Computational aspects of class group actions and applications to post-quantum cryptography

Houben, M.R.

Summary

Cryptography is about securing information in such a way that only the intended parties have access to that information. For example, let us say that a hypothetical person, named Alice, would like to send a message to another hypothetical person, named Bob. If Alice and Bob had a way to communicate in such a way that no one else could overhear their conversation, then this would be easy; they could just converse in plain text. However, in the real world, a perfectly secure channel of communication is virtually impossible to guarantee, especially when communication happens over the internet. Somehow, Alice and Bob have to agree on some sort of code language. But how can they do that, if we assume malicious entities can listen in on all of their conversations? One way, is through something called a *Diffie–Hellman key exchange*; a method for two parties to establish a shared secret over a public communication channel. A common way, used by many end-to-end encrypted messaging applications, is based on mathematical objects called *elliptic curves*. An example of an elliptic curve is the collection of points (x, y) in the plane satisfying the equation $y^2 = x^3+3x^2-x-3$; see Figure 9.5.

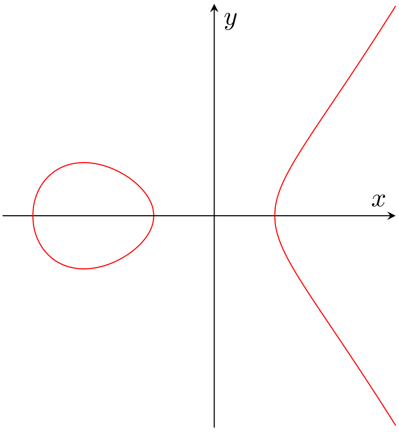

Figure 9.5: An elliptic curve given by the equation $y^2 = x^3 + 3x^2 - x - 3$.

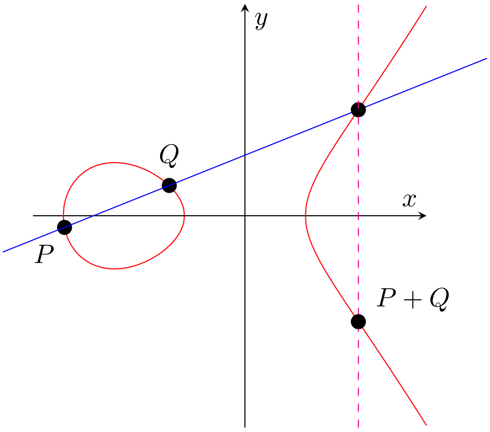

Figure 9.6: Adding two points on an elliptic curve.

What is special about elliptic curves, is that there is a geometric recipe to add points on the curve to each other, which is described as follows. When we would like to add two points P and Q on the curve, we draw the line connecting P and Q, which intersects the curve in exactly one other point.[3] The vertical line through this latter

[3] Unless the line is vertical; then we say $P + Q = O$, where O is called the *point at infinity*. If the

point intersects the curve in exactly one other point, which is $P+Q$; the sum of P and Q. In the case that $P = Q$, then we say that the line connecting P and Q is the tangent to the curve at P. This way, we have a recipe to compute $n \cdot P = \underbrace{P + P + \ldots + P}_{n \text{ times } P}$ for any integer $n > 0$. Such multiples can be computed in a faster way than just adding the point to itself $n-1$ times, through a procedure called *double-and-add*. For example, we can compute $20 \cdot P = 2 \cdot (2 \cdot ((2 \cdot (2 \cdot P)) + P))$ by just five additions (of which four are a doubling; i.e. adding a point to itself). Now, if Alice and Bob would like to establish a common secret, they could execute the following procedure.

(i) Alice and Bob agree publicly on a point P on an elliptic curve.

(ii) Alice and Bob generate (large) secret integers a and b.

(iii) Alice computes the point $P_A = a \cdot P$ and sends the result to Bob.

(iv) Bob computes the point $P_B = b \cdot P$ and sends the result to Alice.

(v) Using her secret and the point from Bob, Alice computes $a \cdot P_B = (a \cdot b) \cdot P$.

(vi) Using his secret and the point from Alice, Bob computes $b \cdot P_A = (a \cdot b) \cdot P$.

Since Alice and Bob both end up at the same point on the elliptic curve, they have successfully established a shared secret; that is, the *key echange* is complete. This common key can then be used to encrypt messages they would like to send to each other securely.

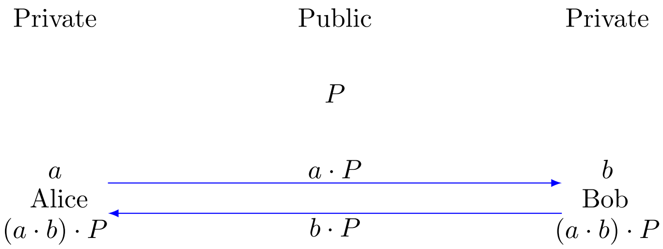

Figure 9.7: An Elliptic Curve Diffie–Hellman key exchange.

The security of this protocol relies on the assumption that it is impossible to recover Alice's secret a only using the publicly available information of P and $a \cdot P$. This is called the *discrete logarithm problem*. Theoretically, one would eventually be able to find a by computing $2 \cdot P = P + P$, $3 \cdot P = P + P + P$, $4 \cdot P = P + P + P + P$, and so on, until one eventually runs into $a \cdot P$. However, this is infeasible when a is really large; much slower than computing $a \cdot P$ given a and P by using the double-and-add method. Currently, no fast algorithms to solve the discrete logarithm problem

line is tangent to the curve in one of the points, then we count that intersection twice. In this way "the line through P and P" is the tangent to the curve at P.

in general are known. That is, unless we take into account *quantum computers*. A quantum computer is a special type of device that bases its computational power on the remarkable physical properties of subatomic particles. On such computers, there are known to exist fast algorithms to solve the discrete logarithm problem. To date, as far as we know, no one was able to build a quantum computer powerful enough to break any practically used cryptographic protocol. However, it is unclear whether such a device will be constructed in the near future. This has sparked a new area of research called *post-quantum cryptography*, which searches for ways to encrypt information that are secure against attacks by quantum computers. One such proposal is called *isogeny-based cryptography*. Isogenies are maps between elliptic curves; a type of transformation that takes you from one elliptic curve to the other. When chosen in a smart way, such maps can be used to establish a key exchange as before. This time, Alice and Bob publicly agree, not on a point on an elliptic curve, but on an elliptic curve itself. They apply successive transformations to the curve in such a way that they end up at the same elliptic curve, which then forms their shared secret. Abstractly, this is pictured in Figure 9.8.

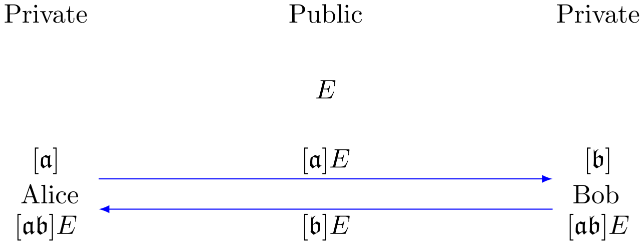

Figure 9.8: An isogeny-based key exchange protocol.

The assumption underlying the security of isogeny-based cryptography, is that it is difficult, given two elliptic curves E_1 and E_2, to find a transformation from E_1 to E_2. This is called the *isogeny path problem*. It is assumed that this problem is difficult even for quantum computers.

In this thesis, we consider several computational problems associated to isogenies between elliptic curves.

Chapters 1, 2, and 3 are introductory and end with a high-level overview of the main results presented in later chapters.

In Chapter 4 and 5, we show how, in certain instances, maps on elliptic curves called *pairings* can be used to disprove computational hardness assumptions related to isogeny-based cryptography. In special cases, we find efficient solutions to the isogeny path problem, as well as to a weaker problem known as the *Decisional Diffie–Hellman Problem*.

In Chapter 6, we develop a multivariate generalization of *Hilbert class polynomials*; polynomials that encode elliptic curves with a certain structure (given by their *endomorphism ring*). We in particular discuss the computational benefits of these novel polynomials compared to previously known class polynomials.

In Chapter 7, we study a method to compute chains of isogenies efficiently through equations called *radical isogeny formulae.* We develop a new method to obtain such formulae, and improve on the efficiency of their evaluation. This leads to a speed-up in the execution of certain isogeny-based cryptographic protocols.