elements $\vec{e_i}$, $1 \leq i \leq s$, can be expressed as a linear combination of the elements of H. This is equivalent to saying that there exists a $t \times s$ integer matrix X such that

$$\begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,t} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,t} \\ \vdots & \vdots & \ddots & \vdots \\ h_{s,1} & h_{s,2} & \cdots & h_{s,t} \end{pmatrix} \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,s} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ x_{t,1} & x_{t,2} & \cdots & x_{t,s} \end{pmatrix} \equiv E \mod \begin{pmatrix} n_1 \\ n_2 \\ \vdots \\ n_s \end{pmatrix},$$

where E is the $s \times s$ identity matrix. In the matrix multiplication above all computations involving $h_{i,j}$ are done modulo n_i. Thus the fact that H generates G can be verified by solving the system of linear modular equations.

For constructing a random group G of a given cardinality limit $\lceil \log \#G \rceil = m$ we use the following algorithm. For a fixed n, denote by c_n the number of isomorphism classes of abelian groups of order n. Denote by $c_{\max}$ the maximal value of c_n achieved on the interval $n \in \{2^{m-1}+1, \ldots, 2^m\}$. The value $c_{\max}$ is achieved for $n = 2^m$ and can be found as the number of partitions of m. Consider Algorithm 4.3.

Alg. 4.3 Random sampling of a group

Input: $m \in \mathbb{N}$

$c_{\max} \leftarrow \#\,\text{partitions}(m)$
repeat
  $n \xleftarrow{R} \{2^{m-1}+1, \ldots, 2^m\}$
  $c_n \leftarrow \#\{\text{isomorphism classes of groups of cardinality } n\}$
  $r \xleftarrow{R} \{0, \ldots, c_{\max}-1\}$
until $r < c_n$
$G \xleftarrow{R} \{\text{isomorphism classes of groups of cardinality } n\}$

Output: G

Lemma 4.8. *Algorithm 4.3 outputs a group sampled uniformly at random from the finite abelian groups of cardinalities $n \in \{2^{m-1}+1, \ldots, 2^m\}$.*

Proof. On any individually taken repeat-until round, the probability to choose n equals

$$\Pr[\,\text{choose } n \text{ in a round}\,] = \frac{c_n}{2^{m-1} c_{\max}},$$

and the probability to proceed to the next round (call this event E) can be found as

$$\Pr[E] = \sum_{n=2^{m-1}+1}^{2^m} \frac{1}{2^{m-1}} \left(1 - \frac{c_n}{c_{\max}}\right).$$

Now we can find the probability that the algorithm outputs a group of cardinality n:

$$\Pr[\,\#G = n \mid G \leftarrow \text{Algorithm 4.3}\,] = \frac{c_n}{2^{m-1} c_{\max}} \sum_{i=1}^{\infty} \Pr[E]^{i-1} = c\, c_n,$$

where the constant c is independent from n.

The total number of possible isomorphism classes on the interval $n \in \{2^{m-1}+1, \ldots, 2^m\}$ equals $N = \sum_{n=2^{m-1}+1}^{2^m} c_n$. The probability that a group chosen uniformly at random has cardinality n is

$$\Pr[\,\#G = n \mid G \xleftarrow{R} \{\text{groups of cardinalities } n, \lceil \log n \rceil = m\}\,] = \frac{c_n}{N}.$$

Now the equality

$$\frac{1}{N}\sum_{n=2^{m-1}+1}^{2^m} c_n = c\sum_{n=2^{m-1}+1}^{2^m} c_n = 1$$

implies that $c = \frac{1}{N}$. □

In Table 4.5 we present the expected values of L and in Table 4.6 the standard deviations of L obtained in Experiment 4.1. In both tables, the average is taken over random choices of abelian groups of orders n satisfying $\lceil \log n \rceil = m$ for a fixed m, random choices of supporting sets and random choices of problem instances. Mantissas are rounded to four decimal digits.

We can observe that the expected value of L increases as the group order n grows. However, for $r \geq 4$, the slope of $L(n)$ decreases as n gets bigger. The same holds for the standard deviations of L. We also see a few inconsistent values of $\mathrm{Stdev}(L)$ for $n < 2^{40}$ (in the three leftmost columns of Table 4.6). Most likely, these are caused by walk loops.

We also computed the values $\sigma = \mathrm{E}(L)/\,\mathrm{E}(L_\pi)$ and approximated them by functions of the form

$$\sigma(n) = c_0 - \frac{c_1}{n^{c_2}}. \tag{4.22}$$

The formula for $\sigma(n)$ is hypothetical, but with carefully chosen constants c_0, c_1 and c_2 we get a very close approximation of the experimental results, cf. Fig. 4.2. The constants are listed in Table 4.7 for completeness. Using Theorem 4.1, (4.22) and Table 4.7 one may easily compute the expected value of L for any group order.

4.C.2 How Aperiodicity of the Supporting Set Affects L

A set $\{g_1, \ldots, g_r\} \subset \mathbb{Z}_n^+$ is called *aperiodic* if $\gcd(g_2 - g_1, \ldots, g_r - g_1, n) = 1$ and $g_1, \ldots, g_r$ are all distinct. Dai and Hildebrand, studying the mixing property of adding walks, note that the aperiodicity of the supporting set is a necessary condition for the adding walk to look like a random mapping [33]. Indeed, a subset $H = \{g_1, \ldots, g_r\}$ is aperiodic if and only if the subset $\{g_2 - g_1, \ldots, g_r - g_1\}$ generates $\mathbb{Z}_n^+$. When the additive walk is started from 0, its state after m hops can be expressed as

$$z_m = \sum_{i=1}^{r} m_i g_i = m g_1 + \sum_{i=2}^{r} (m_i g_i - m_i g_1).$$

The rightmost sum is an element of the subgroup $\langle g_2 - g_1, \ldots, g_r - g_1 \rangle$, and so in order for z_m to be a random element of $\mathbb{Z}_n^+$, the subset H has to be aperiodic. Consider the following example where the subset $H = \{1, 3\}$ generates the group $\mathbb{Z}_{2n}^+$, but H is not aperiodic in $\mathbb{Z}_{2n}^+$ since $\gcd(3 - 1, 2n) = 2$. For the walk supported by H, we have that z_m is even when m is even, and z_m is odd when m is odd. Such a walk is certainly not random-looking.

		$\lceil \log n \rceil$							
w	r	28	32	36	40	44	48	52	56
1	3	2.8547	2.9982	3.1380	3.2735	3.4079	3.5355	3.6681	3.7812
	4	2.2923	2.3101	2.3247	2.3371	2.3484	2.3518	2.3661	2.3661
	5	2.1039	2.0975	2.0968	2.0984	2.0978	2.1007	2.1009	2.1004
	6	2.0178	2.0099	2.0052	2.0032	2.0026	2.0038	2.0022	2.0023
	7	1.9696	1.9612	1.9543	1.9519	1.9519	1.9531	1.9508	1.9543
	8	1.9400	1.9289	1.9242	1.9206	1.9210	1.9177	1.9202	1.9146
	9	1.9203	1.9083	1.9035	1.8995	1.8989	1.8933	1.8988	1.8984
	10	1.9021	1.8932	1.8862	1.8849	1.8831	1.8816	1.8769	1.8879
	11	1.8916	1.8801	1.8757	1.8730	1.8715	1.8644	1.8714	1.8700
	12	1.8827	1.8709	1.8659	1.8627	1.8617	1.8531	1.8607	1.8613
	13	1.8744	1.8621	1.8575	1.8552	1.8530	1.8531	1.8524	1.8503
	14	1.8684	1.8566	1.8508	1.8480	1.8465	1.8474	1.8443	1.8497
	15	1.8620	1.8507	1.8442	1.8416	1.8398	1.8424	1.8369	1.8432
	16	1.8575	1.8455	1.8407	1.8384	1.8361	1.8302	1.8369	1.8357
$\frac{3}{4}$	3	2.8980	3.0437	3.1893	3.3280	3.4667	3.6046	3.7199	3.8457
	4	2.3461	2.3667	2.3848	2.3986	2.4087	2.4129	2.4188	2.4243
	5	2.1635	2.1608	2.1598	2.1621	2.1634	2.1677	2.1619	2.1631
	6	2.0840	2.0756	2.0734	2.0717	2.0713	2.0764	2.0620	2.0694
	7	2.0410	2.0321	2.0272	2.0261	2.0260	2.0209	2.0250	2.0248
	8	2.0154	2.0055	2.0021	1.9990	1.9981	1.9987	2.0012	1.9968
	9	1.9995	1.9900	1.9849	1.9833	1.9819	1.9831	1.9846	1.9839
	10	1.9883	1.9775	1.9733	1.9710	1.9706	1.9638	1.9721	1.9654
	11	1.9805	1.9699	1.9649	1.9648	1.9611	1.9611	1.9616	1.9629
	12	1.9755	1.9644	1.9604	1.9580	1.9568	1.9566	1.9565	1.9606
	13	1.9728	1.9628	1.9568	1.9547	1.9544	1.9541	1.9513	1.9499
	14	1.9738	1.9627	1.9581	1.9556	1.9533	1.9532	1.9537	1.9472
	15	1.9700	1.9592	1.9549	1.9520	1.9520	1.9485	1.9489	1.9501
	16	1.9679	1.9565	1.9506	1.9500	1.9484	1.9502	1.9445	1.9481
$\frac{1}{2}$	3	3.1089	3.2761	3.4406	3.5985	3.7500	3.9086	4.0331	4.1785
	4	2.6071	2.6436	2.6723	2.6938	2.7101	2.7307	2.7315	2.7406
	5	2.4586	2.4665	2.4723	2.4782	2.4802	2.4875	2.4821	2.4776
	6	2.4000	2.4022	2.4063	2.4068	2.4086	2.4079	2.4069	2.4128
	7	2.3738	2.3771	2.3773	2.3774	2.3793	2.3800	2.3835	2.3889
	8	2.3616	2.3621	2.3631	2.3654	2.3662	2.3643	2.3636	2.3723
	9	2.3567	2.3557	2.3593	2.3574	2.3590	2.3553	2.3578	2.3533
	10	2.3529	2.3536	2.3527	2.3553	2.3563	2.3486	2.3533	2.3557
	11	2.3526	2.3529	2.3524	2.3534	2.3538	2.3577	2.3538	2.3541
	12	2.3513	2.3522	2.3526	2.3539	2.3535	2.3538	2.3553	2.3567
	13	2.3513	2.3519	2.3517	2.3519	2.3533	2.3538	2.3484	2.3492
	14	2.3520	2.3503	2.3507	2.3522	2.3523	2.3577	2.3488	2.3540
	15	2.3528	2.3512	2.3520	2.3511	2.3514	2.3506	2.3530	2.3524
	16	2.3516	2.3523	2.3519	2.3519	2.3536	2.3524	2.3465	2.3576
$\frac{1}{3}$	3	3.4906	3.6994	3.9037	4.1006	4.2821	4.4686	4.6310	4.7936
	4	3.0835	3.1559	3.2119	3.2526	3.2813	3.3060	3.2997	3.3238
	5	2.9829	3.0273	3.0590	3.0748	3.0873	3.0924	3.1058	3.0993
	6	2.9493	2.9903	3.0159	3.0304	3.0379	3.0467	3.0491	3.0478
	7	2.9418	2.9784	3.0033	3.0148	3.0236	3.0325	3.0234	3.0255
	8	2.9402	2.9759	2.9982	3.0113	3.0177	3.0115	3.0251	3.0170
	9	2.9405	2.9740	2.9978	3.0083	3.0154	3.0142	3.0255	3.0175
	10	2.9397	2.9762	2.9973	3.0088	3.0143	3.0153	3.0173	3.0266
	11	2.9419	2.9743	2.9953	3.0090	3.0139	3.0180	3.0207	3.0234
	12	2.9406	2.9750	2.9964	3.0074	3.0139	3.0153	3.0292	3.0249
	13	2.9422	2.9760	2.9947	3.0078	3.0134	3.0127	3.0201	3.0172
	14	2.9395	2.9739	2.9954	3.0076	3.0157	3.0184	3.0187	3.0238
	15	2.9425	2.9751	2.9978	3.0082	3.0154	3.0245	3.0176	3.0243
	16	2.9436	2.9810	2.9987	3.0119	3.0179	3.0323	3.0151	3.0258
$\frac{1}{4}$	3	3.8596	4.1194	4.3652	4.5978	4.8213	5.0395	5.2484	5.4338
	4	3.5425	3.6694	3.7582	3.8280	3.8771	3.9015	3.9372	3.9517
	5	3.4753	3.5732	3.6423	3.6830	3.7103	3.7322	3.7295	3.7407
	6	3.4608	3.5566	3.6145	3.6526	3.6743	3.6985	3.6845	3.6845
	7	3.4594	3.5495	3.6107	3.6450	3.6665	3.6689	3.6818	3.6885
	8	3.4569	3.5488	3.6094	3.6456	3.6649	3.6782	3.6831	3.6854
	9	3.4585	3.5493	3.6087	3.6437	3.6630	3.6826	3.6891	3.6876
	10	3.4578	3.5486	3.6064	3.6454	3.6658	3.6672	3.6853	3.6833
	11	3.4590	3.5467	3.6068	3.6423	3.6638	3.6833	3.6668	3.6797
	12	3.4647	3.5467	3.6066	3.6452	3.6630	3.6755	3.6951	3.6914
	13	3.4620	3.5485	3.6074	3.6457	3.6656	3.6825	3.6795	3.6924
	14	3.4630	3.5490	3.6091	3.6471	3.6645	3.6733	3.6848	3.6865
	15	3.4608	3.5477	3.6071	3.6433	3.6624	3.6753	3.6749	3.6923
	16	3.4607	3.5498	3.6070	3.6427	3.6639	3.6747	3.6808	3.6880

Table 4.5: Expected values of L obtained experimentally.

		$\lceil \log n \rceil$							
w	r	28	32	36	40	44	48	52	56
1	3	1.6119	1.5704	1.6389	1.7109	1.7827	1.8459	1.9133	1.9738
	4	1.2709	1.2075	1.2151	1.2210	1.2273	1.2310	1.2417	1.2395
	5	1.1583	1.1011	1.0950	1.0964	1.0968	1.0980	1.1020	1.1028
	6	1.0547	1.0485	1.0471	1.0459	1.0472	1.0428	1.0459	1.0455
	7	1.0866	1.0238	1.0203	1.0200	1.0203	1.0176	1.0189	1.0225
	8	1.0164	1.0068	1.0046	1.0041	1.0042	1.0045	1.0050	1.0024
	9	1.0128	1.0077	0.9937	0.9929	0.9923	0.9943	0.9912	0.9925
	10	0.9980	0.9880	0.9857	0.9852	0.9844	0.9838	0.9847	0.9840
	11	0.9871	0.9811	0.9789	0.9788	0.9781	0.9757	0.9776	0.9790
	12	0.9885	0.9765	0.9744	0.9730	0.9725	0.9701	0.9755	0.9717
	13	0.9792	0.9721	0.9706	0.9690	0.9691	0.9666	0.9679	0.9682
	14	0.9749	0.9694	0.9663	0.9661	0.9650	0.9631	0.9623	0.9694
	15	0.9724	0.9663	0.9636	0.9619	0.9622	0.9638	0.9607	0.9662
	16	0.9705	0.9637	0.9622	0.9606	0.9601	0.9588	0.9605	0.9575
$\frac{3}{4}$	3	1.7501	1.5911	1.6671	1.7401	1.8111	1.8816	1.9396	2.0115
	4	1.2529	1.2408	1.2455	1.2521	1.2583	1.2666	1.2634	1.2688
	5	1.4546	1.1281	1.1293	1.1298	1.1308	1.1286	1.1318	1.1278
	6	1.0913	1.0834	1.0832	1.0825	1.0823	1.0831	1.0820	1.0803
	7	1.0660	1.0609	1.0594	1.0590	1.0590	1.0555	1.0594	1.0583
	8	1.0536	1.0478	1.0452	1.0457	1.0448	1.0447	1.0470	1.0429
	9	1.0507	1.0387	1.0360	1.0362	1.0357	1.0387	1.0425	1.0324
	10	1.0555	1.0333	1.0312	1.0311	1.0295	1.0271	1.0312	1.0260
	11	1.0412	1.0416	1.0271	1.0266	1.0256	1.0263	1.0283	1.0206
	12	1.0323	1.0257	1.0243	1.0227	1.0221	1.0231	1.0221	1.0243
	13	1.0312	1.0243	1.0218	1.0216	1.0209	1.0142	1.0183	1.0196
	14	1.0949	1.0242	1.0225	1.0222	1.0206	1.0205	1.0180	1.0175
	15	1.0332	1.0227	1.0206	1.0199	1.0200	1.0182	1.0184	1.0192
	16	1.1664	1.0333	1.0191	1.0189	1.0175	1.0163	1.0194	1.0201
$\frac{1}{2}$	3	1.6342	1.7247	1.7982	1.8805	1.9621	2.0379	2.1049	2.1894
	4	1.5050	1.3880	1.3960	1.4081	1.4169	1.4189	1.4333	1.4281
	5	1.5815	1.3030	1.2924	1.2945	1.2965	1.2997	1.3009	1.2946
	6	1.3156	1.2559	1.2570	1.2579	1.2591	1.2617	1.2573	1.2589
	7	1.2517	1.2616	1.2419	1.2419	1.2436	1.2474	1.2437	1.2472
	8	1.2637	1.2407	1.2347	1.2360	1.2365	1.2387	1.2370	1.2414
	9	1.2609	1.2306	2.4437	1.2321	1.2324	1.2354	1.2347	1.2287
	10	1.2490	1.2331	1.2300	1.2313	1.2309	1.2277	1.2314	1.2296
	11	2.1772	1.2957	1.2291	1.2310	1.2304	1.2293	1.2279	1.2300
	12	1.2504	1.2274	1.2289	1.2305	1.2300	1.2314	1.2313	1.2310
	13	1.2700	1.2395	1.2289	1.2290	1.2299	1.2318	1.2268	1.2279
	14	1.2878	1.2289	1.2471	1.2293	1.2285	1.2301	1.2267	1.2321
	15	1.5222	1.2547	1.2455	1.2285	1.2285	1.2313	1.2316	1.2332
	16	1.4334	1.2279	1.2339	1.2290	1.2304	1.2257	1.2209	1.2349
$\frac{1}{3}$	3	1.8637	1.9686	2.0390	2.1425	2.2398	2.3367	2.4248	2.5050
	4	1.6339	1.6496	1.6789	1.6994	1.7148	1.7271	1.7314	1.7390
	5	1.7092	1.5832	1.5986	1.6080	1.6130	1.6182	1.6190	1.6174
	6	1.5760	1.5632	1.5805	1.5840	1.5873	1.5927	1.5954	1.5952
	7	1.6433	1.5602	1.5685	1.5750	1.5804	1.5834	1.5794	1.5831
	8	1.6090	1.5694	1.5665	1.5738	1.5760	1.5734	1.5854	1.5787
	9	2.0606	1.5568	1.5664	1.5712	1.5758	1.5739	1.5823	1.5775
	10	2.2107	1.5768	1.5658	1.5724	1.5762	1.5786	1.5760	1.5889
	11	4.2293	1.6783	1.5649	1.5723	1.5760	1.5769	1.5784	1.5834
	12	2.5554	1.9046	1.5652	1.5724	1.5756	1.5809	1.5765	1.5808
	13	2.3661	1.5711	1.5642	1.5727	1.5753	1.5686	1.5804	1.5756
	14	1.9056	1.5662	1.5658	1.5712	1.5754	1.5784	1.5794	1.5773
	15	1.8106	1.5994	1.5672	1.5725	1.5751	1.5789	1.5788	1.5742
	16	1.7797	6.3869	1.5689	1.5742	1.5770	1.5827	1.5783	1.5859
$\frac{1}{4}$	3	2.4416	2.1520	2.2821	2.4048	2.5200	2.6389	2.7395	2.8478
	4	1.8971	1.9198	1.9632	2.0001	2.0279	2.0439	2.0516	2.0760
	5	2.5911	1.8910	1.9018	1.9248	1.9383	1.9461	1.9521	1.9605
	6	2.2320	2.0988	1.8906	1.9094	1.9192	1.9230	1.9251	1.9331
	7	2.8160	1.8566	1.8870	1.9046	1.9170	1.9145	1.9236	1.9341
	8	1.9505	1.8729	1.8847	1.9054	1.9172	1.9196	1.9269	1.9314
	9	2.6773	1.8568	1.8841	1.9058	1.9152	1.9262	1.9324	1.9154
	10	2.1574	2.0621	1.8860	1.9047	1.9153	1.9196	1.9282	1.9281
	11	2.4498	1.8807	1.8879	1.9048	1.9143	1.9226	1.9214	1.9316
	12	4.6256	1.9160	1.8942	1.9038	1.9145	1.9193	1.9359	1.9319
	13	4.2067	1.8591	1.8853	1.9060	1.9153	1.9216	1.9149	1.9325
	14	3.2386	1.9366	1.8853	1.9064	1.9159	1.9168	1.9219	1.9273
	15	2.5811	1.8688	1.8900	1.9048	1.9159	1.9244	1.9171	1.9251
	16	3.6983	1.8879	1.8844	1.9022	1.9153	1.9151	1.9254	1.9310

Table 4.6: Standard deviations of L obtained experimentally.

w	r	c_0	c_1	c_2
	4	1.165	0.309	0.093
	5	1.060	0.106	0.143
	6	1.031	0.100	1.000
	7	1.020	0.100	1.000
	8	1.013	0.100	1.000
	9	1.010	0.100	1.000
1	10	1.008	0.100	1.000
	11	1.006	0.100	1.000
	12	1.005	0.100	1.000
	13	1.005	0.100	1.000
	14	1.004	0.100	1.000
	15	1.003	0.100	1.000
	16	1.003	0.100	1.000
	4	1.169	0.503	0.121
	5	1.071	0.124	0.138
	6	1.041	1.037	0.444
	7	1.030	0.100	1.000
	8	1.024	0.100	1.000
	9	1.021	0.100	1.000
$\frac{3}{4}$	10	1.018	0.100	1.000
	11	1.017	0.100	1.000
	12	1.017	0.100	1.000
	13	1.016	0.100	1.000
	14	1.017	0.100	1.000
	15	1.017	0.100	1.000
	16	1.016	0.100	1.000
	4	1.229	0.590	0.108
	5	1.125	0.960	0.211
	6	1.102	0.236	0.160
	7	1.096	0.100	0.108
	8	1.089	0.156	0.148
	9	1.084	1.952	0.308
$\frac{1}{2}$	10	1.084	0.462	0.222
	11	1.085	0.210	0.174
	12	1.085	0.257	0.182
	13	1.083	1.403	0.286
	14	1.084	0.316	0.200
	15	1.084	0.116	0.154
	16	1.084	0.302	0.200

w	r	c_0	c_1	c_2
	4	1.317	1.249	0.129
	5	1.235	1.283	0.167
	6	1.216	1.681	0.190
	7	1.209	1.777	0.200
	8	1.206	2.000	0.211
	9	1.206	1.640	0.200
$\frac{1}{3}$	10	1.206	2.000	0.211
	11	1.207	1.395	0.190
	12	1.208	1.222	0.182
	13	1.205	1.920	0.211
	14	1.207	1.421	0.190
	15	1.207	1.669	0.200
	16	1.208	1.664	0.200
	4	1.418	1.527	0.118
	5	1.339	1.888	0.154
	6	1.324	1.862	0.160
	7	1.319	2.000	0.167
	8	1.322	1.854	0.160
	9	1.322	1.845	0.160
$\frac{1}{4}$	10	1.319	2.000	0.167
	11	1.318	1.988	0.167
	12	1.322	1.828	0.160
	13	1.322	1.832	0.160
	14	1.320	2.000	0.167
	15	1.319	2.000	0.167
	16	1.319	2.000	0.167

Table 4.7: Coefficients for approximating $\mathrm{E}(L)/\,\mathrm{E}(L_\pi)$ by formula (4.22).

w	r	periodic	aperiodic
1	4	2.3473 (1.2267)	2.3474 (1.2264)
	6	2.0024 (1.0466)	2.0012 (1.0463)
	10	1.8826 (0.9839)	1.8837 (0.9848)
	16	1.8358 (0.9591)	1.8361 (0.9592)
$\frac{1}{2}$	4	2.7090 (1.4153)	2.7081 (1.4155)
	6	2.4086 (1.2583)	2.4093 (1.2582)
	10	2.3558 (1.2321)	2.3550 (1.2324)
	16	2.3518 (1.2294)	2.3527 (1.2291)
$\frac{1}{4}$	4	3.8771 (2.0266)	3.8722 (2.0238)
	6	3.6750 (1.9207)	3.6731 (1.9202)
	10	3.6638 (1.9150)	3.6625 (1.9144)
	16	3.6655 (1.9158)	3.6629 (1.9136)

Table 4.8: Values E(L) (Stdev(L)) for periodic and aperiodic supporting sets.

However important the aperiodicity is for the mixing of adding walks, we show that it is not necessary for achieving collisions. In other words, the aperiodicity of supporting set does not affect the performance of Algorithm $\mathcal{A}$.

We will call a set *periodic* if $\gcd(g_2 - g_1, \ldots, g_r - g_1, n) > 1$ and $g_1, \ldots, g_r$ are all distinct. The following experimental evidence shows that the expected value and the standard deviation of L do not depend on the aperiodicity property of the supporting set, when a cyclic group and a supporting set are sampled uniformly at random.

Experiment 4.2 (Comparison of L for Periodic and Aperiodic Supporting Sets)**.** Let $G = \mathbb{Z}_n^+$, where n can take values satisfying $\lceil \log n \rceil = 44$. For each of the values $r \in \{4, 6, 10, 16\}$ and $w \in \{1, \frac{1}{2}, \frac{1}{4}\}$ conduct k_1 experiments. In each experiment, randomly choose a composite integer n. Then choose a random periodic supporting set that generates G and a random aperiodic supporting set. Choose a problem instance at random and run algorithm $\mathcal{A}$. Results are given in Table 4.8.

The differences between the values in Table 4.8 are within the confidence intervals. Thus it is sufficient to choose a random supporting set in Algorithm $\mathcal{A}$, irrespective of the aperiodicity. However, it is crucial to check that the supporting set generates G.

4.C.3 How Group Cyclicity Affects L

Experiment 4.3 (Comparison of L in Finite Groups of Different Ranks)**.** Let $\lceil \log n \rceil = 44$. For each of the values $s \in \{1, 2, 4, 6\}$, $r \in \{4, 6, 10, 16\}$ and $w \in \{1, \frac{1}{2}, \frac{1}{4}\}$ conduct k_1 experiments. In every experiment choose a random group G of

w	r	$\operatorname{rank} G = 1$	$\operatorname{rank} G = 2$	$\operatorname{rank} G = 4$	$\operatorname{rank} G = 6$
1	4	2.3474 (1.2273)	2.3474 (1.2273)	2.3466 (1.2268)	–
	6	2.0018 (1.0465)	2.0017 (1.0455)	2.0009 (1.0456)	2.0031 (1.0471)
	10	1.8835 (0.9835)	1.8843 (0.9844)	1.8832 (0.9841)	1.8827 (0.9843)
	16	1.8367 (0.9598)	1.8361 (0.9595)	1.8371 (0.9593)	1.8363 (0.9591)
$\frac{1}{2}$	4	2.7085 (1.4151)	2.7085 (1.4153)	2.7100 (1.4174)	–
	6	2.4095 (1.2602)	2.4096 (1.2590)	2.4096 (1.2597)	2.4090 (1.2585)
	10	2.3554 (1.2315)	2.3548 (1.2316)	2.3569 (1.2328)	2.3566 (1.2303)
	16	2.3525 (1.2300)	2.3525 (1.2296)	2.3530 (1.2299)	2.3513 (1.2281)
$\frac{1}{4}$	4	3.8766 (2.0250)	3.8737 (2.0265)	3.8755 (2.0224)	–
	6	3.6764 (1.9214)	3.6737 (1.9217)	3.6729 (1.9207)	3.6752 (1.9214)
	10	3.6637 (1.9168)	3.6622 (1.9157)	3.6654 (1.9159)	3.6637 (1.9138)
	16	3.6634 (1.9138)	3.6640 (1.9149)	3.6643 (1.9152)	3.6654 (1.9148)

Table 4.9: Values $\mathrm{E}(L)$ ($\mathrm{Stdev}(L)$) in groups of rank 1, 2, 4 and 6.

rank s and a random subset of r elements that generates G. Use $\theta = n^{-1/4}$ and the partitioning probabilities decreasing with ratio w. The results are listed in Table 4.9. Mantissas are rounded to four decimal digits.

For constructing a random group $G = \mathbb{Z}_{n_1} \oplus \cdots \oplus \mathbb{Z}_{n_s}$ of a given rank s and a given cardinality limit $\lceil \log \#G \rceil = l$ we use a modified version of Algorithm 4.3, namely the algorithm is restricted to groups of a fixed rank s. The value c_n is replaced by $c_{n,s}$, the number of isomorphism classes of abelian groups of order n and rank s. The value $c_{\max}$ is now computed as the number of partitions of l of length s. The rest of the algorithm remains unchanged.

We see that the differences between the values in Table 4.9 for groups of various ranks do not exceed the size of the confidence interval. This allows us to hypothesize that the expected value and the standard deviation of L is independent of the rank, when a group of a fixed small rank is sampled uniformly at random.

4.C.4 Values of L in Ideal Class Groups of Elliptic Curve CM Fields

In this section we show that the values $\mathrm{E}(L)$ and $\mathrm{Stdev}(L)$ for the $\mathcal{CL}$-GAIP are the same as for random abelian groups, and hence the results of Experiment 4.1 also apply to random instances of $\mathcal{CL}$-GAIP.

Experiment 4.4 (Measuring L in Ideal Class Groups of Elliptic Curve CM Fields)**.** In every experiment, choose a random 90-bit prime p and a random elliptic curve $E/\mathbb{F}_p$ with a fundamental Frobenius discriminant Δ_E. Run the algorithm $\mathcal{A}$ in the additive group of integers isomorphic to the class group $\mathcal{CL}(\Delta_E)$, choosing

		E(L)			Stdev(L)		
w	r	Exp. 4.1	Exp. 4.4	diff. %	Exp. 4.1	Exp. 4.4	diff. %
1	4	2.3476	2.3473	−0.01	1.2279	1.2282	0.03
	6	2.0028	2.0027	−0.00	1.0458	1.0467	0.09
	10	1.8843	1.8827	−0.08	0.9847	0.9835	−0.12
	16	1.8371	1.8364	−0.04	0.9602	0.9606	0.04
$\frac{1}{2}$	4	2.7123	2.7098	−0.09	1.4168	1.4162	−0.04
	6	2.4082	2.4097	0.06	1.2588	1.2590	0.01
	10	2.3539	2.3553	0.06	1.2298	1.2308	0.08
	16	2.3524	2.3531	0.03	1.2293	1.2302	0.07
$\frac{1}{4}$	4	3.8756	3.8706	−0.13	2.0255	2.0238	−0.08
	6	3.6709	3.6753	0.12	1.9179	1.9210	0.16
	10	3.6605	3.6629	0.07	1.9152	1.9161	0.05
	16	3.6605	3.6640	0.09	1.9126	1.9153	0.14

Table 4.10: Values E(L) and Stdev(L) in ideal class groups of elliptic curve CM fields and in random finite abelian groups.

the supporting set at random. Run k_1 experiments for every combination of $r \in \{4, 6, 10, 16\}$ and $w \in \{1, 1/2, 1/4\}$.

Results are given in Table 4.10. The average class number in Experiment 4.4 was approximately $2^{43.715}$.

We see that the difference between values E(L) in class groups and in random finite groups does not exceed 0.2 %. Since we are comparing results from two experiments with ±0.1 % confidence intervals, and expecting results from the two experiments to be the same, we conclude that the experiment confirms our hypothesis, with the obtained difference being within the margin of error. The same holds for Stdev(L).

Bibliography

[1] Milton Abramowitz and Irene A. Stegun, editors. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover Publications Inc., New York, 1992. ISBN 0-486-61272-4. Reprint of the 1972 edition. (Cited on pages 97 and 101.)

[2] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. ISSN 0209-9683. URL `http://dx.doi.org/10.1007/BF02579403`. (Cited on page 36.)

[3] Daniel V. Bailey, Lejla Batina, Daniel J. Bernstein, Peter Birkner, Joppe W. Bos, Hsieh-Chung Chen, Chen-Mou Cheng, Gauthier van Damme, Giacomo de Meulenaer, Luis Julian Dominguez Perez, Junfeng Fan, Tim Gneysu, Frank Gurkaynak, Thorsten Kleinjung, Tanja Lange, Nele Mentens, Ruben Niederhagen, Christof Paar, Francesco Regazzoni, Peter Schwabe, Leif Uhsadel, Anthony Van Herrewege and Bo-Yin Yang. Breaking ECC2K-130. Cryptology ePrint Archive, Report 2009/541, 2009. URL `http://eprint.iacr.org/2009/541`. (Cited on pages 77 and 78.)

[4] Stephane Beauregard. Circuit for Shor's algorithm using $2n + 3$ qubits, 2003. URL `http://arxiv.org/abs/quant-ph/0205095v3`. (Cited on page 2.)

[5] Karim Belabas, Francisco Diaz y Diaz and Eduardo Friedman. Small generators of the ideal class group. *Math. Comp.*, 77(262):1185–1197, 2008. ISSN 0025-5718. URL `http://dx.doi.org/10.1090/S0025-5718-07-02003-0`. (Cited on page 32.)

[6] Juliana Belding, Reinier Bröker, Andreas Enge and Kristin Lauter. Computing Hilbert class polynomials. In *Algorithmic number theory*, volume 5011 of *Lecture Notes in Comput. Sci.*, pages 282–295. Springer, Berlin, 2008. URL `http://dx.doi.org/10.1007/978-3-540-79456-1_19`. (Cited on page 3.)

[7] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO' 93*, volume 773 of *Lecture Notes in Comput. Sci.*, pages 232–249. Springer-Verlag, Berlin, 1994. (Cited on pages 9 and 52.)

[8] Mihir Bellare, Anand Desai, David Pointcheval and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology—CRYPTO 1998*, volume 1462 of *Lecture Notes in Comput. Sci.*, pages 26–46. Springer, Berlin, 1998. (Cited on pages 27 and 59.)

[9] Daniel Bernstein. Grover vs. McEliece. In Nicolas Sendrier, editor, *Post-Quantum Cryptography*, volume 6061 of *Lecture Notes in Comput. Sci.*, pages 73–80. Springer Berlin / Heidelberg, 2010. URL `http://dx.doi.org/10.1007/978-3-642-12929-2_6`. (Cited on page 2.)

[10] Daniel J. Bernstein, Johannes Buchmann and Erik Dahmen, editors. *Post-quantum cryptography*. Springer-Verlag, Berlin, 2009. ISBN 978-3-540-88701-0. URL `http://dx.doi.org/10.1007/978-3-540-88702-7`. (Cited on pages 1 and 2.)

[11] Jean-François Biasse. Improvements in the computation of ideal class groups of imaginary quadratic number fields. *Adv. Math. Commun.*, 4(2):141–154, 2010. ISSN 1930-5346. URL `http://dx.doi.org/10.3934/amc.2010.4.141`. (Cited on pages 36 and 76.)

[12] Dario Bini and Victor Y. Pan. *Polynomial and Matrix Computations. Vol. 1.* Progress in Theoretical Computer Science. Birkhäuser Boston Inc., Boston, MA, 1994. ISBN 0-8176-3786-9. (Cited on pages 34 and 41.)

[13] Gaetan Bisson and Andrew V. Sutherland. Computing the endomorphism ring of an ordinary elliptic curve over a finite field. *Journal of Number Theory*, 131(5):815–831, 2011. ISSN 0022-314X. URL `http://www.sciencedirect.com/science/article/pii/S0022314X09002789`. Elliptic Curve Cryptography. (Cited on page 68.)

[14] Simon R. Blackburn and Sean Murphy. The number of partitions in Pollard rho, May 1998. Preprint. (Cited on pages 77 and 78.)

[15] Simon Blake-Wilson, Don Johnson and Alfred Menezes. Key agreement protocols and their security analysis (extended abstract). In *Cryptography and coding (Cirencester, 1997)*, volume 1355 of *Lecture Notes in Comput. Sci.*, pages 30–45. Springer, Berlin, 1997. URL `http://dx.doi.org/10.1007/BFb0024447`. (Cited on page 8.)

[16] Wieb Bosma, John Cannon and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. ISSN 0747-7171. URL `http://dx.doi.org/10.1006/jsco.1996.0125`. (Cited on page 13.)

[17] Richard P. Brent and John M. Pollard. Factorization of the eighth Fermat number. *Math. Comp.*, 36(154):627–630, 1981. ISSN 0025-5718. URL `http://dx.doi.org/10.2307/2007666`. (Cited on page 77.)

[18] Reinier Broker, Kristin Lauter and Andrew V. Sutherland. Modular polynomials via isogeny volcanoes. *Math. Comp.*, 2011. URL `http://dx.doi.org/10.1090/S0025-5718-2011-02508-1`. To appear. (Cited on pages 3 and 22.)

[19] Christian Cachin and Ueli Maurer. Smoothing probability distributions and smooth entropy (extended abstract). Institute for Theoretical Computer Science, ETH Zürich, 1996. (Cited on page 27.)

[20] Christian Cachin and Ueli Maurer. Smoothing probability distributions and smooth entropy. In *IEEE International Symposium on Information Theory (ISIT 1997)*, page 91, 1997. (Cited on pages 27 and 60.)

[21] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. Cryptology ePrint Archive, report 2001/040, 2001. URL `http://eprint.iacr.org/2001/040`. (Cited on pages 26, 52, 57 and 58.)

[22] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology—EUROCRYPT 2001 (Innsbruck)*, volume 2045 of *Lecture Notes in Comput. Sci.*, pages 453–474. Springer, Berlin, 2001. URL `http://dx.doi.org/10.1007/3-540-44987-6_28`. (Cited on pages 26 and 52.)

[23] Denis Charles and Kristin Lauter. Computing modular polynomials. *LMS J. Comput. Math.*, 8:195–204 (electronic), 2005. ISSN 1461-1570. (Cited on pages 3 and 22.)

[24] Denis X. Charles, Kristin E. Lauter and Eyal Z. Goren. Cryptographic hash functions from expander graphs. *J. Cryptology*, 22(1):93–113, 2009. ISSN 0933-2790. URL `http://dx.doi.org/10.1007/s00145-007-9002-x`. (Cited on pages 7, 22 and 28.)

[25] David Chaum, Jan-Hendrik Evertse and Jeroen van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In David Chaum and Wyn Price, editors, *Advances in Cryptology EUROCRYPT 87*, volume 304 of *Lecture Notes in Comput. Sci.*, pages 127–141. Springer Berlin / Heidelberg, 1988. URL `http://dx.doi.org/10.1007/3-540-39118-5_13`. (Cited on page 7.)

[26] Andrew M. Childs, David Jao and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time, 2010. URL `http://arxiv.org/abs/1012.4019v1`. (Cited on pages 2, 5 and 71.)

[27] H. Cohen and H. W. Lenstra, Jr. Heuristics on class groups of number fields. In *Number theory, Noordwijkerhout 1983 (Noordwijkerhout, 1983)*, volume 1068 of *Lecture Notes in Math.*, pages 33–62. Springer, Berlin, 1984. URL `http://dx.doi.org/10.1007/BFb0099440`. (Cited on page 88.)

[28] Henri Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993. ISBN 3-540-55640-0. (Cited on pages 10, 11, 12, 13, 15 and 34.)

[29] J.-M. Couveignes, L. Dewaghe and F. Morain. Isogeny cycles and the Schoof-Elkies-Atkin algorithm. research report LIX/RR/96/03, Laboratoire d'Informatique de l'Ecole Polytechnique, 1996. (Cited on page 33.)

[30] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. URL `http://eprint.iacr.org/2006/291`. (Cited on pages 7, 8, 22 and 68.)

[31] David A. Cox. *Primes of the Form x^2+ny^2*. A Wiley-Interscience Publication. John Wiley & Sons Inc., New York, 1989. ISBN 0-471-50654-0; 0-471-19079-9. (Cited on pages 10, 13, 14, 15 and 28.)

[32] Cas J. F. Cremers. Feasibility of multi-protocol attacks. In *ARES*, pages 287–294. IEEE Computer Society, 2006. URL `http://doi.ieeecomputersociety.org/10.1109/ARES.2006.63`. (Cited on page 63.)

[33] Jack J. Dai and Martin V. Hildebrand. Random random walks on the integers mod n. *Statist. Probab. Lett.*, 35(4):371–379, 1997. ISSN 0167-7152. URL `http://dx.doi.org/10.1016/S0167-7152(97)00035-7`. (Cited on pages 81 and 104.)

[34] He Debiao, Chen Jianhua and Hu Jin. A random number generator based on isogenies operations. Cryptology ePrint Archive, Report 2010/094, 2010. URL `http://eprint.iacr.org/2010/094`. (Cited on page 9.)

[35] He Debiao, Chen Jianhua and Hu Jin. An authenticated key agreement protocol using isogenies between elliptic curves. *International Journal of Computers Communications & Control*, 6:258–265, 2011. ISSN 1841-9836. (Cited on pages 9, 10 and 68.)

[36] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976. (Cited on page 54.)

[37] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. (Cited on page 52.)

[38] David S. Dummit and Richard M. Foote. *Abstract Algebra*. Wiley, third edition, 2004. (Cited on pages 53 and 55.)

[39] M. I. Dyakonov. Is fault-tolerant quantum computation really possible?, 2006. URL `http://arxiv.org/abs/quant-ph/0610117v1`. (Cited on page 6.)

[40] ECRYPT2. Yearly report on algorithms and keysizes. Technical Report D.SPA.7, European Network of Excellence in Cryptology II, July 2009. (Cited on page 38.)

[41] ECRYPT2. Yearly report on algorithms and keysizes. Technical report, European Network of Excellence in Cryptology II, March 2010. (Cited on pages 1 and 2.)

[42] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Advances in cryptology—CRYPTO '86 (Santa Barbara, Calif., 1986)*, volume 263 of *Lecture Notes in Comput. Sci.*, pages 186–194. Springer, Berlin, 1987. (Cited on page 43.)

[43] Steven Galbraith and Anton Stolbunov. Improved algorithm for the isogeny problem for ordinary elliptic curves, 2011. URL `http://arxiv.org/abs/1105.6331v1`. (Cited on page 5.)

[44] Steven D. Galbraith. Constructing isogenies between elliptic curves over finite fields. *LMS J. Comput. Math.*, 2:118–138 (electronic), 1999. ISSN 1461-1570. (Cited on pages 4, 22, 37, 68, 69 and 71.)

[45] Steven D. Galbraith, Florian Hess and Nigel P. Smart. Extending the GHS Weil descent attack. In *Advances in cryptology—EUROCRYPT 2002 (Amsterdam)*, volume 2332 of *Lecture Notes in Comput. Sci.*, pages 29–44. Springer, Berlin, 2002. URL `http://dx.doi.org/10.1007/3-540-46035-7_3`. (Cited on pages 3, 4, 22, 36, 38, 68, 70, 71, 72, 74, 75 and 86.)

[46] Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, Cambridge, 2001. ISBN 0-521-79172-3. URL `http://dx.doi.org/10.1017/CBO9780511546891`. Basic tools. (Cited on pages 27 and 52.)

[47] Oded Goldreich. Lecture notes: Randomized methods in computation, 2001. `http://www.wisdom.weizmann.ac.il/~oded/rnd.html`. (Cited on page 76.)

[48] Oded Goldreich, Silvio Micali and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *IEEE Symposium on Foundations of Computer Science*, pages 174–187, 1986. (Cited on page 7.)

[49] Oded Goldreich, Silvio Micali and Avi Wigderson. Proofs that yield nothing but their validity, or All languages in NP have zero-knowledge proof systems. *J. Assoc. Comput. Mach.*, 38(3):691–729, 1991. ISSN 0004-5411. URL `http://dx.doi.org/10.1145/116825.116852`. (Cited on page 7.)

[50] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. System Sci.*, 28(2):270–299, 1984. ISSN 0022-0000. URL `http://dx.doi.org/10.1016/0022-0000(84)90070-9`. (Cited on pages 27, 52 and 59.)

[51] I. S. Gradshteyn and I. M. Ryzhik. *Table of integrals, series, and products*. Elsevier/Academic Press, Amsterdam, seventh edition, 2007. ISBN 978-0-12-373637-6; 0-12-373637-4. Translated from the Russian. (Cited on pages 99 and 101.)

[52] James L. Hafner and Kevin S. McCurley. A rigorous subexponential algorithm for computation of class groups. *J. Amer. Math. Soc.*, 2(4):837–850, 1989. ISSN 0894-0347. URL `http://dx.doi.org/10.2307/1990896`. (Cited on page 32.)

[53] Sean Hallgren. Fast quantum algorithms for computing the unit group and class group of a number field. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 468–474. ACM, New York, 2005. URL `http://dx.doi.org/10.1145/1060590.1060660`. (Cited on page 38.)

[54] Bernard Harris. Probability distributions related to random mappings. *Ann. Math. Statist.*, 31:1045–1062, 1960. ISSN 0003-4851. (Cited on pages 76 and 93.)

[55] Huseyin Hisil, Kenneth K.-H. Wong, Gary Carter and Ed Dawson. Faster group operations on elliptic curves. In Ljiljana Brankovic and Willy Susilo, editors, *Australasian Information Security Conference (AISC 2009)*, volume 98 of *CRPIT*, pages 7–19, Wellington, New Zealand, 2009. ACS. URL `http://crpit.com/confpapers/CRPITV98Hisil.pdf`. (Cited on page 42.)

[56] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman and Wiliam Whyte. Hybrid lattice reduction and meet in the middle resistant parameter selection for NTRUEncrypt. IEEE P1363.1: Public-Key Cryptographic Techniques Based on Hard Problems over Lattices. URL `http://grouper.ieee.org/groups/1363/lattPK/submissions/ChoosingNewParameters.pdf`. (Cited on page 2.)

[57] Michael J. Jacobson, Jr. Applying sieving to the computation of quadratic class groups. *Math. Comp.*, 68(226):859–867, 1999. ISSN 0025-5718. URL `http://dx.doi.org/10.1090/S0025-5718-99-01003-0`. (Cited on pages 35 and 38.)

[58] Michael J. Jacobson, Jr. Computing discrete logarithms in quadratic orders. *J. Cryptology*, 13(4):473–492, 2000. ISSN 0933-2790. URL `http://dx.doi.org/10.1007/s001450010013`. (Cited on page 38.)

[59] Michael J. Jacobson, Jr., Shantha Ramachandran and Hugh C. Williams. Numerical results on class groups of imaginary quadratic fields. In *Algorithmic number theory*, volume 4076 of *Lecture Notes in Comput. Sci.*, pages 87–101. Springer, Berlin, 2006. URL `http://dx.doi.org/10.1007/11792086_7`. (Cited on page 88.)

[60] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34. Springer Berlin / Heidelberg, 2011. ISBN 978-3-642-25404-8. URL `http://dx.doi.org/10.1007/978-3-642-25405-5_2`. (Cited on page 8.)

[61] David Jao, Stephen D. Miller and Ramarathnam Venkatesan. Do all elliptic curves of the same order have the same difficulty of discrete log? In *Advances in cryptology—ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Comput. Sci.*, pages 21–40. Springer, Berlin, 2005. URL `http://dx.doi.org/10.1007/11593447_2`. (Cited on pages 3, 22, 37 and 68.)

[62] David Jao, Stephen D. Miller and Ramarathnam Venkatesan. Expander graphs based on GRH with an application to elliptic curve cryptography. *J. Number Theory*, 129(6):1491–1504, 2009. ISSN 0022-314X. URL `http://dx.doi.org/10.1016/j.jnt.2008.11.006`. (Cited on page 76.)

[63] David Y. Jao and Ramarathnam Venkatesan. Use of isogenies for design of cryptosystems. US patent 7499544, March 2009. URL `http://www.freepatentsonline.com/7499544.html`. (Cited on page 9.)

[64] David Y. Jao and Ramarathnam Venkatesan. Use of isogenies for design of cryptosystems. European patent EP1528705, April 2009. URL `http://www.freepatentsonline.com/EP1528705B1.html`. (Cited on page 9.)

[65] David Y. Jao, Peter L. Montgomery, Ramarathnam Venkatesan and Victor Boyko. Systems and methods for generation and validation of isogeny-based signatures. US patent 7617397, November 2009. URL `http://www.freepatentsonline.com/7617397.html`. (Cited on page 9.)

[66] Donald E. Knuth. *The Art of Computer Programming. Vol. 2. Seminumerical Algorithms. Third Edition.* Addison-Wesley, 1997. ISBN 0-201-89684-2. (Cited on page 76.)

[67] K. Ko, S. Lee, J. Cheon, J. Han, J.S. Kang and C. Park. New public-key cryptosystem using braid groups. In *Advances in Cryptology—CRYPTO 2000*, pages 166–183. Springer, 2000. (Cited on page 51.)

[68] Ann Hibner Koblitz, Neal Koblitz and Alfred Menezes. Elliptic curve cryptography: the serpentine course of a paradigm shift. *J. Number Theory*, 131(5):781–814, 2011. ISSN 0022-314X. URL `http://dx.doi.org/10.1016/j.jnt.2009.01.006`. (Cited on page 68.)

[69] Neal Koblitz and Alfred Menezes. Another look at "provable security". II. In Rana Barua and Tanja Lange, editors, *INDOCRYPT*, volume 4329 of *Lecture Notes in Comput. Sci.*, pages 148–175. Springer, 2006. ISBN 3-540-49767-6. URL `http://dx.doi.org/10.1007/11941378_12`. (Cited on pages 52 and 63.)

[70] Neal Koblitz and Alfred Menezes. Another look at "provable security". *J. Cryptology*, 20(1):3–37, 2007. (Cited on page 52.)

[71] David Kohel. *Endomorphism rings of elliptic curves over finite fields.* PhD thesis, University of California at Berkeley, 1996. (Cited on pages 3, 22, 29, 31, 32 and 69.)

[72] Caroline Kudla and Kenneth G. Paterson. Modular security proofs for key agreement protocols. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology.* LNCS, Springer-Verlag, 2005. (Cited on page 52.)

[73] Serge Lang. *Elliptic functions*, volume 112 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, second edition, 1987. ISBN 0-387-96508-4. With an appendix by J. Tate. (Cited on page 10.)

[74] James L. Massey. An introduction to contemporary cryptology. *Proceedings of the IEEE*, 76(5):533–549, 1988. (Cited on page 46.)

[75] James L. Massey and Jimmy K. Omura. Method and apparatus for maintaining the privacy of digital messages conveyed by public transmission. US patent 4567600, January 1986. URL `http://www.freepatentsonline.com/4567600.html`. (Cited on page 46.)

[76] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998. (Cited on page 82.)

[77] Gérard Maze, Chris Monico and Joachim Rosenthal. Public key cryptography based on semigroup actions. *Adv. Math. Commun.*, 1(4):489–507, 2007. ISSN 1930-5346. URL `http://dx.doi.org/10.3934/amc.2007.1.489`. (Cited on pages 26 and 51.)

[78] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press Series on Discrete Mathematics and its Applications. CRC Press, Boca Raton, FL, 1997. ISBN 0-8493-8523-7. With a foreword by Ronald L. Rivest. (Cited on pages 24 and 46.)

[79] Christopher J. Monico. *Semirings and semigroup actions in public-key cryptography*. PhD thesis, The Graduate School of the University of Notre Dame, Indiana, 2002. (Cited on pages 23, 25, 51 and 54.)

[80] Ravi Montenegro. A simple heuristic for complexity of birthday attacks. Unpublished preprint, 2011. (Cited on page 85.)

[81] NIST. Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revisited). NIST special publication 800-56A, National Institute for Standards and Technology, March 2007. (Cited on page 40.)

[82] Stephen C. Pohlig and Martin E. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Information Theory*, IT-24(1):106–110, 1978. ISSN 0018-9448. (Cited on page 46.)

[83] J. M. Pollard. A Monte Carlo method for factorization. *Nordisk Tidskr. Informationsbehandling (BIT)*, 15(3):331–334, 1975. (Cited on page 76.)

[84] J. M. Pollard. Monte Carlo methods for index computation (mod p). *Math. Comp.*, 32(143):918–924, 1978. ISSN 0025-5718. (Cited on pages 38 and 77.)

[85] C. Popescu. A secure authenticated key agreement protocol. In *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, volume 2, pages 783–786. IEEE, 2004. (Cited on pages 8 and 9.)

[86] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves, 2003. URL `http://arxiv.org/abs/quant-ph/0301141v2`. (Cited on page 2.)

[87] Anatoly P. Prudnikov, Yuriy A. Brychkov and Oleg I. Marichev. *Integrals and Series. Vol. 1. Elementary Functions. Second Edition.* Fizmatlit., Moscow, 2002. ISBN 5-9221-0323-7. (Cited on pages 95, 97, 98 and 102.)

[88] Anatol Rapoport. Cycle distributions in random nets. *Bulletin of Mathematical Biology*, 10:145–157, 1948. ISSN 0092-8240. (Cited on page 76.)

[89] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, report 2006/145, April 2006. URL `http://eprint.iacr.org/2006/145`. (Cited on pages 6, 7, 8, 22, 51, 56 and 68.)

[90] Alexander Rostovtsev, Elena Makhovenko and Olga Shemyakina. Elliptic curve ordered digital signature. Saint-Petersburg State Polytechnical University, April 2004. URL `http://www.ssl.stu.neva.ru/ssl/archieve/ordered_digital_signature.pdf`. Preprint. (Cited on pages 6 and 22.)

[91] Martin Rötteler. Quantum algorithms for highly non-linear boolean functions, 2008. URL `http://arxiv.org/pdf/0811.3208`. (Cited on page 39.)

[92] Reinhard Schertz. *Complex multiplication*, volume 15 of *New Mathematical Monographs*. Cambridge University Press, Cambridge, 2010. ISBN 978-0-521-76668-5. URL `http://dx.doi.org/10.1017/CBO9780511776892`. (Cited on page 10.)

[93] Arthur Schmidt. Quantum algorithm for solving the discrete logarithm problem in the class group of an imaginary quadratic field and security comparison of current cryptosystems at the beginning of quantum computer age. In Günter Müller, editor, *Emerging Trends in Information and Communication Security (ETRICS 2006)*, volume 3995 of *Lecture Notes in Comput. Sci.*, pages 481–493. Springer, 2006. ISBN 3-540-34640-6. (Cited on page 38.)

[94] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991. ISSN 0933-2790. URL `http://dx.doi.org/10.1007/BF00196725`. (Cited on page 43.)

[95] René Schoof. Quadratic fields and factorization. In *Computational methods in number theory, Part II*, volume 155 of *Math. Centre Tracts*, pages 235–286. Math. Centrum, Amsterdam, 1982. (Cited on page 76.)

[96] René Schoof. Counting points on elliptic curves over finite fields. *J. Théor. Nombres Bordeaux*, 7(1):219–254, 1995. ISSN 1246-7405. URL `http://jtnb.cedram.org/item?id=JTNB_1995__7_1_219_0`. (Cited on pages 3, 21 and 33.)

[97] E. Schulte-Geers. Collision search in a random mapping: some asymptotic results. Presentation at ECC 2000 (Essen, Germany), 2000. (Cited on page 82.)

[98] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. ISSN 0097-5397. URL `http://dx.doi.org/10.1137/S0097539795293172`. (Cited on pages 21 and 39.)

[99] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, report 2004/332, 2004. URL `http://eprint.iacr.org/2004/332`. (Cited on page 60.)

[100] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1986. ISBN 0-387-96203-4. (Cited on pages 8 and 30.)

[101] Joseph H. Silverman. *Advanced Topics in the Arithmetic of Elliptic Curves*, volume 151 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1994. ISBN 0-387-94328-5. (Cited on pages 28 and 29.)

[102] T. T. Soong. *Fundamentals of Probability and Statistics for Engineers*. John Wiley & Sons Inc., Hoboken, NJ, 2004. ISBN 0-470-86814-7. (Cited on pages 80, 82, 91, 99 and 100.)

[103] Anitha Srinivasan. Computations of class numbers of real quadratic fields. *Math. Comp.*, 67(223):1285–1308, 1998. ISSN 0025-5718. URL `http://dx.doi.org/10.1090/S0025-5718-98-00965-X`. (Cited on page 36.)

[104] Anton Stolbunov. ClassEll package, ver. 0.1. `http://www.item.ntnu.no/people/personalpages/phd/anton/software`, last visited 07/04/2011. (Cited on pages 4 and 87.)

[105] Anton Stolbunov. Public-key encryption based on cycles of isogenous elliptic curves, 2004. MSc thesis at Saint-Petersburg State Polytechnical University, in Russian. (Cited on page 6.)

[106] Anton Stolbunov. Reductionist security arguments for public-key cryptographic schemes based on group action. In Stig F. Mjølsnes, editor, *Norwegian Information Security Conference (NISK 2009)*, pages 97–109, Trondheim, Norway, 2009. Tapir Akademisk Forlag. (Cited on page 25.)

[107] Anton Stolbunov. Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. *Adv. Math. Commun.*, 4(2):215–235, 2010. ISSN 1930-5346. URL `http://dx.doi.org/10.3934/amc.2010.4.215`. (Cited on pages 68, 72, 76 and 82.)

[108] Anton Stolbunov. *Cryptographic Schemes Based on Isogenies*. PhD thesis, Norwegian University of Science and Technology (NTNU), 2012. This document. (Cited on pages 78, 80, 81 and 83.)

[109] John William Strutt [Lord Rayleigh]. On the resultant of a large number of vibrations of the same pitch and of arbitrary phase. *Philos. Mag.*, 10(60):73–78, 1880. (Cited on page 80.)

[110] John Tate. Endomorphisms of abelian varieties over finite fields. *Invent. Math.*, 2: 134–144, 1966. ISSN 0020-9910. (Cited on pages 29 and 67.)

[111] Edlyn Teske. On random walks for Pollard's rho method. *Math. Comp.*, 70(234):809–825, 2001. ISSN 0025-5718. URL `http://dx.doi.org/10.1090/S0025-5718-00-01213-8`. (Cited on pages 72 and 77.)

[112] Edlyn Teske. An elliptic curve trapdoor system. Cryptology ePrint Archive, Report 2003/058, 2003. URL http://eprint.iacr.org/2003/058. (Cited on page 6.)

[113] Edlyn Teske. An elliptic curve trapdoor system. *J. Cryptology*, 19(1):115–133, 2006. ISSN 0933-2790. URL http://dx.doi.org/10.1007/s00145-004-0328-3. (Cited on pages 6, 22, 37, 68 and 74.)

[114] Wim van Dam, Sean Hallgren and Lawrence Ip. Quantum algorithms for some hidden shift problems. *SIAM J. Comput.*, 36(3):763–778 (electronic), 2006. ISSN 0097-5397. URL http://dx.doi.org/10.1137/S009753970343141X. (Cited on page 39.)

[115] Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptology*, 12(1):1–28, 1999. ISSN 0933-2790. URL http://dx.doi.org/10.1007/PL00003816. (Cited on pages 72, 74, 76 and 93.)

[116] Thomas Wang. Integer hash function. http://www.concentric.net/~Ttwang/tech/inthash.htm, last visited 08/06/2010. (Cited on page 82.)

[117] Xiang Wang, WanSu Bao and XiangQun Fu. A quantum algorithm for searching a target solution of fixed weight. *Chinese Science Bulletin*, 56:484–489, 2011. ISSN 1001-6538. URL http://dx.doi.org/10.1007/s11434-010-4113-4. 10.1007/s11434-010-4113-4. (Cited on page 2.)

[118] Lawrence C. Washington. *Elliptic curves*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2003. ISBN 1-58488-365-0. (Cited on page 10.)

[119] William C. Waterhouse. Abelian varieties over finite fields. *Ann. Sci. École Norm. Sup. (4)*, 2:521–560, 1969. ISSN 0012-9593. (Cited on pages 29 and 70.)

[120] Han Weiwei and He Debiao. An authenticated key agreement protocol using isogenies between elliptic curves. In *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*, volume 1, pages 366 –369, march 2010. (Cited on pages 8 and 9.)

[121] E. T. Whittaker and G. N. Watson. *A course of modern analysis*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, 1996. ISBN 0-521-58807-3. Reprint of the fourth (1927) edition. (Cited on page 99.)