

# A Taste of Elliptic Curve Cryptography

Shrenik Shah<sup>†</sup>

Harvard University '09

Cambridge, MA 02138

sshah@fas.harvard.edu

## Abstract

This paper develops several classical algorithms and cryptosystems in cryptography, and develops the theory of elliptic curves to reveal the improvements provided by elliptic curve cryptography. The prerequisites to this paper are an understanding of groups, fields, and some elementary number theory.

## 2.1 Introduction

Elliptic curve cryptography is not only a surprising application of a deep and powerful area of number theory to computer science, but as we shall see, is also a very practical technique that is used today around the world. In this article we present a small taste of cryptography, presenting some classical cryptographic protocols and factorization methods. After this we describe some of the mathematical theory of elliptic curves. These are combined in a section that shows how to use elliptic curves in the earlier protocols, and explains why these represent an improvement over traditional methods, while citing some limitations to this viewpoint.

It is impossible, of course, to give anything approximating a complete account of these subjects in a short article. Thus I strive to give an overview of the structure of these fields by including a few examples from cryptography and developing the theory needed to present the corresponding elliptic curve cryptosystems. Moreover, the paper is intended for one who is familiar with the basic properties of groups and fields, as well as elementary number theory, but with no exposure to cryptography or elliptic curves. Section 2.4 together with Section 2.5.2.2 and Section 2.5.4 sketch the proofs of some key results that follow from the existence of the Weil pairing, an algebraic structure defined on an elliptic curve, and require some additional mathematical maturity. Since Section 2.3 makes no use of Section 2.2, some readers may want to begin with Section 2.3, and use Section 2.2 as a reference when reading Sections 2.4 and 2.5.

## 2.2 Cryptography

At the heart of cryptography is security—cryptography allows one to very carefully control the information and powers available in a system to various parties in a way that cannot be manipulated or broken by dishonest participants. Thus cryptosystems are often tested for resistance to various “attacks” and required to have a very small probability of being broken. Most of classical cryptography proves results that are conditional on central assumptions. These assumptions are made to be general enough so that they do not rely on the difficulty of a specific problem, such as factoring integers. However, the central assumptions are far from being proven, so cryptography in some sense rests on fragile ground. This section will describe these assumptions as well as some types of cryptosystems that can be constructed under these assumptions. It will also present classical cryptosystems as concrete examples. These cryptosystems will then be modified to use elliptic curves later in this paper. The last subsection will explain some of the mathematics behind the problem of

---

<sup>†</sup>Shrenik Shah, Harvard '09, is a mathematics concentrator and English minor. He is also enrolled in a concurrent masters program in computer science. He is a founding member of The HCMR and currently serves as Articles Editor.

factoring integers and describe classical algorithms that aim to perform this task more efficiently than the usual brute force methods.

## 2.2.1 Theory of Computation

The most common formalization for the notion of a “computer” is the Turing machine. For our purposes, we will instead refer nonrigorously to the notion of an **algorithm** and assume that all basic operations, such as addition and multiplication, all take a single time step. These assumptions serve our purposes because we will not analyze the precise complexity of the algorithms we study. The time an algorithm takes is measured as a function of the input size, where the input is a string of binary digits. For the interested reader, [Si] is a good introduction to the formal theory of computation.

An algorithm can either take as an input a string in binary and output “accept” or “reject,” or take in a string in binary and output another string in binary. An algorithm of the latter sort is said to be computing a function. Any subset of  $\{0, 1\}^*$ , the set of all binary strings, is termed a **language**. The subset of  $\{0, 1\}^*$  accepted by an algorithm  $A$  is a language said to be **decided** by  $A$ . The time  $t(n)$  taken by the algorithm is measured as the maximum number of time steps for an input of length  $n$ . The class **P** consists of all languages that can be decided by an algorithm that runs in **polynomial time**, meaning that  $t(n) \leq p(n)$  for some polynomial  $p$ .

Some languages have the property that there exists some string (specific to a particular string  $x \in L$ ) with which membership in the language can be verified quickly. For example, those familiar with basic graph theory will recall the definition of a **Hamiltonian path**, which is a path that visits every vertex exactly once. It is conjectured to be a difficult computational problem to determine, for a given graph  $G$ , whether such a path exists. However, given such a path, an algorithm can very easily verify that it is indeed a Hamiltonian path. The description of the path is called a **witness** testifying to the existence of a Hamiltonian path for  $G$ .

The class **NP** consists of all languages  $L$  with the following property: There exists a polynomial-time algorithm  $A$  such that if  $x \in L$ , then there exists a witness  $w$  such that  $A(x, w) = 1$ . However, if  $x \notin L$ , then  $A(x, w) = 0$  for all choices of  $w$ . It is a very well-known open problem to determine whether **P** and **NP** are equal (or not).

In fact, the problem of determining whether a Hamiltonian path exists is called **NP-complete** because it can be proven that for every language  $L$  in the class **NP**, there is a polynomial time algorithm that maps any  $x \in L$  to graphs that have a Hamiltonian path and  $x \notin L$  to graphs that have none. Thus, if a polynomial time algorithm were ever written to determine whether a Hamiltonian path exists, an algorithm could be then developed to decide any language in **NP**. By contrapositive, if any **NP** program were ever proven to have no polynomial time algorithm, then testing for existence of a Hamiltonian path could not be done in polynomial time either.

It seems, then, that proving that **P**  $\neq$  **NP** would be rather powerful, in that it shows that every **NP-complete** problem is difficult to solve. Unfortunately, this difficulty is **worst-case hardness**, which means that for every algorithm, all that is known is that there exists an input on which it is wrong, not that no algorithm can solve the problem for most inputs. Thus, although the idea of basing cryptography on the assumption that **P**  $\neq$  **NP** is an attractive notion, especially since these classes seem “surely” different, a much stronger assumption is needed, as will be seen in the next section.

The last notions needed are merely some technical definitions: A function  $\nu : \mathbb{N} \rightarrow \mathbb{R}$  is **negligible** if for any exponent  $\alpha \geq 0$  there exists a constant  $c_\alpha$  such that  $n^{-\alpha} > \nu(n)$  for  $n > c_\alpha$ . Roughly, if  $\nu$  eventually shrinks faster than any inverse polynomial, then  $\nu$  is negligible. We define the **big-O notation**  $f(n) = O(g(n))$  to mean that there exists  $N \in \mathbb{N}$  and  $C > 0$  such that for  $n > N$ ,  $|f(n)| \leq C|g(n)|$ , and  $f(n) = \Theta(g(n))$  to mean that there exists  $c, C > 0$  such that  $c|g(n)| \leq |f(n)| \leq C|g(n)|$ , again for  $n > N$ .

Finally, a **probabilistic polynomial time** (PPT) algorithm is an algorithm with the additional property that it may generate uniform random bits whenever it wishes. In other words, a PPT algorithm may, in addition to following deterministic instructions, flip a fair two-sided coin.

### 2.2.2 Central Assumptions

This section details the main assumptions of cryptography. These definitions are taken from [GB]. The main assumption of cryptography is that one-way functions exist. A **one-way function**  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  has two properties:

- There is a polynomial-time algorithm that computes the one-way function.
- For any PPT  $A$ , on a randomly chosen input  $x \in \{0, 1\}^n$  with uniform distribution, there exists a negligible function  $\nu_A$  (allowed to depend on  $A$ ) such that the probability that  $A(1^n, f(x)) = z$  where  $f(z) = f(x)$  is less than or equal to  $\nu_A(n)$  (for  $n$  sufficiently large). (Note that  $1^n = 1 \dots 1$  with 1 repeated  $n$  times.)

The second property above is a mouthful, so we'll satisfy ourselves with an imprecise definition: given the output on a randomly chosen input of a one-way function, a polynomially bounded algorithm has a low probability of finding an input that would produce the same output. Note that the  $1^n$  in the input to  $A$  is to ensure that its running time is based on the size of the space of possible  $x$ , rather than on the output  $f(x)$ , which could be much smaller.

The existence of one-way functions implies that  $\mathbf{P} \neq \mathbf{NP}$ , but the converse is not true. The field of average-case complexity strives to prove such an equivalence. Unfortunately, many of the results related to this question are pessimistic, and imply that proving an equivalence would require fairly sophisticated, non-intuitive methods.

Some candidate one-way functions:

- Computing the product  $n = pq$  of two prime numbers  $p, q$ . We will discuss later in this paper some algorithms faster than the naïve  $O(\sqrt{n})$  algorithm for factoring such numbers.
- The **discrete logarithm problem**: Given the multiplicative group  $(\mathbb{Z}/p\mathbb{Z})^\times$ ,  $g^x$ , and  $g$ , where  $g$  is a generator of this group, determine  $x$ .
- Computing modular square roots: Given a quadratic residue  $a \bmod n$ , compute a value  $x$  such that  $x^2 \equiv a \bmod n$ .

As a notational remark, we will use, as above,  $a \equiv b \bmod n$  for  $n \mid (a - b)$ . We will frequently use  $=$  for  $\equiv$ , particularly when working in the ring  $\mathbb{Z}/n\mathbb{Z}$ . We will also use  $(a, b)$  for the greatest common divisor of  $a$  and  $b$ .

Another problem that is conjectured to be hard is the **Diffie-Hellman problem**. In  $(\mathbb{Z}/p\mathbb{Z})^\times$ , given the generator  $g$  together with  $g^x$  and  $g^y$ , the problem is to compute  $g^{xy}$ . This problem is important in several protocols that we will discuss later. The hardness of this problem is frequently assumed by cryptographers. This is known as the **Diffie-Hellman assumption**. If one has a solution to the discrete logarithm problem for *either*  $g, g^x$  or  $g, g^y$ , one can certainly compute  $g^{xy}$ , so the Diffie-Hellman assumption is *stronger* than the assumption that computing discrete logarithms is hard.

As it turns out, cryptographers do not yet know how to create certain encryption schemes (discussed later) without an even stronger assumption. This is that a **trapdoor function**  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  exists, with the following properties:

- The function  $f$  is one-way.
- There exists a PPT algorithm  $T$  so that for every input length  $n$ , there exists a  $t_n \in \{0, 1\}^*$  of length bounded by a polynomial  $p$ , and for all  $x \in \{0, 1\}^n$ ,  $T(f(x), t_n) = x$  with the property that  $f(z) = f(x)$ .
- It is also important that a trapdoor function can be generated together with its trapdoors  $t_n$  efficiently.

Trapdoor functions essentially have a built in method to invert the function, which is important if one party computes the function at a point, while another needs to invert this computation in order to access information. This occurs in **public key encryption**, described in the next section.

### 2.2.3 Types of Cryptographic Schemes

**Cryptographic schemes** are simple tasks that represent common tasks requiring security against a very specific breach. This is in contrast to **cryptographic protocols**, which are often rather complex, and composed of many different schemes. Election systems and auction systems are examples of such protocols. In this section, we describe a few of the most common schemes in a nonrigorous fashion, followed by examples. Texts such as [Go1] and [GB] are excellent references for those interested in a formal treatment.

#### 2.2.3.1 Public Key Encryption

In a public key encryption scheme, Alice wants to send a message  $M$  to Bob, while an eavesdropper Eve is listening over the channel. Bob generates a secret decryption key  $d$  and publishes a public key  $e$ . Alice computes the encryption  $C = \text{Enc}_e(M)$  and sends it to Bob. Bob receives  $C$  and computes  $M = \text{Dec}_d(C)$ . We require various minimal properties of this encryption system:

- There should be some efficient algorithm to generate pairs  $(d, e)$  of a private key and associated public key.
- The algorithms  $\text{Enc}_e(\cdot)$  and  $\text{Dec}_d(\cdot)$  should be efficient.
- Knowing  $e$  and  $C$  should not reveal information about  $M$ .

In fact, we can define more properties that guard against more subtle attacks. For example, seeing many messages sent across this channel should not reveal additional information about any of the messages. The **one-time pad** discussed in the next section will illustrate the importance of this requirement; it is aptly named, as it should only ever be used once. Another example is **non-malleability**, and requires that an eavesdropper cannot meaningfully modify the message during transmission.

**One-time Pad.** This does not fit any of the above cryptographic schemes, but is of great historical significance, predating modern cryptography by over half a century. In these earlier times, actual books of bits would be distributed physically, and the sender would indicate which bits were to be used for the decryption. The sheet used would be destroyed after use. The idea is that Alice wants to send a message  $M \in \{0, 1\}^n$  to Bob, and they have earlier agreed upon a secret one-time pad  $s \in \{0, 1\}^n$ . Alice sends  $C = M \oplus s$  to Bob, where  $\oplus$  denotes the **exclusive-or** operation, which is defined as bitwise addition modulo 2 of two given binary strings. For example, for any string  $t \in \{0, 1\}^n$ ,  $t \oplus t = 0^n$ . Bob computes  $M = C \oplus s = M \oplus s \oplus s$ , and both Alice and Bob destroy all traces of  $s$ . If Eve knows no information about  $s$ , she knows absolutely nothing about  $M$ , even if she has  $C$ .

The “one-time” property, however, is critical. If Alice sends another message  $M'$  with the same key  $s$ , then Eve now might have both  $C = M \oplus s$  and  $C' = M' \oplus s$ . By computing  $C \oplus C' = M \oplus M'$ , Eve gains some information about the messages  $M$  and  $M'$ . Using some algorithm that perhaps takes advantage of the sparseness of the English language, Eve could possibly decipher  $M$  or  $M'$  from this information.

**RSA Public Key Encryption.** In the RSA (Rivest, Shamir, Adleman) cryptosystem, Alice chooses two large primes  $p, q$  and releases the number  $n = pq$  and an exponent  $e$  as her public key, so that  $(e, \varphi(n)) = 1$ . (We define the **Euler totient function**  $\varphi$  by  $\varphi(n) = |\{a \mid (n, a) = 1, 0 \leq a < n\}| = |(\mathbb{Z}/n\mathbb{Z})^\times|$ .) Then she computes  $d = e^{-1} \pmod{\varphi(n)}$  as her private key. Bob wants to send the message  $M$  to Alice, and encrypts it using the function  $C = \text{Enc}_{n,e}(M) = M^e \pmod{n}$ . Finally, Alice simply computes  $M = \text{Dec}_d(C) = C^d = M^{ed} = M^{k\varphi(n)+1} = M \pmod{n}$  by Euler’s theorem.

Unfortunately, RSA is vulnerable to some attacks. For example, if the same message is sent using  $e$  different choices of  $n$ , the message can be decrypted via simple Chinese remaindering. This is particularly dangerous when a small  $e$  (like 3) is chosen for efficiency purposes. A solution is to pad all messages with random bits at the end, chosen differently with every encryption. Another vulnerability is that if the same message is sent using two relatively prime choices of  $e$  for the same value of  $n$ , the message can be decrypted using the Euclidean algorithm. A weakness that renders

RSA relatively useless for the purposes of identity-based cryptography is that knowing a pair  $e, d$  of encryption and decryption keys allows one to factor  $n$ , which we prove in Corollary 5. See [Bo] for an excellent survey of the attacks on this cryptosystem.

**Elgamal Public Key Encryption.** In Elgamal Public Key encryption, Alice chooses a multiplicative group  $\mathbb{F}_p^\times$ , a generator  $g$  of this group, and a secret integer  $x$ . She computes  $g_A = g^x$ , and publishes as her public key  $(\mathbb{F}_p^\times, g, g_A)$ . To encrypt  $0 < M < p$ , Bob sends his message by picking a uniformly random secret integer  $0 < y < p - 1$  and computing and sending  $C_1 = g^y$ ,  $C_2 = Mg_A^y = Mg^{xy}$  to Alice, who decrypts by computing  $C_2C_1^{-x} = Mg^{xy}g^{-xy} = M$ .

The key observation is that all of the operations above could be replicated with any cyclic group in place of  $\mathbb{F}_p^\times$  above. We will later replace this with a group structure associated to an elliptic curve.

Cryptographically, this system has a key weakness. The pair  $(C_1, C_2)$  relate to  $M$  by the simple formula  $M = C_2C_1^{-x}$ . Thus, if Eve has the ability to modify the message as it is sent to Bob, she can change  $M$  to the known function of  $M$ ,  $f_k(M) = kM$ , for some  $k$ , without Bob knowing that any change had been made. Some cryptosystems make it difficult for Eve to control exactly how her modification might affect the sent message—this property is called **non-malleability**. RSA is also malleable, since an encryption of  $M^2$  can be computed from an encryption of  $M$ .

**Theorem 1.** *If the Diffie-Hellman problem is intractible, then Elgamal Public Key Encryption is unbreakable.*

*Proof.* Suppose for contradiction that given public key  $g^x$ , and message  $(g^y, D)$  (even for only one such  $D$ ), one could compute  $M = g^{-xy}D$ . Then one could compute  $M^{-1}D = g^{xy}D^{-1}D = g^{xy}$ , violating the intractibility of the Diffie-Hellman problem.  $\square$

### 2.2.3.2 Identity-based Encryption

A weakness of public key encryption is that an adversary can pretend to be Bob, the intended recipient, and request the message to be encrypted in his public key instead of Bob's—he could then decrypt the message received. To solve this problem, one might try to design a trusted database that holds everyone's public keys, but even then, the adversary can intercept communications with the database. The solution provided by **identity-based encryption**, proposed by Shamir in [Sham], is to use some identity-based value that is a function of some unique information about Bob that anyone sending Bob a message would know. As an example, one might use the output of a publicly known hash function (defined in Section 2.2.3.5) on Bob's name, cell phone number, and address as an identifier. A trusted server provides a decryption key to Bob on some occasion in an authenticated manner. Bob can then decrypt messages forever afterwards.

### 2.2.3.3 Digital Signatures

In a digital signature scheme, Alice wants to sign a document  $M$  in a way that is verifiable and unforgeable. She publishes a public key  $e$  that anyone can see, and keeps a secret key  $s$ . She signs the message with the function  $C = \text{Sig}_s(M)$ , and sends  $(M, C)$  to the desired recipients. Anyone can run the verification algorithm  $\text{Ver}_e(M, C)$  to test whether the message was indeed signed by Alice.

There are several levels of information we can assume an adversary might have. In the weakest case, the adversary Eve might know only the public key  $e$ , with no examples. More realistically, she might know pairs  $(M, C)$  that were produced by Alice earlier. The worst case is that Eve may have forced Alice to sign certain documents of her choice in her efforts to forge a signature on a document that Alice has not yet signed.

We also can place different requirements on possible forgeries, but for the purposes of this paper, we will use the strongest possible requirement, that of **existential unforgeability**: Eve should not be able to sign any message for which she has not yet seen a signature. There are weaker levels of security than this, detailed in [GB].

We finally require, naturally, that  $\text{Sig}_s(\cdot)$  and  $\text{Ver}_e(\cdot, \cdot)$  should be efficiently computable.

In practice, digital signatures are very frequently attached to emails and other forms of electronic correspondence.

**Digital Signature Algorithm.** Recall that in a digital signature scheme, Alice wants to equip a document  $M$  with an unforgeable signature. The setup for this algorithm is more complex. Alice needs to pick a group  $\mathbb{F}_q^\times$  and a large prime  $p$  such that  $p \mid q - 1$ , and  $\frac{q-1}{p} = e$  is as small as possible. The message will be in the range  $0 < M < p$ . She also picks an  $g \in \mathbb{F}_q^\times$  such that  $g$  has order exactly  $p$ . By random guesses, exponentiated by a factor of  $\frac{q-1}{p}$ , this can usually be done quickly. Note that if  $q = 2p + 1$ , meaning that  $p, q$  are a pair of Sophie-Germaine primes, then then one can use any nontrivial quadratic residue for  $g$ , since the group of residues modulo  $q$  is then of order  $p$ . Alice also picks a publicly known hash function  $H : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$ , randomly chooses a secret integer  $0 < a < p$ , computes  $g_A = g^a$ , and publishes  $(\mathbb{F}_q^\times, p, g, g_A, H)$ .

To sign  $M \in \{1, \dots, q - 1\}$ , Alice computes  $H(M)$ , picks a random integer  $0 < k < p$ , and computes  $x = g^k \bmod p$  (in  $\mathbb{F}_q^\times$  first, then reduced modulo  $p$ ) and  $y = k^{-1}(H(M) + ax) \bmod p$ , producing a signed document  $(M, x, y)$ . The verifier Bob computes  $c_1 = y^{-1}H(M) \bmod p$ ,  $c_2 = y^{-1}x \bmod p$ , and accepts if  $x = g^{c_1}g_A^{c_2} \bmod p$ . A correctly signed document is always accepted, since  $x = g^k = g^{y^{-1}(H(M)+ax)} = g^{c_1+ac_2} \bmod p$ .

This algorithm is frequently used in practice, though it is in fact an open problem to prove that the hardness of breaking this cryptosystem follows from the intractability of the discrete logarithm problem. On the other hand, the next section describes a provably existentially unforgeable protocol to compute digital signatures.

**Elgamal Digital Signatures.** Let  $q = 2p + 1$  be a pair  $p, q$  of Sophie-Germaine primes, and  $G$  the group of squares in  $\mathbb{Z}_q^\times$ . Fix a generator  $g$  of  $G$ . Alice fixes a private key  $x$  randomly selected from  $\{0, \dots, p - 1\}$ , computes  $g_A = g^x$ , and publishes the public key  $(p, g, g_A)$ .

To sign  $M \in \{1, \dots, q - 1\}$ , Alice picks  $y$  randomly selected from  $\{0, \dots, p - 1\}$ , and computes  $h = g^y$ . Then Alice computes  $H(M\|y)$ , where  $\|$  denotes concatenation, and  $c = -xH(M\|h) - y \bmod p$ . Finally Alice publishes  $\text{SIG}_A(M) = (M, h, c)$ .

Verification simply requires checking that  $g_A^{H(M\|h)}g^ch = 1$ .

**Theorem 2.** *If computing the discrete logarithm  $x$  of  $g^x$  is hard, and the hash function is a random oracle, then Elgamal Digital Signatures are existentially unforgeable.*

*Proof.* If an adversary were to compute  $(M, h, c)$  meeting the requirements for a fixed message  $M$ , then it would be necessary for the adversary to have obtained  $H(M\|h)$  from the oracle, otherwise the value of  $g_A^{H(M\|h)}g^ch$  would be a random value. If the adversary knows  $H(M\|h)$ , this implies that  $M$  and  $h$  are fixed, since it is difficult (in the random oracle model) to find another message such that  $H(M\|h)$  is the same. To forge a single message  $M$ , one must be able to determine a value  $c$  and a value  $c'$  for at least two possible values  $a, a'$  for  $H(M\|y)$ , since  $H(M\|y)$  is a random value. Thus, the adversary has  $g_A^ag^c h = g_A^{a'}g^{c'} h$ , or  $g^{x(a-a')} = g^{c-c'}$ . The number  $a - a'$  has an inverse modulo  $p$  since it is nonzero, so  $g^x = g^{(c-c')(a-a')^{-1}}$  yields  $x \equiv (c-c')(a-a')^{-1} \bmod p$ , thus solving the discrete logarithm problem that was assumed hard.  $\square$

### 2.2.3.4 Key Exchange

Diffie and Hellman developed the notion of a **key exchange**, wherein Alice and Bob have no prior shared secret, but they want to be able to send messages via private key cryptography (a cryptographic scheme we will not discuss in this paper). The desired property is that Alice and Bob both compute the same value, and that an eavesdropper Eve is unable to determine anything about that value with high probability. There is a natural generalization to  $n$ -partite key exchanges, where  $n$  trusted players want to agree on a single key.

This seems fairly straightforward, though there are some interesting issues that arise. With just these properties, Eve could impersonate Alice and share a key with Bob. Bob, thinking that Eve is Alice, will use the shared key to encrypt messages, which Eve can then read. To counter an attack like this, one needs to use authentication, a topic discussed in detail in [GB] and [MvO].

From the Diffie-Hellman problem described above, it is not difficult to guess their protocol for the key exchange. A group  $\mathbb{F}_p^\times$  with generator  $g$  is fixed ahead of time. Alice picks  $a \in \mathbb{F}_p^\times$

at random, and Bob similarly picks  $b$ . Alice sends  $g^a$  to Bob, who sends  $g^b$  to Alice. They both exponentiate to compute  $g^{ab}$ , which is the shared key.

There is an  $n$ -partite version of this protocol, again with a publicly known field  $\mathbb{F}_p^\times$  with generator  $g$ . Players  $p_1, \dots, p_n$  pick secrets  $s_1, \dots, s_n$ . In round  $k$ ,  $k = 0, \dots, n-2$ ,  $p_i$  sends  $g^{s_i s_{j_1} \cdots s_{j_k}}$  to all players, where  $j_1, \dots, j_k$  are distinct elements of  $\{1, \dots, i-1, i+1, \dots, n\}$ . The players then use  $g^{s_1 \cdots s_n}$  as their shared secret.

### 2.2.3.5 Cryptographic Hash Functions

Hash functions are a useful tool, often used within the earlier-mentioned cryptosystems. A hash function  $H : \{0, 1\}^* \rightarrow \{0, 1\}^*$  sends any string of arbitrary length to its **hash**, of length polynomial in the length of the input (though often of constant length, in practice). In the usual definition of a **collision-free hash function**, it should be intractable to find a **collision** in time polynomial in the input length  $k$ , where a collision is a pair of strings  $s_1, s_2$  such that  $H(s_1) = H(s_2)$ .  $H$  is usually required to be a one-way function.

The **random oracle model** of a hash function works as follows: given a message  $M \in \{0, 1\}^*$ , the random oracle always outputs a random string in  $\{0, 1\}^k$ , except when a previous query is repeated, in which case it produces the original output. When using a hash function in a cryptosystem, one sometimes proves results about the security of the cryptosystem by assuming that the hash function is a random oracle. The assumption that a hash function  $H$  is indistinguishable from a random oracle is stronger than assuming that  $H$  is a collision-free hash function.

### 2.2.4 Miller-Rabin and Pollard's Algorithm

In a sense that the following theorem makes precise, the group  $(\mathbb{Z}/n\mathbb{Z})^\times$  contains all the information about the factorization of  $n$ , and this information can be extracted efficiently knowing very little about this group. In fact, simply knowing a reasonably small multiple of its order suffices to factor  $n$  in polynomial time.

**Theorem 3.** *Suppose that we know  $n$  and  $k\varphi(n)$  for some positive integer  $k$ . Assume also that  $\log k = \log^{O(1)} n$ . Then  $n$  can be factored efficiently.*

*Proof.* It suffices to find a single factor, because given a factorization  $n = n_1 \cdot n_2$ , one can run the algorithm again on  $n_1$  and  $n_2$ , as  $\varphi(n_i) \mid \varphi(n)$  (and one can argue that the run time of this recursive algorithm is still polynomial). One can efficiently check whether  $n$  is a prime power, so assume this is not the case.

In this case, we can design an algorithm as follows. Given a composite input  $n$ , we first factor  $k\varphi(n) = 2^\ell m$  where  $\ell \in \mathbb{Z}$  is chosen maximally such that  $m \in \mathbb{Z}$ . Next, we pick a random integer  $a$ , where  $1 < a < n$ , and compute  $(a, n)$ . If  $(a, n) \neq 1$  then we are done, since in this case we have found a nontrivial factor of  $n$ . Otherwise,  $(a, n) = 1$  and

$$a^{\varphi(n)} \equiv a^{k\varphi(n)} \equiv a^{2^\ell m} \equiv 1 \pmod{n}.$$

Thus, if we consider

$$a^m, a^{2m}, a^{2^2 m}, \dots, a^{2^\ell m} \pmod{n}$$

we have a sequence that eventually becomes 1. For fixed  $a$ , consider the largest value of  $j$  such that  $a^{2^j m} \not\equiv 1 \pmod{n}$ . In Lemma 4, we show that for at least  $\frac{1}{2}$  of the possible choices of  $a$ , we have that  $a^{2^j m} \not\equiv -1 \pmod{n}$ . Then,  $a^{2^{j+1} m} - 1 \equiv (a^{2^j m} + 1)(a^{2^j m} - 1) \equiv 0 \pmod{n}$ . Thus,  $(a^{2^j m} + 1)(a^{2^j m} - 1) = sn$  for some integer  $s$ . Also, neither of the factors on the left are 0 or multiples of  $n$ , since we required  $a^{2^j m} \not\equiv -1 \pmod{n}$ . So we can compute  $(a^{2^j m} + 1, n)$  and  $(a^{2^j m} - 1, n)$  to obtain a factorization of  $n$  into  $n_1 \cdot n_2$ , as desired. By choosing random values for  $a$  until we find one that yields a factorization of  $n$ , this algorithm terminates in expected polynomial time.  $\square$

**Lemma 4.** *In the proof of Theorem 3 above, at least  $\frac{1}{2}$  of our choices of  $a$  have the property that there exists  $j$  such that  $a^{2^j m} \not\equiv 1, -1 \pmod n$  while  $a^{2^{j+1} m} \equiv 1 \pmod n$ . Moreover, we will use only the fact that  $a^{2^\ell m} \equiv 1 \pmod n$  for all  $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ .*

*Proof.* We adapt the proof of correctness of the Miller-Rabin primality test from [CLRS]. Define a pair  $(a, j)$  of integers to be **bad** if  $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ ,  $j \in \{0, 1, \dots, \ell\}$ , and  $a^{2^j m} \equiv -1 \pmod n$ . The desired result translates into proving that the number of bad pairs  $(a, j)$  is at most  $\frac{1}{2}|(\mathbb{Z}/n\mathbb{Z})^\times|$ , since for any  $a$  of the form described in the statement of the lemma, there is a unique value of  $j$  such that  $(a, j)$  is bad. Note that since  $m$  is odd,  $(n-1, 0)$  is bad. Thus there exists at least one bad pair, so we may pick the largest possible  $j$  such that there is a bad pair  $(a, j)$ , and fix a value of  $a$  so that  $(a, j)$  is bad. Note that  $j < \ell$  since, by assumption,  $a^{2^\ell m} \equiv 1 \pmod n$  for all  $a$ . Let

$$S = \left\{ x \in (\mathbb{Z}/n\mathbb{Z})^\times \mid x^{2^j m} \equiv \pm 1 \pmod n \right\}.$$

This set is closed under multiplication modulo  $n$ , so it is a subgroup of  $(\mathbb{Z}/n\mathbb{Z})^\times$ . Every bad pair  $(a, j)$  has  $a$  a member of  $S$ , because we picked  $a$  to be maximal and we allow  $x^{2^j m} \equiv \pm 1 \pmod n$  in the definition of  $S$ . If  $S$  also contained numbers  $a$  such that  $(a, j)$  is not bad, this would be fine, as we are only proving a bound on the number of bad pairs.

We now prove that  $S \neq (\mathbb{Z}/n\mathbb{Z})^\times$ . Note that  $n$  by assumption is not a prime power, so  $n = n' \cdot p^\alpha$  for some prime  $p \mid n$ , where  $\alpha$  is chosen maximally with  $p^\alpha \mid n$ , and  $n' \neq 1$ . Since  $a^{2^j m} \equiv -1 \pmod n$ , we have  $a^{2^j m} \equiv -1 \pmod{n'}$  by the Chinese Remainder Theorem, as  $(n', p^\alpha) = 1$ . Moreover, again by this theorem, there exists  $b$  such that  $b \equiv a \pmod{n'}$ ,  $b \equiv 1 \pmod{p^\alpha}$ . Thus, by our preceding calculation,  $b^{2^j m} \equiv -1 \pmod{n'}$ ,  $b^{2^j m} \equiv 1 \pmod{p^\alpha}$ . By the Chinese Remainder Theorem,  $b^{2^j m} \not\equiv 1 \pmod{n'}$  implies  $b^{2^j m} \not\equiv 1 \pmod n$ , and  $b^{2^j m} \not\equiv -1 \pmod{p^\alpha}$  implies  $b^{2^j m} \not\equiv -1 \pmod n$ . Thus  $b^{2^j m} \not\equiv \pm 1 \pmod n$ , so  $b \notin S$ .

It suffices, then, to show that  $b \in (\mathbb{Z}/n\mathbb{Z})^\times$ . Note that since  $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ ,  $(a, n) = 1$ , so  $(a, n') = 1$ . Since  $b \equiv a \pmod{n'}$ ,  $(b, n') = 1$ . Also by definition,  $b \equiv 1 \pmod{p^\alpha}$ , so  $(b, p^\alpha) = 1$ . Thus,  $b$  is relatively prime to both  $n'$  and  $p^\alpha$ , so  $b$  is relatively prime to their product,  $n$ . Thus  $b \in (\mathbb{Z}/n\mathbb{Z})^\times$ , as desired. So  $S$  is strictly contained in  $(\mathbb{Z}/n\mathbb{Z})^\times$ , and by Lagrange's theorem, it has order at most  $\frac{1}{2}|(\mathbb{Z}/n\mathbb{Z})^\times|$ .  $\square$

**Corollary 5.** *If we have a pair  $e, d$  of corresponding RSA encryption and decryption keys, then we can factor  $n$ .*

*Proof.* Since  $ed \equiv 1 \pmod{\varphi(n)}$ ,  $ed - 1$  is a multiple of  $\phi(n)$ , whereby we can factor  $n$  using Theorem 3.  $\square$

A **Carmichael number** is a number  $n$  such that  $a^{n-1} \equiv 1 \pmod n$  for all integers  $a$  such that  $(a, n) = 1$ .

**Corollary 6.** *If we have a bound of  $\log^{O(1)} n$  on the size of the primes dividing  $\varphi(n)$ , or if  $n$  is a Carmichael number, we can factor  $n$  efficiently.*

*Proof.* The first statement can be shown by using a weak estimate on the density of primes to find an upper bound on the product of primes smaller than the bound  $\log^{O(1)} n$  in order to find a small multiple of  $\varphi(n)$ . The second follows from the theorem that for a prime  $p$  dividing a Carmichael number  $n$ ,  $p-1 \mid n-1$ .  $\square$

We can similarly define the Rabin-Miller randomized polynomial-time primality test: Given a positive integer  $n$ , we first check that it is not a perfect power. We then factor  $n-1 = 2^\ell m$ , where  $\ell$  is maximal, pick a random  $1 < a < n$ , and compute  $(a, n)$ . If  $(a, n) \neq 1$  then we're done, since  $n$  is then composite. Otherwise,  $(a, n) = 1$ . We then check that  $a^{2^\ell m} \equiv a^{n-1} \equiv 1 \pmod n$ , which can fail only if  $n$  is composite. We then consider

$$a^m, a^{2m}, a^{2^2 m}, \dots, a^{2^{\ell-1} m} \pmod n,$$

a sequence that eventually becomes 1. If  $a^{2^j m} \neq \pm 1$  while  $a^{2^{j+1} m} = 1$ , then  $x^2 - 1$  has more than two roots in  $\mathbb{Z}/n\mathbb{Z}$ , so this ring cannot be a field and  $n$  cannot be prime.

**Corollary 7.** *If this test outputs composite, it is correct. If  $n$  is prime, the test outputs prime with probability  $\geq \frac{1}{2}$ .*

*Proof.* The first sentence is clear from the algorithm. So suppose  $n$  is composite. Note that the  $a \in (\mathbb{Z}/n\mathbb{Z})^\times$  with  $a^{n-1} \equiv 1 \pmod n$  form a subgroup  $T$  of  $(\mathbb{Z}/n\mathbb{Z})^\times$ . We divide into cases, depending on whether or not  $T = (\mathbb{Z}/n\mathbb{Z})^\times$ .

If there exists any  $x \in (\mathbb{Z}/n\mathbb{Z})^\times \setminus T$ , then by Lagrange’s theorem, at least  $\frac{1}{2}$  of the choices of  $x$  will have this property, since the smallest possible index for a proper subgroup is 2. If our algorithm as described above picks any  $a \in (\mathbb{Z}/n\mathbb{Z})^\times \setminus T$ , which it does with probability  $\geq \frac{1}{2}$ , it correctly classifies  $n$ .

If, instead,  $T = (\mathbb{Z}/n\mathbb{Z})^\times$ ,  $n - 1$  has all of the necessary properties that  $k\varphi(n)$  had in the proof of Lemma 4 (as remarked in its statement). This lemma then shows that at least  $\frac{1}{2}$  of the choices of  $a$  will satisfy  $a^{2^j m} \not\equiv -1 \pmod n$  for the largest value of  $j$  such that  $a^{2^j m} \not\equiv 1 \pmod n$ , and will thus lead the algorithm to classify  $n$  as composite.  $\square$

We can exploit the fact above by essentially “guessing” the structure of  $(\mathbb{Z}/n\mathbb{Z})^\times$ , trying to pick a multiple of  $\varphi(n)$  by choosing numbers with many small prime divisors. The homomorphism  $\mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{Z}/p_i\mathbb{Z}$  given by sending  $a$  to its residue modulo  $p$  implies that raising a number  $a \in (\mathbb{Z}/n\mathbb{Z})^\times$  such that  $p \nmid a$  to a power that is a multiple of  $p_i - 1$  will yield a multiple of  $p_i$ .

This leads to a Pollard’s factorization algorithm: We pick an integer  $0 < a < n$ , pick an integer  $k$  that is a multiple of many small primes (say  $\text{lcm}(1, \dots, m)$ ), and compute  $(a^k - 1, n)$ . If  $(a^k - 1, n) = n$ , we use the algorithm above to factor  $n$  with  $k$  in place of  $k\varphi(n)$ , and if  $(a^k - 1, n) = 1$ , we pick a larger value of  $k$  (or a larger choice of  $m$ ). If  $p_i - 1 \mid k$  for some but not all  $i$ ,  $(a^k - 1, n)$  is likely to be a proper nontrivial factor of  $n$ , as  $a^k - 1$  is unlikely to be a multiple of  $p_j - 1$ , where  $p_j - 1 \nmid k$  (it is not difficult to see that this occurs with probability  $\Theta(\frac{1}{p_j - 1})$ ). If  $(a^k - 1, n) = 1$ , then increasing  $k$  is natural, because this could only occur if  $k$  were not a multiple of  $p_i - 1$  for any prime  $p_i$ .

Unfortunately, this algorithm sometimes will be very inefficient, particularly when the  $p_i - 1$  are not products of small primes. This observation illustrates the failure of  $(\mathbb{Z}/n\mathbb{Z})^\times$  to reveal the factorization of  $n$  easily, even though its order alone would be sufficient to factor  $n$ . Although in most cases, the order  $\varphi(n)$  is a product of mostly small primes, in the worst case, when  $\varphi(n)$  is a multiple of some very large primes, this approach gives us little hope. Thus Pollard’s algorithm is an efficient way to factor most  $n$ , but not all.

We thus see that the general problem of determining the properties of  $(\mathbb{Z}/n\mathbb{Z})^\times$  is one that may be easy for most randomly chosen inputs, but very hard on certain inputs. It would be nice, then, to have a group that contains all of the information about the factorization of  $n$ , yet whose order is “random.” If this were the case, there would be a good chance of being able to factor  $n$  by trying many times on randomly chosen orders, using the same methodology as Pollard’s algorithm.

This is where elliptic curves come in handy. Each elliptic curve  $E$  over the ring  $\mathbb{Z}/n\mathbb{Z}$  associates to  $n$  a group  $G(E, \mathbb{F}_{p_i})$  that, in a sense made precise by Theorem 8, contains the same information about  $n$  as the group  $(\mathbb{Z}/n\mathbb{Z})^\times$ . Moreover, we will see that the role of  $p_i - 1$  in this problem is replaced by a number in the interval  $p_i + 1 - 2\sqrt{p_i} < |G(E, \mathbb{F}_{p_i})| < p_i + 1 + 2\sqrt{p_i}$ . It is rather likely that some number in this range will have small prime factors, and as a consequence, the algorithm will much more efficiently be able to factor numbers.

## 2.3 Elliptic Curves

Elliptic curves may be viewed from the perspectives of many fields of mathematics, including number theory, analysis, and algebraic geometry. Although we’ll give a taste of this in Section 2.4, the focus of this section will be on the special case of elliptic curves over a finite field. Our goal is to define the **group law** associated to an elliptic curve over a field and remark upon the generalization to a ring such as  $\mathbb{Z}/n\mathbb{Z}$ .

### 2.3.1 Preliminaries

We first define affine and projective space over a field  $K$ . The **affine space**  $\mathbf{A}_K^n$  is defined to be the set  $K^n$ , with no vector space structure. The **projective space**  $\mathbf{P}_K^n$  is defined, again as a set, to be  $\{(\alpha_0, \dots, \alpha_n) \in K^{n+1} \setminus \{0\}\}$  modulo the equivalence relation  $(\alpha_0, \dots, \alpha_n) \sim (\lambda\alpha_0, \dots, \lambda\alpha_n)$  for  $\lambda \in K^\times$ . We will write the equivalence class of  $(\alpha_0, \dots, \alpha_n) \in \mathbf{P}_K^n$  as  $[\alpha_0, \dots, \alpha_n]$ . Intuitively, the projective space contains points “at infinity” corresponding to intersections of parallel hyperplanes in affine space. On the other hand, it is clear from the definition that there are no distinguished points in projective space, and so there is no preferred affine subset of projective space. A cover of projective space by subsets “isomorphic” to affine space can be found by considering the  $n+1$  subsets where  $\alpha_i = 0$  for each  $i$ . These are called  $\mathbf{A}_{K,i}^n$ , though the  $K$  is often omitted. The technicalities of defining the structure on these spaces, which are actually **varieties**, can be found in [Shaf] or [Ha]. None of these details will be important to the overview in this paper, though it is occasionally useful to keep in mind that the endomorphisms on elliptic curves we consider in 2.4 are cases of a more general notion.

An elliptic curve is defined over a field or ring. Let the field  $K$  have characteristic neither equal to 2 or 3, an assumption we will hold throughout this paper. Then an **elliptic curve**  $E$  is the set of points  $[x, y, z] \in \mathbf{P}_K^2$  defined by the homogeneous polynomial  $y^2z = x^3 + axz^2 + bz^3$  over the projective plane  $\mathbf{P}_K^2$  for  $a, b \in K$ , or the curve  $y^2 = x^3 + ax + b$  defined over the affine plane  $\mathbf{A}_K^2$  together with a point at infinity, corresponding to  $[0, 1, 0]$  in the projective form of the curve. One should also think of the defining polynomial itself as being part of the information present in  $E$ . Under the conditions on the characteristic, any other object one might call an “elliptic curve” can be transformed by a simple change of variables into the form given. Over a ring  $R$ , the definition is similar, though in this case, we will require  $2, 3 \in R^\times$ . The projective space  $\mathbf{P}_R^2$  will also need to be redefined, which we do in Section 2.3.3. Note that we say “elliptic curve over  $K$ ” (or  $R$ ) to mean an elliptic curve whose coefficients  $a, b$  are in  $K$  (or  $R$ ).

### 2.3.2 The Group Law

Suppose that we have points  $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$  on an elliptic curve  $y^2 = x^3 + ax + b$  with  $a, b \in \mathbb{Q}$  (or, more generally, any field  $K$ ). Then an equation for the line through  $P_1$  and  $P_2$  is obtained by taking  $m = \frac{y_1 - y_2}{x_1 - x_2}$  to be the slope of this line and using the formula  $y = m(x - x_1) + y_1$ . This line intersects the curve in a third point, which we shall solve for by substituting this expression for  $y$ :  $(m(x - x_1) + y_1)^2 = x^3 + ax + b$ , or

$$x^3 - m^2x^2 + (-2my_1 + 2m^2x_1 + a)x - m^2x_1^2 + 2mx_1y_1 - y_1^2 + b = 0.$$

Since the coefficient  $m^2$  is the sum of the roots of the polynomial, while  $x_1, x_2$  are already roots, we can find the abscissa of the third point as  $x_3 = m^2 - x_1 - x_2$ , and the ordinate from the equation of the line,  $y_3 = m(x_3 - x_1) + y_1$ . We label  $P_3 = (x_3, y_3)$ , and we define  $0 = P_1 + P_2 + P_3$  in the additive group of points on the elliptic curve, so that  $P_1 + P_2 = -P_3$ . We define the negation of a point to be its reflection over the  $x$ -axis, meaning that  $-(x, y) = (x, -y)$ . The third point on this line connecting  $(x, y)$  and  $(-x, y)$  is the point at infinity, and the identity of this additive group.

Commutativity and identity properties follow immediately from the above definition, but associativity is tedious to verify, as it breaks up into cases. We refer a reader interested in the proof to [ST], though we will provide a somewhat opaque proof of this result for subfields of  $\mathbb{C}$  when we motivate the Weil pairing in Section 2.4.

Associated to an elliptic curve  $E$  over a field  $K$  we have defined an abelian group structure  $G(E, K)$  on its points. Moreover, for a field extension  $L/K$ , there is a natural injection  $G(E, K) \hookrightarrow G(E, L)$ . Many amazing results in number theory describe various properties of  $G(E, \mathbb{Q})$ . Although we will not need them, they are worth mentioning. Mordell’s theorem shows that this group is always finitely generated. Mazur’s theorem shows that the torsion subgroup is either  $C_n$  for  $1 \leq n \leq 10$  or  $n = 12$ , or  $C_2 \times C_{2n}$  for  $1 \leq n \leq 4$ . Determining  $G(E, \mathbb{F}_p)$  is generally easier than computing the group law over  $\mathbb{Q}$ , and in fact, the most important information about  $G(E, \mathbb{F}_p)$  for our purposes will be its order (which is finite).

### 2.3.3 Elliptic Curves over $\mathbb{Z}/n\mathbb{Z}$

One can define elliptic curves over rings as well, though to do so requires a more complicated group law, since division is not permissible and other issues arise. The story is detailed in Lenstra's paper [Le], which also describes the factoring algorithm we present in Section 2.5.1. There are various constraints that make the process simpler. We'll avoid delving into the general theory and study the specific case of  $\mathbb{Z}/n\mathbb{Z}$ , where  $n$  is prime to 2 and 3. This will suffice for our purposes, because our factoring algorithm will specifically check for divisibility by small primes. Over a ring  $R$ , the projective space  $\mathbf{P}_R^n$  is redefined as

$$\{(\alpha_0, \dots, \alpha_n) \in R^{n+1} \mid (\alpha_0, \dots, \alpha_n) = (1)\} / (\alpha_0, \dots, \alpha_n) \sim (\lambda\alpha_0, \dots, \lambda\alpha_n), \lambda \in R^\times,$$

and it becomes generally more important to consider the elliptic curve within projective space. We still define the curve as the solutions to the homogeneous equation

$$y^2z = x^3 + axz^2 + bz^3$$

in  $\mathbf{P}_R^2$ , though we require that the discriminant  $-4a^3 - 27b^2$  be a unit (which just means nonzero over a field, so this is consistent).

The ring  $\mathbb{Z}/n\mathbb{Z}$  holds information about its factors within its subgroups, and has the property that if  $n = n_1n_2$ , where  $(n_1, n_2) = (1)$ , then  $\mathbb{Z}/n\mathbb{Z} = (\mathbb{Z}/n_1\mathbb{Z}) \times (\mathbb{Z}/n_2\mathbb{Z})$ . We can show that this translates to the elliptic curve as well. Note that for an elliptic curve  $E$  over  $R$  and an injection  $\varphi : R \hookrightarrow S$ ,  $G(E, R) \subseteq G(E, S)$ . But we can replace the injection  $\varphi$  with any homomorphism, and apply this map both to the coefficients of  $E$  and to  $R$  to obtain a homomorphism  $\tilde{\varphi} : G(E, R) \rightarrow G(E, S)$ . In particular, for an elliptic curve  $E$  over  $\mathbb{Z}/n\mathbb{Z}$  and  $\varphi_i$  reduction modulo  $n_i$ , we obtain maps  $\tilde{\varphi}_i : G(E, \mathbb{Z}/n\mathbb{Z}) \rightarrow G(E, \mathbb{Z}/n_i\mathbb{Z})$ . We can now prove:

**Theorem 8** ([Wa, pp. 65–66]). *The map*

$$\tilde{\varphi}_1 \times \tilde{\varphi}_2 : G(E, \mathbb{Z}/n\mathbb{Z}) \rightarrow G(E, \mathbb{Z}/n_1\mathbb{Z}) \times G(E, \mathbb{Z}/n_2\mathbb{Z})$$

*is an isomorphism.*

*Proof.* Note first of all that the discriminant of  $E$  when reduced modulo  $n_1$  or  $n_2$  is still a unit, since it is relatively prime to  $n$ . Thus the groups on the right above are well-defined. Reduction modulo  $n_i$  yields, via the Chinese Remainder Theorem,

$$\varphi_1 \times \varphi_2 : \mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z},$$

and thus a bijection between triples of these elements. This map induces maps on  $\mathbf{P}_{\mathbb{Z}/n\mathbb{Z}}^2$ , which we'll denote  $\bar{\varphi}_1, \bar{\varphi}_2$ , since if  $(x, y, z) = (ux', uy', uz')$  are equivalent triples, their images are equivalent via the image of the unit  $u$ . When applied to the bijection on triples, this yields a bijection

$$\bar{\varphi}_1 \times \bar{\varphi}_2 : \mathbf{P}_{\mathbb{Z}/n\mathbb{Z}}^2 \rightarrow \mathbf{P}_{\mathbb{Z}/n_1\mathbb{Z}}^2 \times \mathbf{P}_{\mathbb{Z}/n_2\mathbb{Z}}^2.$$

Finally,  $y^2z = x^3 + axz^2 + bz^3 \pmod{n}$  implies  $y^2z = x^3 + axz^2 + bz^3 \pmod{n_i}$  for  $i = 1, 2$ . The converse is true, again by the Chinese Remainder Theorem. Thus the map  $\bar{\varphi}_1 \times \bar{\varphi}_2$  is a bijection. That it is a homomorphism can be verified by a tedious but simple computation that we will omit.  $\square$

Our last comment is that although we have not specified the exact group law over a ring  $\mathbb{Z}/n\mathbb{Z}$ , one can use the usual group law over a field  $K$  when dealing only with units. This fails to yield the right answer exactly when a nonzero nonunit appears in one of the coordinates of one of the points. For the purpose of factoring, however, we actually need not explain how to carry out the group operation in this situation, as a nonzero nonunit appears exactly when we have successfully managed to factor  $n$ , so the algorithm can terminate at this stage.

## 2.4 The Weil Pairing

The Weil pairing is a construction on the  $n$ -torsion points of an elliptic curve that is important for proving theoretical results, such as the analogue of the Riemann Hypothesis for elliptic curves, as well as for computational applications, discussed in Section 2.5.2.2 and Section 2.5.4. The progression used in the subsections following the motivation come largely from Chapters 2 to 4 and 11 to 12 of [Wa], though we provide few proofs. Our goal is to sketch the ideas behind the results on elliptic curves over finite fields most relevant to cryptography. Readers without background in complex analysis can safely skip Section 2.4.1.

The texts [Si1], [Si2] provide an abstract viewpoint on pairings that is more useful from the purposes of number theory, while [CFA] provides a comprehensive treatment of the cryptographic applications of the Weil pairing and the related Tate-Lichtenbaum pairing.

### 2.4.1 Motivation

For those familiar with either the theory of algebraic curves over  $\mathbb{C}$  or of compact Riemann surfaces, the following will serve as motivation for some of the theorems that follow (and provide proofs of special cases of results we will not prove).

If we view an elliptic curve in  $\mathbf{P}_{\mathbb{C}}^2$  as a compact Riemann surface  $X$  of genus 1, there exists a lattice  $\Lambda \subseteq \mathbb{C}$  such that  $X$  maps biholomorphically to the quotient  $E = \mathbb{C}/\Lambda$ . Indeed, for the inverse, the Weierstrass  $\wp$ -function defines the map  $z \mapsto (\wp(z), \wp'(z))$  sending  $\mathbb{C}/\Lambda$  to the solutions of  $y^2 = 4x^3 - 60G_4(\Lambda)x - 140G_6(\Lambda)$ , where  $G_4, G_6$  are the Eisenstein invariants of the lattice  $\Lambda$ .

The importance of this perspective comes from the obvious addition law on points in  $\mathbb{C}/\Lambda$ , which is carried via the isomorphism to yield an abelian group structure on  $E$ . This formula is given by

$$\wp(z_1 + z_2) = \frac{1}{4} \left( \frac{\wp'(z_2) - \wp'(z_1)}{\wp(z_2) - \wp(z_1)} \right)^2 - \wp(z_1) - \wp(z_2).$$

One can derive the associativity of the group law over subfields of  $\mathbb{C}$  from this isomorphism. We also can easily determine the  $n$ -torsion points of  $\mathbb{C}/\Lambda$ : if  $\Lambda = \mathbb{Z}\omega_1 \oplus \mathbb{Z}\omega_2$ , then these are precisely  $\frac{\ell_1\omega_1}{n} + \frac{\ell_2\omega_2}{n}$ , for  $0 \leq \ell_1, \ell_2 < n$ . Thus the  $n$ -torsion group, which we denote by  $G_n(E, \mathbb{C})$ , is isomorphic to  $\mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ , for any  $n$ . More generally, we let  $G_n(E, K)$  denote the subgroup of  $G(E, \bar{K})$  (where  $\bar{K}$  denotes the algebraic closure) of points  $P$  such that  $nP = 0$ . With some additional technical steps, one can use the case of  $\mathbb{C}$  to conclude that  $G_n(E, K) \cong \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$  for general fields of characteristic 0.

We denote by  $C_n = \langle \zeta \rangle$  the group of  $n^{\text{th}}$  roots of unity in  $\mathbb{C}$ , where  $\zeta$  is a primitive root. In [Ga], a Weil pairing  $w_n : G_n(E, \mathbb{C}) \times G_n(E, \mathbb{C}) \rightarrow C_n$  is defined by the formula:

$$w_n \left( \frac{\ell_1\omega_1}{n} + \frac{\ell_2\omega_2}{n}, \frac{m_1\omega_1}{n} + \frac{m_2\omega_2}{n} \right) = \exp \left( \frac{2\pi i(\ell_1 m_2 - m_1 \ell_2)}{n} \right)$$

The key properties of this pairing follow directly from this definition. It is linear in each component, and satisfies  $w_n(0, g) = w_n(g, 0) = w_n(g, g) = 1$  for all  $g$ . On the other hand, for  $g \neq 0$ , there exist  $h, h'$  such that  $w_n(g, h) \neq 0$ ,  $w_n(h, g) \neq 0$ —this means that  $w_n$  is **nondegenerate**. Finally  $w_n(g, h) = w_n(h, g)^{-1}$ . In the terminology of linear algebra, one will easily recognize that these properties are analogous to those of a nondegenerate skew-symmetric bilinear form.

### 2.4.2 Endomorphisms

We next study certain maps on elliptic curves. These maps are a general case of **regular maps**. Let us consider a curve in affine space,  $\mathbf{A}_K^2$ . Then the coordinates  $x, y$  themselves define functions assigning to each point of the curve a value in  $K$ . One can generate many functions on the curve by considering polynomials in  $x, y$ . Such maps are examples of **regular functions**. One can also define a map  $\tau : E \rightarrow \mathbf{A}_K^2$  by defining  $\tau(x, y) = (p_1(x, y), p_2(x, y))$  for a pair of regular functions  $p_1, p_2$ . If the image of an elliptic curve  $E$  under a map of this form happens to lie on  $E$ , and the map acts as a group homomorphism on  $G(E, K)$ , then the map is called a **endomorphism**.

We allow, in fact,  $p_1$  and  $p_2$  to be rational functions. Under a map of this form, it is possible for points of the curve to be sent to the point at infinity.

We can determine a convenient form in which all such maps can be written. Since all points on the curve  $E$  satisfy  $y^2 = x^3 + ax + b$ , we can assume that the numerator and denominator of  $p_1, p_2$  have no powers of  $y$  larger than 1. In fact, multiplying the denominator by its conjugate with respect to  $y$  (meaning we replace  $y$  with  $-y$ ), we find that  $p_i(x, y) = \frac{q_i(x) + s_i(x)y}{t_i(x)}$  for functions  $q_i, s_i, t_i \in K(x, y)$  and  $i \in \{1, 2\}$ . Since  $\tau(x, -y) = \tau(-x, y) = -\tau(x, y)$ , in fact, we have  $s_1(x) = 0$  and  $q_2(x) = 0$ , so  $\tau(x, y) = (r_1(x), r_2(x)y)$ , where  $r_1, r_2 \in K(x)$  are rational functions of  $x$ .

If the map  $\tau$  is nonzero, we define the **degree** of this map to be the larger among the degree of the numerator and denominator of  $r_1$ , and if  $\tau$  is zero, we define the degree to be 0. The sum of endomorphisms defines an endomorphism by summing the images using the group law, and endomorphisms admit multiplication by an integer in the same way. We denote the identity endomorphism  $\tau(x, y) = (x, y)$  by 1 and, over  $\mathbb{F}_q$ , the **Frobenius endomorphism** by  $\varphi_q(x, y) = (x^q, y^q)$ , easily checked to be an endomorphism.

Following [Wa], we will use the Weil pairing in the next section to derive the following result:

**Theorem 9.** *Let  $\sigma, \tau$  be endomorphisms and  $s, t \in \mathbb{Z}$ . Then*

$$\deg(s\sigma + t\tau) = s^2 \deg \sigma + t^2 \deg \tau + st(\deg(\sigma + \tau) - \deg \sigma - \deg \tau).$$

Another way to create new endomorphisms is via **composition**, for example  $\varphi_q^n = \varphi_q \circ \dots \circ \varphi_q$ ,  $n$  times. The endomorphism  $\tau$  is defined to be **separable** if  $r_1'(x)$  is not identically 0. This rather technical definition is explained by Lemma 10, which expresses the notion of separability in terms of the degree and kernel of the map, two seemingly more intrinsic qualities. This lemma can be proven by carefully keeping track of the roots of the numerator and denominator of  $r_1, r_2$ .

**Lemma 10.** ([Wa], p. 49-50) *For  $\tau$  a nonzero endomorphism on an elliptic curve over an algebraically closed field,  $\deg \tau \geq |\ker \tau|$ , with equality if and only if  $\tau$  is separable.*

Thus, if we wanted to compute the number of points on the curve  $E$  over  $\mathbb{F}_{p^k}$ , we might consider  $\varphi_p^k$ , which fixes  $\mathbb{F}_{p^k}$  and no other elements of  $\overline{\mathbb{F}_p}$ . Then the endomorphism  $\varphi_{p^k} - 1$  has kernel exactly corresponding to the points of  $E$  over  $\mathbb{F}_{p^k}$ . However, we need to know whether this endomorphism is separable or not. This question is answered by another lemma, a convenient condition for separability of certain endomorphisms over  $\mathbb{F}_{p^k}$ .

**Lemma 11.** ([Wa], p. 54) *Over  $\mathbb{F}_{p^k}$ , where  $r, s \in \mathbb{Z}$ ,  $r\varphi_{p^k} + s$  is separable exactly when  $p \nmid s$ .*

The main idea of the proof is to first reduce to the question of whether  $s$  is separable, and to then answer this by first showing that if  $n = (r_1(x), r_2(x)y)$ ,  $\frac{r_1'(x)}{r_2(x)} = n$ .

These results will be used later to conclude the Hasse bound. The final step simply involves computing the degree of  $\varphi_{p^k} - 1$  using Theorem 9.

Another result that we will invoke later is:

**Theorem 12** ([Wa, p. 95–96]). *If  $|G(E, \mathbb{F}_{p^k})| = p^k + 1 - \ell$ , then  $\varphi_{p^k}^2 - \ell\varphi_{p^k} + p^k = 0$  as endomorphisms. Moreover, for  $m \neq \ell$ ,  $\varphi_{p^k}^2 - m\varphi_{p^k} + p^k \neq 0$ .*

For the first statement, the key idea is to show that  $\varphi_{p^k}^2 - \ell\varphi_{p^k} + p^k$  is identically zero on  $G_n(E, \mathbb{F}_p)$  for infinitely many choices of  $n$ . Then Lemma 10 implies that this endomorphism is identically zero. The second is easy: If  $\varphi_{p^k}^2 - m\varphi_{p^k} + p^k = 0$ , then by subtracting  $\varphi_{p^k}^2 - \ell\varphi_{p^k} + p^k = 0$  we find  $(\ell - m)\varphi_{p^k} = 0$  as endomorphisms, which can happen only if  $\ell = m$ .

### 2.4.3 The Weil Pairing and its Consequences

We shall state two key theorems, whose proofs are in [Wa].

**Theorem 13** ([Wa, p. 75]). *If  $\text{char } K \nmid n$  or  $n = 0$ , then  $G_n(E, K) \cong \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ .*

**Theorem 14** ([Wa, pp. 83, 334-335]). *Let  $C_n \subset \overline{K}$  be the group of  $n^{\text{th}}$  roots of unity and  $E$  be the elliptic curve  $y^2 = x^3 + ax + b$ . Then there exists a map  $w_n : G_n(E, K) \times G_n(E, K) \rightarrow C_n$  called the **Weil pairing**, with the properties that*

- The map  $w_n$  is bilinear and nondegenerate in each variable (as defined in Section 2.4.1).
- We have  $w_n(g, g) = 1$  and  $w_n(g, h) = w_n(h, g)^{-1}$  for  $g, h \in G_n(E, K)$ .
- If  $\varphi : \overline{K} \rightarrow \overline{K}$  is an automorphism with  $a, b$  as fixed points,  $w_n(\varphi(g), \varphi(h)) = \varphi(w_n(g, h))$ .
- If  $\sigma$  is an endomorphism of  $E$ ,  $w_n(\sigma(g), \sigma(h)) = w_n(g, h)^{\deg \sigma}$ .

If  $\sigma$  is an endomorphism of  $E$ , it is easy to see that it restricts to an endomorphism of  $G_n(E, K)$ . Using Theorem 13 we can choose a basis  $\omega_1, \omega_2$  for  $G_n(E, K)$ , so that  $\sigma$  is defined by  $(\omega_1, \omega_2) \mapsto (\ell_1\omega_1 + \ell_2\omega_2, m_1\omega_1 + m_2\omega_2)$ . Then one can show via simple computations (see [Wa]) that  $w_n(\omega_1, \omega_2)$  is a primitive  $n^{\text{th}}$  root of unity and that  $\det \begin{pmatrix} \ell_1 & \ell_2 \\ m_1 & m_2 \end{pmatrix} = \deg \alpha$ . As a consequence, one can prove that Theorem 9 holds over fields of characteristic  $p$  by computing the determinants of both sides as endomorphisms over  $G_n(E, K)$  for each  $n$  with  $p \nmid n$ . One can extend this result to all  $n$  by applying Lemma 10.

#### 2.4.4 The Hasse Bound

Over finite fields, it is possible to very accurately determine the order of the group of points on an elliptic curve. We will use some of the facts about endomorphisms we stated without proof in Section 2.4.2.

Let  $E$  have coefficients in a finite field  $\mathbb{F}_{p^k}$ . By the remarks above, which used Lemma 10 and 11,  $\deg(\varphi_{p^k} - 1) = |G(E, \mathbb{F}_{p^k})|$ .

**Theorem 15** ([Wa, pp. 91-94]). *For an elliptic curve  $E$  over  $\mathbb{F}_{p^k}$ ,  $2\sqrt{p^k} \geq p^k + 1 - |G(E, \mathbb{F}_{p^k})| \geq -2\sqrt{p^k}$ .*

*Proof.* Define  $q = p^k$  for simplicity. By the preceding observation,  $q + 1 - |G(E, \mathbb{F}_q)| = q + 1 - \deg(\varphi_q - 1)$ , a quantity we will call  $\ell$ . By Theorem 9, for  $s, t \in \mathbb{Z}$  with  $p \nmid t$ ,

$$\begin{aligned} \deg(s\varphi_q - t) &= s^2 \deg(\varphi_q) + t^2 \deg(-1) + st(\deg(\varphi_q - 1) - \deg(\varphi_q) - \deg(-1)) \\ &= qs^2 + t^2 - st\ell \end{aligned}$$

Since  $\deg(s\varphi_q - t) = qs^2 + t^2 - st\ell \geq 0$ ,  $q \left(\frac{s}{t}\right)^2 - \ell \left(\frac{s}{t}\right) + 1 \geq 0$ . Since  $\{\frac{s}{t} : p \nmid t\}$  is dense in  $\mathbb{R}$ , this inequality holds for all real  $r \in \mathbb{R}$  in place of  $\frac{s}{t}$ , so  $qr^2 - \ell r + 1 \geq 0$ . This implies that the discriminant is nonpositive, so  $\ell^2 - 4q \leq 0$ , or  $|\ell| \leq 2\sqrt{q}$ , as desired.  $\square$

This powerful result has many important cryptographic consequences. Several algorithms use the narrow range of possible orders for  $G(E, \mathbb{F}_q)$  given by the Hasse bound in an essential way to compute the order of the group. We also can use the range for the order to bound the running time of algorithms, such as those for computing factorizations and discrete logarithms.

#### 2.4.5 The Riemann Hypothesis for Elliptic Curves

Note that the following results are entirely optional, as they are not used in the remainder of the paper. The purpose of this section is to prove that the Hasse bound implies an analogue of the Riemann hypothesis for curves. This connection is suggestive of a far more general analogy between points on varieties and primes in number fields. The usual Riemann hypothesis is equivalent to a statement about the density of primes. In the same sense, the Riemann hypothesis on an elliptic curve is equivalent to a statement about the number of points on the curve.

The usual Riemann zeta function  $\zeta(s)$ , given by  $\sum_{n=1}^{\infty} n^{-s}$  for  $\text{Re}(s) > 1$  (and which can be extended to  $\mathbb{C} \setminus 1$ ), satisfies the identity

$$\pi^{-\frac{s}{2}} \Gamma\left(\frac{s}{2}\right) \zeta(s) = \pi^{-\frac{1-s}{2}} \Gamma\left(\frac{1-s}{2}\right) \zeta(1-s),$$

written with phrening symmetry. A famous conjecture asserts that for  $s \in \mathbb{C} \setminus \{-2, -4, \dots\}$  such that  $\zeta(s) = 0$ ,  $\operatorname{Re}(s) = \frac{1}{2}$ . For an elliptic curve  $E$  over  $\mathbb{F}_{p^k}$ , where we will write  $q = p^k$  and  $\ell = q + 1 - |G(E, \mathbb{F}_q)|$ , we define

$$\zeta_E(s) = \exp\left(\sum_{n=1}^{\infty} \frac{|G(E, \mathbb{F}_{q^n})|}{n} q^{-ns}\right).$$

**Theorem 16** ([Wa, pp. 97, 355]). *We have:*

$$\zeta_E(s) = \frac{q^{1-2s} - \ell q^{-s} + 1}{(1 - q^{-s})(1 - q^{1-s})}$$

which has the evident symmetry  $\zeta_E(s) = \zeta_E(1 - s)$ .

*Proof.* First consider the polynomial  $x^2 - \ell x + q = (x - \rho_1)(x - \rho_2)$ . This divides  $(x^n - \rho_1^n)(x^n - \rho_2^n) = x^{2n} - (\rho_1^n + \rho_2^n)x^n + q^n$  (since  $\rho_1\rho_2 = q$ ), with quotient we will call  $f(x)$ .

We have  $\varphi_q^2 - \ell\varphi_q + q = 0$  by Theorem 12, so  $0 = f(\varphi_q)(\varphi_q^2 - \ell\varphi_q + q) = \varphi_q^{2n} - (\rho_1^n + \rho_2^n)\varphi_q^n + q^n = 0$ , so by the uniqueness of  $\ell$  from Theorem 12,  $\rho_1^n + \rho_2^n = q^n + 1 - |G(E, \mathbb{F}_{q^n})|$ . (It is easy to see inductively that  $\rho_1^n + \rho_2^n$  is an integer for all  $n$ .) In particular,

$$\zeta_E(s) = \exp\left(\sum_{n=1}^{\infty} \frac{|G(E, \mathbb{F}_{q^n})|}{n} q^{-ns}\right) = \exp\left(\sum_{n=1}^{\infty} \frac{q^n + 1 - \rho_1^n - \rho_2^n}{n} q^{-ns}\right)$$

Using the Taylor expansion  $\log(1 - q^{-s}) = \sum -\frac{q^{-ns}}{s}$ , we obtain

$$\begin{aligned} \zeta_E(s) &= \exp\left(-\log(1 - q^{1-s}) - \log(1 - q^{-s}) + \log(1 - \rho_1 T) + \log(1 - \rho_2 T)\right) \\ &= \frac{q^{1-2s} - \ell q^{-s} + 1}{(1 - q^{-s})(1 - q^{1-s})}. \end{aligned} \quad \square$$

Moreover, we can verify the Riemann hypothesis in this case.

**Theorem 17** ([Wa, p. 357]). *If  $\zeta_E(s) = 0$ ,  $\operatorname{Re}(s) = \frac{1}{2}$ .*

*Proof.* Indeed, for the numerator  $q^{-2s}(q^{2s} - \ell q^s + q)$  to vanish, we find by the quadratic formula (treating the numerator as a quadratic in  $q^s$ ) that  $q^s = \frac{\ell \pm \sqrt{\ell^2 - 4q}}{2}$ . By the Hasse bound,  $\ell^2 \leq 4q$ , so the two solutions for  $q^s$  satisfy  $|q^s| = \sqrt{\frac{\ell^2}{4} + \frac{4q - \ell^2}{4}} = \sqrt{q}$ . This implies  $|q^s| = q^{\operatorname{Re}(s)} = \sqrt{q}$ , or  $\operatorname{Re}(s) = \frac{1}{2}$ .  $\square$

## 2.5 Elliptic Curve Cryptography

In this section we present the factoring algorithm using elliptic curves. After this we study the discrete logarithm problem on elliptic curves and study its security. Assuming the hardness of this problem, we provide elliptic curve variants of several of the cryptosystems mentioned in Section 2.2.3. Finally, we detail the practical implications of using elliptic curves in place of computations over  $\mathbb{F}_p^\times$ .

### 2.5.1 Factoring via $G(E, \mathbb{Z}/n\mathbb{Z})$

It is now very easy to describe the elliptic curve factoring algorithm, since it is a minor modification of Pollard's algorithm, and uses in a simple way the development of the preceding two sections. The key difference is that instead of working over  $\mathbb{Z}/n\mathbb{Z}$ , we work over  $G(E, \mathbb{Z}/n\mathbb{Z})$  for a suitably chosen elliptic curve  $E$ .

We first randomly choose a curve  $y^2 = x^3 + ax + b$  with a point  $P$  on it. As Lenstra observes in [Le], this can be done efficiently by choosing  $a \in \mathbb{Z}/n\mathbb{Z}$  randomly, choosing a random pair

$P = (u, v) \in (\mathbb{Z}/n\mathbb{Z})^2$ , and setting  $b = v^2 - u^2 - au \pmod n$ , so that  $P$  lies on  $E$ . Then for some large integer  $k$  with many small prime factors, we compute  $kP$  by repeated squaring, using the same formula as if  $\mathbb{Z}/n\mathbb{Z}$  were a field. (We could, as in Pollard's algorithm, take  $k = \text{lcm}(1, \dots, N)$  for some large integer  $N$ .) Note that inverses modulo  $n$  are efficiently computable via Euclid's algorithm, so this process is computationally efficient. If at any point during this computation, we fail because an inverse cannot be computed, we have found a nonzero nonunit of  $\mathbb{Z}/n\mathbb{Z}$ , thus factoring  $n$ .

Write  $\mathbb{Z}/n\mathbb{Z} = \mathbb{Z}/p_1^{\alpha_1}\mathbb{Z} \times \dots \times \mathbb{Z}/p_k^{\alpha_k}\mathbb{Z}$ , and consider the induced decomposition of the elliptic curve given in Theorem 8. Then the failure to compute an inverse by the usual method occurs whenever  $P \in G(E, \mathbb{Z}/p_1^{\alpha_1}\mathbb{Z}) \times \dots \times G(E, \mathbb{Z}/p_k^{\alpha_k}\mathbb{Z})$  has a coordinate that is the point at infinity in  $G(E, \mathbb{Z}/p_i^{\alpha_i}\mathbb{Z})$  for some  $i$ . This is essentially the same behavior as in Pollard's algorithm, except that the component elliptic curves have varying orders. The requirement that  $k$  have many shared factors with  $\varphi(n)$  now changes to having shared factors with the orders of these component curves. By picking many random curves, it is likely that one of the curves will have small factors in  $\varphi(n)$ .

## 2.5.2 Elliptic Curve Discrete Logarithm Problem

The elliptic curve discrete logarithm problem is as follows: Given  $G(E, K)$ , a point  $P \in G(E, K)$ , and  $Q = nP$ , compute  $n$ . There are a tremendous number of algorithms that aim to solve this problem, many for very specific classes of elliptic curves. A short introduction to this can be found in [Wa] and a thorough treatment can be found in [CFA]. Here we cover general methods developed by Pollard, as well as a solution using the results of Section 2.4 specific to a certain class of curves.

### 2.5.2.1 Pollard $\rho$ and $\lambda$ Algorithms

The **Pollard  $\rho$  algorithm** is very simple and general; it can be used to compute the discrete logarithm in  $\mathbb{F}_p^\times$  as well. We will write the algorithm here using the notation of arithmetic on an elliptic curve, with uppercase letters representing points on the curve and lowercase letters representing integers. Pick a function  $f : G(E, K) \rightarrow G(E, K)$  of *sets* that, as [Wa] describes, "behaves rather randomly." While  $f$  need not be an endomorphism, we will require it to be reasonably explicit, in a sense made precise below. The intuition is that if  $f$  were chosen so that its value at every point was another truly random value of the curve, one would expect iterating  $P, f(P), f(f(P))$ , etc. to repeat a value in roughly  $\sqrt{\frac{\pi|G(E, K)|}{2}}$  steps.

Pick random  $i_0, j_0$  and define  $S_0 = p_0P + q_0Q$ , where  $nP = Q$  is the discrete logarithm instance we are trying to solve. Also define  $S_i = f(S_{i-1})$ , but keep track of values  $p_i, q_i$  such that  $S_i = p_iP + q_iQ$ . This imposes a requirement that  $f$  be sufficiently explicit for the image of a point of the form  $S_{i-1} = p_{i-1}P + q_{i-1}Q$  to have an efficiently computable representation in the form  $p_iP + q_iQ$ . In other words, we need to be able to write the difference  $f(S_i) - S_i$  in the form  $pP + qQ$ . Now suppose that for some integers  $\alpha \neq \beta$ ,  $S_\alpha = S_\beta$ . This will eventually be the case, since  $G(E, K)$  is finite. Then  $p_\alpha P + q_\alpha Q = p_\beta P + q_\beta Q$ , or  $P(p_\alpha - p_\beta) = Q(q_\beta - q_\alpha)$ . From here, there are only  $k = \gcd(q_\beta - q_\alpha, |G(E, K)|)$  different possible choices for the logarithm, so we can test them all. This is because modulo  $m = \frac{|G(E, K)|}{k}$ , the value of  $n$  is simply  $(q_\beta - q_\alpha)^{-1}(p_\alpha - p_\beta)$ , and there are only  $k$  possible choices modulo  $|G(E, K)|$  that leave this residue modulo  $m$ .

The **Pollard  $\lambda$  algorithm** is a variant on this idea. Instead of having a single starting point  $S_0$ , this process is simultaneously carried out on an array of values  $S_{0,0}, S_{0,1}, \dots, S_{0,k}$ . If there is a match between two different paths, we can get a relationship similar to that in the  $\rho$  algorithm, and again find few possibilities for the logarithm.

Interestingly, as [Wa] explains, the  $\rho$  and  $\lambda$  are chosen to match the nature of these algorithms. In the  $\rho$  algorithm, one searches for a path that loops back to itself, which might look like the Greek letter  $\rho$ . In the  $\lambda$  algorithm, two paths need to converge together, which looks like a  $\lambda$ .

### 2.5.2.2 Reduction via the Weil Pairing

Starting with a result of Menezes, Okamoto, and Vanstone in [MOV], cryptographers have managed to use Weil and other pairings on elliptic curves to reduce instances of the elliptic curve

discrete logarithm problem to instances of the discrete logarithm problem over finite fields, which are usually much easier to solve.

It is possible to efficiently compute the order of a point as a consequence of Theorem 15, which can be used to restrict the range of possible orders of the point to just finitely many, each of which can then be checked. There are faster algorithms, a subject covered in Section 19.4 of [CFA].

Suppose that we need to compute  $s = \log_P Q$ , where  $P$  has order  $n$ . We have  $G_n(E, \mathbb{F}_p) \subseteq G(E, \mathbb{F}_{p^k})$  for some choice of  $k$ . One chooses  $R \in G(E, \mathbb{F}_{p^k})$ , and computes its order  $m$ . Next one computes  $R' = \frac{m}{(m,n)}R \in G_n(E, \mathbb{F}_p)$  (since  $(m, n) \mid n$ ), and computes  $\ell = \log_{w_n(P, R')} w_n(Q, R')$ , a discrete logarithm problem in  $\mathbb{F}_{p^k}$ . Then  $w_n(P, R')^\ell = w_n(\ell P, R') = w_n(Q, R')$ , implying that in the subgroup  $w_n(G_n(E, \mathbb{F}_p), R') = C_{(m,n)} \subseteq C_n$ , we have  $\ell P = Q$ , which in turn implies that  $s \equiv \ell \pmod{(m, n)}$ . For sufficiently many choices of  $R$ , the values  $\ell \pmod{(m, n)}$  should allow one to reconstruct  $s \pmod n$ .

Unfortunately, the discrete logarithm problem over  $\mathbb{F}_{p^k}$  is difficult if  $k$  is large. We define a curve  $E$  over  $\mathbb{F}_p$  to be **supersingular** if  $|G(E, \mathbb{F}_p)| = p + 1 - a$  where  $a \equiv 0 \pmod p$ . One can show that  $k$  is rather small in the case of such a curve. Following [Wa], we will do this in the special case of  $a = 0$ .

**Theorem 18** ([Wa, p. 146]). *If  $|G(E, \mathbb{F}_p)| = p + 1$ , and there exists  $P \in G(E, \mathbb{F}_p)$  of order  $n$ , then we can take  $k = 2$  above.*

*Proof.* Since  $P$  has order  $n$ ,  $n \mid p + 1$  by Lagrange's theorem, and thus  $1 \equiv -p \pmod n$ . Let  $Q \in G_n(E, \mathbb{F}_p)$ . We have  $\varphi_p^2 = -p$  by 12, so  $\varphi_p^2(Q) = -pQ = 1 \cdot Q = Q$  since  $nQ = 0$ . Since  $\varphi_p^2$  fixes exactly  $G(E, \mathbb{F}_{p^2})$ ,  $Q \in G(E, \mathbb{F}_{p^2})$ .  $\square$

## 2.5.3 Cryptography

Many of the cryptographic schemes presented in Section 2.2.3 depended upon the hardness of the discrete logarithm problem in  $\mathbb{F}_p^\times$ . Resting on the assumption that the elliptic curve discrete logarithm problem is hard for a class of instances of  $G(E, R)$  that can be efficiently generated, we can develop secure cryptographic protocols. The following are given in [Wa], though the details of their security are discussed more fully in [CFA]. As in the last subsection, we use lowercase letters to indicate integers and uppercase letters to denote points on the elliptic curve. In some cases, such as the Elgamal Public Key Encryption scheme, the message should be encoded as a point on the curve. In others, such as the Digital Signature Algorithm, it is encoded directly as an integer.

### 2.5.3.1 Diffie-Hellman Key Exchange

It is an open problem to prove the difficulty of determining the point  $xyP$  from  $P, xP$ , and  $yP$ , even under the assumption that the discrete logarithm problem is hard. This problem is similar to that of the Diffie-Hellman problem described in Section 2.2.2, and is called the **elliptic curve Diffie-Hellman problem**.

For the protocol, Alice and Bob agree on a choice of  $E$  and  $p$  so that the elliptic curve Diffie-Hellman problem is hard for  $G(E, \mathbb{F}_p)$ , and agree on a point  $P \in G(E, \mathbb{F}_p)$ . Alice secretly chooses  $x$  while Bob secretly chooses  $y$ , both integers. Alice sends  $xP$  to Bob, who sends  $yP$  back. They both compute  $xyP$ , and extract a key from it.

### 2.5.3.2 Elgamal Public Key Encryption

Suppose that Bob wants to receive a message. He publishes an elliptic curve  $E$  over  $\mathbb{F}_p$ ,  $p$ , a point  $P$ , and  $xP$ , where  $x$  is his secret key. Alice encrypts her message  $M \in G(E, \mathbb{F}_p)$  by picking a secret integer  $y$  and computing and sending  $C_1 = yP$ ,  $C_2 = M + y(xP)$  to Bob, who decrypts by computing  $C_2 - xC_1 = M$ . It is unclear how an eavesdropper would be able to compute  $M$  without solving the discrete logarithm problem, though in the manner of Theorem 1 one can relate the hardness of breaking this cryptosystem to the elliptic curve Diffie-Hellman problem. Again, the modification to produce an elliptic curve algorithm from that given above was simply a matter of writing the variables additively and using the group of points on a curve  $E$ .

### 2.5.3.3 Digital Signature Algorithm

Recall that Alice has a document  $m \in \mathbb{Z}$  that she wishes to sign. Alice picks  $E$  over  $\mathbb{F}_q$ , where  $|G(E, \mathbb{F}_q)| = ep$  for  $p$  a prime and  $e$  very small, say 1, 2, or 4. She also picks a point  $P \in G(E, \mathbb{F}_q)$  of order  $p$ , just as in the classical variant. Finally she picks a secret integer  $s$  and publishes  $(E, \mathbb{F}_q, p, e, P, sP)$ . Alice picks a random integer  $1 \leq \ell < p$ , computes  $R = \ell P = (x, y)$  and  $a = k^{-1}(m + sx) \bmod p$ , and produces a signed document  $(m, R, a)$ . The verifier Bob computes  $c_1 = a^{-1}m \bmod p$ ,  $c_2 = a^{-1}x \bmod p$ , and accepts if  $R = c_1P + c_2(sP)$ . A correctly signed document is always accepted, since  $R = kP = a^{-1}(m + sx)P = c_1P + c_2(sP)$ . As before, if the discrete logarithm problem is hard, it seems difficult to use the public information, even with many signed documents, to forge a signature, except perhaps in some special cases. It is unknown, however, whether the hardness of breaking this algorithm follows from the hardness of the discrete logarithm.

### 2.5.4 Cryptosystems Using the Weil Pairing

In [BF], Boneh proposed an identity-based encryption scheme using a type of nondegenerate bilinear pairing that can be constructed, for example, from the Weil pairing. In this section, we follow this paper's construction, though for simplicity we provide only the weakest cryptosystem developed (one that is vulnerable to certain attacks). Moreover, we will define the new pairing only for the curve  $E$  given by  $y^2 = x^3 + 1$ .

Let  $p \equiv 2 \pmod 3$  be prime. We choose  $E$  as above because the pairing needed has a particularly natural description in terms of the automorphism  $\sigma : G(E, \mathbb{F}_{p^2}) \rightarrow G(E, \mathbb{F}_{p^2})$  given by  $(x, y) \mapsto (x, \zeta y)$ , where  $\zeta$  is a primitive third root of unity. One can show, moreover that  $|G(E, \mathbb{F}_p)| = p + 1$ , so the weakness proved in Theorem 18 applies here.

We fix a prime  $q \mid p + 1$  and a point  $P \in G(E, \mathbb{F}_p)$  of order  $q$ . Then the **modified Weil pairing** is the function  $\tilde{w}_q(P_1, P_2) = w_q(P_1, \sigma(P_2))$ , though we will be interested in its restriction to  $S \times \sigma(S)$ , where  $S = \langle P \rangle$ . The modified pairing, still bilinear, satisfies a different kind of nondegeneracy property: for  $P'$  a generator of  $G_q(E, \mathbb{F}_p)$ ,  $\tilde{w}_q(P', P')$  is a primitive  $q^{\text{th}}$  root of unity. We denote by  $C_q \subseteq \mathbb{C}$  the group of  $q^{\text{th}}$  roots of unity.

**Setup.** A secret integer  $s$  is chosen by the **Private Key Generator** (PKG), while  $sP = Q$  is made public, together with  $p, q$ , and  $P$ . Two cryptographic hash functions  $H_1 : \{0, 1\}^* \rightarrow S \setminus \{0\}$  and  $H_2 : C_n \rightarrow \{0, 1\}^m$  for some fixed  $m$  are also made public.

Each party comes to collect from the PKG their private key  $d_B$ , which the PKG generates from their identifier  $I_B$  by the formula  $d_B = sH_1(I_B)$ .

**Encryption.** To send a message  $m$  to Bob, who has identifier  $I_B$ , Alice generates  $r$  randomly from  $C_q$ , computes the encryption key  $e_B = \tilde{w}_q(H_1(I_B), Q)$ , and computes the cyphertext  $C = (rP, m \oplus H_2(e_B^r))$ . She sends  $C$  to Bob.

**Decryption.** Given the cyphertext  $C = (c_1, c_2)$ , Bob computes

$$\begin{aligned} c_2 \oplus H_2(\tilde{w}_q(d_B, c_1)) &= m \oplus H_2(e_B^r) \oplus H_2(\tilde{w}_q(sH_1(I_B), rP)) \\ &= m \oplus H_2(\tilde{w}_q(H_1(I_B), Q)^r) \oplus H_2(\tilde{w}_q(H_1(I_B), Q)^r) = m \end{aligned}$$

where the second equality is by linearity.

Nondegeneracy is critical in proving the security of this algorithm, which is proved in [BF].

As a final note, a simpler example of the utility of the Weil pairing is in the one-round tripartite matching protocol proposed by Antoine Joux in [Jo]. In this protocol, parties  $p_1, p_2, p_3$  have a public non-degenerate (in the sense defined in this section) bilinear map  $b : G \times G \rightarrow C_n$  such as the modified Weil pairing above and generator  $g \in G$ . They pick random secrets  $s_1, s_2, s_3$ , and  $p_i$  sends  $s_i g$  to every other player. Finally  $p_1$  computes  $b(s_2 g, s_3 g)^{s_1} = b(g, g)^{s_1 s_2 s_3}$ , as do each of the other players. The value of  $b(g, g)^{s_1 s_2 s_3}$  is the established common key.

The theory of pairings is one of the most active and interesting in elliptic curve cryptography. Indeed, there is an entire conference each year, called Pairing, dedicated solely to their study. There are additional pairings other than the Weil and modified Weil pairings discussed here, including the Tate and Tate-Lichtenbaum pairings, and many more applications. One can even define pairings on more general varieties than elliptic curves.

### 2.5.5 Practical Considerations

The value of elliptic curve cryptography, already illustrated by the discussion of pairings in the previous section, is increased by several more pragmatic considerations.

**Security.** The primary benefit of elliptic curve cryptography is that seemingly greater security may be achieved than for cryptosystems implementing the usual discrete logarithm problem. As explained in [HMV], the most efficient algorithms to solve the discrete logarithm problem over  $(\mathbb{Z}/p\mathbb{Z})^\times$  are in subexponential time, far more efficient than the best algorithms to solve the discrete logarithm problem on general elliptic curves (Pollard’s algorithm, discussed in Section 2.5.2.1, is essentially the best known method).

**Efficiency.** The Elgamal cryptosystem based on the usual discrete logarithm problem was discussed in the section on cryptography. This algorithm requires computations that in practice are far more time consuming than the equivalent computations on elliptic curves, where doubling and addition tend to be more efficient. Moreover, the security benefit mentioned earlier creates a new source of efficiency. The integers needed to do elliptic cryptography with comparable security to classical cryptosystems are usually a tenth of the size, creating a significant speedup in arithmetic on the curve.

**Versatility.** Elliptic curves, as illustrated by the factoring algorithm, are numerous for a given choice of ring  $\mathbb{Z}/n\mathbb{Z}$ . This creates an extra dimension of versatility and functionality missing in classical algorithms, and explains the huge improvement over Pollard’s factoring methods.

**Structure.** The Weil pairing and other structures defined on elliptic curve provide a rich range of tools. As Sections 2.5.2.2 and 2.5.4 reveal, the interaction of these structures on the curve can be helpful for designing cryptosystems or performing computational tasks.

It is not wise to immediately replace all one’s cryptosystems by those employing elliptic curve methods, however. Elliptic curves are very complex objects, and in most algorithms above, the curve  $E$  is published. The discrete log problem has been shown to be much easier on many subclasses of elliptic curves by results such as Theorem 18, so if one selects a curve with particular properties, many of which might not be readily apparent, an adversary could crack the cryptosystem using “special” techniques specific to that curve. That said, much work has indeed been done on the selection process to produce a curve with optimal security, and [CFA] and [HMV] provide substantial coverage of this.

The study of more sophisticated number theory such as modular forms and complex multiplication can have cryptographic implications, as discussed in [CFA]. A generalization of elliptic curve cryptography is hyperelliptic curve cryptography, which performs arithmetic on the Jacobian of hyperelliptic curves. Elliptic curve cryptography is a subject where deep number theory has direct impact on practical matters, and the results described in Sections 2.4 and 2.5 only scratch the surface.

## 2.6 Conclusions

For the reader interested in investigating this material further, the following books provide additional information on the topics discussed in this paper:

**Cryptography.** The approach taken by this paper most closely parallels the philosophy of [MvOV], which focuses on providing descriptions of real cryptosystems, rather than establishing the foundations of the subject. An amazing introduction to foundational cryptography can be found in [Go1] and [Go2].

**Elliptic Curves.** Three introductory texts on elliptic curves are [M2], [ST], and [Wa], which together cover all the material here and much more. For a more advanced introduction, look to [Si1] and [Si2] or [Hu]. For those interested in the complex-analytic aspects of the subject, [La] and [Fo] are excellent texts. From an algebro-geometric viewpoint, one might look into [Ha] or [M1].

**Elliptic Curve Cryptography.** For all-purpose text that is encyclopedic in both its coverage of the theory and practice of elliptic curve cryptography, look to [CFA]. The text [HMV] focuses on the practical aspects of the subject and is very readable.

The group of points on an elliptic curve can be used as a replacement for multiplicative groups in many situations. As we saw in the factoring algorithm, the key improvement was that factoring  $n$  via Pollard's algorithm could be reduced to "guessing" a number that is either a multiple of  $\varphi(n)$ , or has many factors in common with it, while over an elliptic curve,  $\varphi(n)$  is replaced with the group order, which can take on many values for fixed  $n$  as  $E$  varies. This added flexibility gives a new dimension to the development of cryptographic systems, since, as seen in the examples above, one can vary the curve  $E$  in setting parameters while keeping the field fixed. Although we did not delve into the more complex algorithms that make more important use of this phenomenon, the factoring algorithm shows that elliptic curves can provide important computational savings. We also saw that structures such as the Weil pairing can give rise to cryptosystems for which no simple implementation using classical methods are known. For their security, efficiency, and versatility, elliptic curve methods are used for real cryptographic applications every day, revealing their amazing pervasiveness in both theory and applications within mathematics and computer science.

## References

- [BF] Dan Boneh and Matt Franklin: Identity-Based Encryption from the Weil Pairing, *Advances in Cryptology-Crypto 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings* (2001).
- [Bo] Dan Boneh: Twenty years of attacks on the RSA cryptosystem, *Notices of the Amer. Math. Soc.* **46** #2 (1999), 203–213.
- [CFA] Henri Cohen, Gerhard Frey, and Roberto Avanzi: *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Boca Raton: CRC Press 2006.
- [CLRS] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein: *Introduction to Algorithms*. Cambridge, MA: MIT Press, McGraw-Hill 2001.
- [Fo] Otto Forster: *Lectures on Riemann Surfaces*. New York: Springer-Verlag 1981.
- [Ga] Steven D. Galbraith: The Weil Pairing on Elliptic Curves over  $\mathbb{C}$ , preprint (2005).
- [GB] Shafi Goldwasser and Mihir Bellare: Lecture notes on cryptography, *Summer Course, "Cryptography and Computer Security," at MIT* 1996.
- [Go1] Oded Goldreich: *Foundations of Cryptography: Volume 1, Basic Tools*. New York: Cambridge Univ. Press 2001.
- [Go2] Oded Goldreich: *Foundations of Cryptography: Volume 2, Basic Applications*. New York: Cambridge Univ. Press 2004.
- [Ha] Robin Hartshorne: *Algebraic Geometry*. New York: Springer-Verlag 1977.
- [HMV] Darrel R. Hankerson, Alfred J. Menezes, and Scott A. Vanstone: *Guide to Elliptic Curve Cryptography*. New York: Springer-Verlag 2004.
- [Hu] Dale Husemoller *Elliptic curves*. New York: Springer-Verlag 2004.
- [Jo] Antoine Joux: A One Round Protocol for Tripartite Diffie–Hellman, *J. of Cryptology* **17** #4 (2004), 263–276.
- [Ko] Neal I. Koblitz: *A Course in Number Theory and Cryptography*. New York: Springer-Verlag 1994.
- [La] Serge Lang: *Elliptic Functions*. New York: Springer-Verlag 1987.
- [Le] Hendrik W. Lenstra: Elliptic curves and number-theoretic algorithms, (1986).

- [MOV] Alfred J. Menezes, Tatsuaki Okamoto, and Scott A. Vanstone: Reducing elliptic curve logarithms to logarithms in a finite field, *Information Theory, IEEE Transactions on*, **39** #5 (1993), 1639–1646.
- [MvOV] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone: *Handbook of Applied Cryptography*. Boca Raton: CRC Press 1997.
- [M1] James S. Milne: Abelian varieties, course notes.
- [M2] James S. Milne: Elliptic curves, course notes.
- [Shaf] Igor R. Shafarevich: *Basic Algebraic Geometry, Volume 1*. New York: Springer-Verlag 1994.
- [Sham] Adi Shamir: Identity-based cryptosystems and signature schemes, *Proceedings of CRYPTO 84* (1985), 47–53.
- [Si1] Joseph H. Silverman: *The Arithmetic of Elliptic Curves*. New York: Springer-Verlag 1986.
- [Si2] Joseph H. Silverman: *Advanced Topics in the Arithmetic of Elliptic Curves*. New York: Springer-Verlag 1994.
- [ST] Joseph H. Silverman and John Tate: *Rational Points on Elliptic Curves*. New York: Springer-Verlag 1992.
- [Si] Michael Sipser: *Introduction to the Theory of Computation*. Course Technology 1996.
- [Wa] Lawrence C. Washington: *Elliptic Curves: Number Theory and Cryptography*. Boca Raton: Chapman & Hall/CRC Press 2003.