DRP Fall 2020 Report: Elliptic Curve Cryptography

Sunay Joshi

July 3, 2026

1 Introduction

How can we communicate in private? Cryptography is the ancient field that offers answers to this question.

The earliest systems used **private key cryptography**, also known as symmetric encryption. Suppose Alice in Bob want to communicate. In private key cryptography, Alice and Bob share a key that allows them to communicate. If Alice wants to send Bob a message, she encrypts her message with the key, sends the encrypted message to Bob, and Bob decrypts the message with the same key. One early example of private key system was the Caesar cipher, in which the key represents the size of a cyclic shift of the alphabet. If Alice and Bob agree on the key $k = 1$, Alice could encrypt the message"MATH" as "NBUI" [10].

In order to set up a private key system, Alice and Bob must have a secure channel to initially transport their shared key. However, in our digital age, a secure channel is hard to come by. Adversaries are constantly trying to steal our information on the web. A new insight that got around this issue came in 1976. Diffie, Hellman, and Merkle introduced **public key cryptography**, also known as asymmetric encryption. In public key cryptography, if Alice wants to send Bob a message, she looks up his publically-available public key pk, encrypts her message with pk, and sends it to Bob. Then, Bob decrypts the message with his private/secret key sk. The encryption scheme is chosen so that decrypting the message would be practically impossible without knowing sk. Such a system avoids the need for a secure channel [10]. It even allows Alice and Bob to set up symmetric encryption, as outlined in the Diffie-Hellman-Merkle key exchange protocol [8].

How do you choose an encryption scheme? Many encryption schemes are based on the difficulty of the **discrete log problem** (**DLP**) in certain groups. Consider $\mathbb{Z}_p$, the integers modulo p. Given a generator $g \in \mathbb{Z}_p^\times$ and $y = g^x \in \mathbb{Z}_p^\times$ for an unknown x, DLP asks you to find $\log_g y = x$. No fast classical algorithms are known for solving DLP, which makes it perfect for designing encryption schemes [2].

Historically, public key cryptosystems utilized DLP over $\mathbb{Z}_p$. However, in 1985, Koblitz and Miller independently proposed a radical new idea: instead of working with the group $\mathbb{Z}_n$, why not

use $E(\mathbb{F}_q)$, the group of points in the finite field $\mathbb{F}_q$ on a so-called **elliptic curve** E? This led to the field of **elliptic curve cryptography**, or **ECC** [14].

What is an elliptic curve? It is a type of cubic curve that can normally be described as $y^2 = x^3 + Ax + B$. Its key property is the group structure of its points. Its history can be traced back to the 18th century, when mathematicians were investigating the arclength of an ellipse. This led to elliptic integrals, which have a convenient representation in terms of elliptic functions. The work of Abel and Weierstrass showed the connection between these objects and the elliptic curves we study today, and paved the way for the study of the group law starting in the 1860s [13]. It turns out that ECC has numerous advantages to traditional approaches using DLP in $\mathbb{Z}_p$. Most importantly, public and private keys are shorter in ECC. As a result, ECC is widely used today [9].

One major application of public key cryptography is the concept of a **digital signature**. These are used to authenticate communication on the internet and are crucial to everything from emails to file storage to electronic money [11]. The two most important algorithms are the **Digital Signature Algorithm** (**DSA**) and the **Elliptic Curve Digital Signature Algorithm** (**ECDSA**) [1].

This paper is structured as follows. In section 2, we will cover the basic theory behind elliptic curves. In section 3, we will explain DLP, DSA, and ECDSA. In section 4, we will present attacks on DLP. In section 5, we will discuss the future of ECC and offer recommendations for further reading. The prerequisites are basic number theory and abstract algebra.

2 Basic theory

In this section, we present the basic theory of elliptic curves, including the group law, torsion subgroups, results on the size of $E(\mathbb{F}_q)$, and the Weil pairing. Throughout, we assume the characteristics of the fields used are neither 2 nor 3. Most of the material is from [2] and [4].

2.1 Elliptic curves

Definition 2.1 (Elliptic curve)**.** *An elliptic curve over a field K with $charK \neq 2, 3$ is the set of points $(x, y) \in K \times K$ on a cubic equation of the form*

$$E : y^2 = x^3 + Ax + B$$

that is nonsingular, i.e. the discriminant $D = -(4A^3 + 27B^2)$ is nonzero.

Remark 2.2. *Geometrically, a curve is nonsingular if it has no cusps or self-intersections. A tangent line can be drawn at each point. This will be crucial for the group law in the next subsection.*

The set of points on E with coordinates in K will be denoted $E(K)$. We add one extra point to the curve, the point at infinity ∞. Its significance will be explained in the next subsection. For cryptography, K is often a finite field $\mathbb{F}_q$. A typical graph of an elliptic curve over $\mathbb{R}$ is as follows:

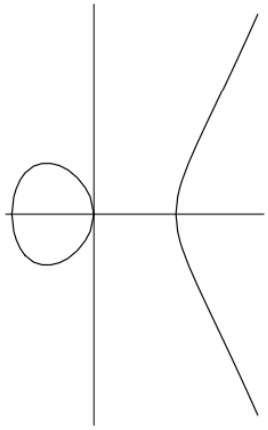


Figure 1: Graph of $y^2 = x^3 - x$ from [2], p 10

Remark 2.3. *Although elliptic curves over finite fields can't be graphed, the geometric intuition from the above diagram allows us to formulate a general group law, as we will see in the next subsection.*

2.2 Group law

The most important property of elliptic curves is the **group law**. Given any two points, we can "add" them to get a third one on the curve. This operation turns $E(K)$ into a group. The group law is more complicated than simple component-wise addition, and is best described geometrically. Consider the graph of a generic elliptic curve, and pick two points $P_1, P_2 \in E(K)$. The following two steps yield $P_1 + P_2$:

- Draw the line P_1P_2, and let it intersect E for a third time at P_3'.
- Draw the vertical line through P_3', and let it intersect E for a second time at $P_3 =: P_1 + P_2$.

Remark 2.4. *There are two special cases of the group law. If $P_1 = P_2 = P$, we draw the tangent line to E at P in the first step. If $P_2 = \infty$, we draw the vertical line through P_1 in the first step. Therefore ∞ serves as the identity for the group $E(K)$, and is located infinitely far above and below the y axis, lying on every vertical line.*

This yields the following algebraic form of the group law.

Definition 2.5 (Group law)**.** *Given $E : y^2 = x^3 + Ax + B$, if $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, and m is the slope of P_1P_2, then the sum $P_3 = P_1 + P_2 = (x_3, y_3)$ has the following coordinates:*

- *If $x_1 \neq x_2$, $x_3 = m^2 - x_1 - x_2$ and $y_3 = m(x_1 - x_3) - y_1$, where $m = \frac{y_2 - y_1}{x_2 - x_1}$;*
- *If $x_1 = x_2$ and $y_1 \neq y_2$, then $P_3 = \infty$;*
- *If $P_1 = P_2$ and $y_1 \neq 0$, then $x_3 = m^2 - 2x_1$ and $y_3 = m(x_1 - x_3) - y_1$, where $m = \frac{3x_1^2 + A}{2y + 1}$;*

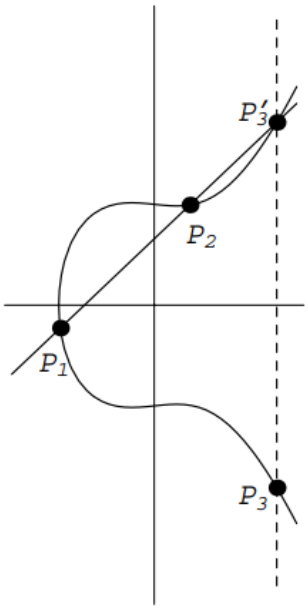


Figure 2: Graph from from [2], p 12

- *If* $P_1 = P_2$ *and* $y_1 = 0$*, then* $P_3 = \infty$*;*
- *If* $P_1 = \infty$*,* $P_3 = P_2$*, and vice versa.*

Further, $(E(K), +)$ *is an abelian group.*

Exercise 2.6. *Derive the group law. (Hint: When solving for* x_3*, use Vieta's Formulas.)*

2.3 Torsion points

The following result tells us more about the structure of $E(K)$.

Theorem 2.7 (Mordell–Weil)**.** *Let* K *be a number field and let* E *be an elliptic curve over* K*. Then* $E(K)$ *is finitely generated [3].*

Recall that any finitely generated abelian group can be written as $G \simeq \mathbb{Z}^r \oplus G_{\text{torsion}}$, where G is the **torsion subgroup** of points of finite order.

Definition 2.8 (n torsion points)**.** *Let* $E[n]$ *denote the set of* n *torsion points with coordinates in the algebraic closure* $\overline{K}$ *of* K*, i.e. all* P *with* $nP = 0$*, i.e. all points of order dividing* n*.*

We have the following structure theorem for $E[n]$.

Theorem 2.9 (Structure of $E[n]$)**.** *If* $charK \nmid n$ *or* $charK = 0$*, then* $E[n] \simeq \mathbb{Z}_n \oplus \mathbb{Z}_n$*. Else, if* $charK = p$ *and* $n = p^r n'$ *with* $p \nmid n'$*, then* $E[n] \simeq \mathbb{Z}_{n'} \oplus \mathbb{Z}_{n'}$ *or* $\mathbb{Z}_n \oplus \mathbb{Z}_{n'}$*.*

2.4 Structure and size of $E(\mathbb{F}_q)$

In this subsection, we restrict to $K = \mathbb{F}_q$. In this case, as $\mathbb{F}_q$ is finite, $E(\mathbb{F}_q)$ is a finite abelian group. We have the following structure theorem for $E(\mathbb{F}_q)$.

Theorem 2.10 (Structure of $E(\mathbb{F}_q)$)**.** *$E(\mathbb{F}_q) \simeq \mathbb{Z}_n$ or $\mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$ with $n_1 | n_2$.*

Proof. Any finite abelian group is isomorphic to $\mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2} \cdots \oplus \mathbb{Z}_{n_r}$, with $n_i | n_{i+1}$ for all $1 \leqslant i < r$. Such a group has n_1^r elements of order dividing n_1. By the structure theorem for $E[n]$, there are at most n_1^2 such points. Thus $r \leqslant 2$, as desired. □

Next, we study the size of $E(\mathbb{F}_q)$. To start, we provide a bound.

Theorem 2.11 (Hasse bound)**.** *Let $a = q + 1 - \#E(\mathbb{F}_q)$. Then $|a| \leqslant 2\sqrt{q}$. In other words, $\#E(\mathbb{F}_q)$ lies in the* ***Hasse interval*** *$[q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$ around $q + 1$.*

There are a number of techniques to find the exact size of $E(\mathbb{F}_q)$. One technique is to use points of large order. If there is a point P of large order, since Lagrange's Theorem implies $\operatorname{ord} P | \#E(\mathbb{F}_q)$, there are only a few possibilities for $\#E(\mathbb{F}_q)$ in the Hasse interval. The best case is when there is only 1 multiple of $\operatorname{ord} P$!

It turns out that points of large order are almost always guaranteed. All that is left is to have an algorithm to find them. The **Baby Step Giant Step algorithm** (**BSGS**) offers a way to find $\operatorname{ord} P$ in any group G, though below, we present it for $E(\mathbb{F}_q)$. Instead of brute force checking all $4\sqrt{q}$ elements of the Hasse interval, it takes only $4q^{1/4}$ steps. The main idea is to find some M with $MP = 0$, and then divide M by its prime factors to find a good candidate for $\operatorname{ord} P$. Note that the while loop is guaranteed to terminate as M is finite.

Algorithm 2.12: BSGS

Input: $E, q, P \in E(\mathbb{F}_q)$;
Output: $\operatorname{ord} P$;
Initialize M;
Step 1: Find M in Hasse interval such that $MP = \infty$
$Q = (q+1)P$;
Pick $m > q^{1/4}$;
$a = [jP \text{ for } j \text{ in } [-m, m]]$;
for k *in* $[-m, m]$ **do**
  if $Q + k(2mP) = jP \in a$ **then**
    $M = q + 1 + 2mk - j$;
  end
end
Step 2: Find prime factors of M
$b = [p_i \text{ for } p_i | M]$;
# using Pollard ρ and fast primality testing, for instance
# Step 3: Find $\operatorname{ord} P$ using M and b
while $\exists p_i \in b$ *such that* $p_i | M$ *and* $\frac{M}{p_i} P = \infty$ **do**
  $M \leftarrow \frac{M}{p_i}$;
end
return M;

2.5 Weil pairing

Definition 2.13 (Weil pairing)**.** *Suppose $E : y^2 = x^3 + Ax + B$ is an elliptic curve over K and n is an integer with $charK \nmid n$. Define the roots of unity $\mu_n = \{x \in \overline{K} | x^n = 1\}$. The* ***Weil pairing*** *$e_n : E[n] \times E[n] \to \mu_n$ is a nondegenerate bilinear map that satisfies:*

- $e_n(T, T) = 1$ *for all* T.
- $e_n(T, S) = e_n(S, T)^{-1}$ *for all* S, T.
- $e_n(\sigma S, \sigma T) = \sigma(e_n(S, T))$ *for all automorphisms* σ *of* $\overline{K}$ *that fix* A, B.
- $e_n(\alpha(S), \alpha(T)) = e_n(S, T)^{\deg(\alpha)}$ *for all separable endomorphisms* α *of* E.

One consequence of the existence of the Weil pairing is the following.

Lemma 2.14. *Show that if $\{T_1, T_2\}$ is a basis for $E[n]$ (see the structure theorem for $E[n]$), then $\zeta = e_n(T_1, T_2)$ is a primitive nth root of unity (i.e. a generator of μ_n).*

Proof. Suppose that $\zeta^d = 1$ for some $d \geqslant 1$. Then by the bilinearity of e_n, $e_n(T_1, T_2)^d = e_n(T_1, dT_2) = 1$. Suppose $P \in E[n]$. Then as $\{T_1, T_2\}$ is a basis for $E[n]$, $P = aT_1 + bT_2$ for some a, b. Thus

$$e_n(P, dT_2) = e_n(aT_1 + bT_2, dT_2) = e_n(aT_1, dT_2) e_n(bT_2, dT_2) = e_n(T_1, dT_2)^a e_n(T_2, T_2)^{bd} = 1,$$

hence $e_n(P, dT_2) = 1$ for all $P \in E[n]$. As e_n is nondegenerate, we must have $dT_2 = \infty$. But $\operatorname{ord} T_2 = n$, thus $n|d$, as desired. □

Using the above, we can show that the Weil pairing "detects" when a point is a multiple of another.

Lemma 2.15. *Show that if* $\operatorname{ord} P = n$ *with* $\mathrm{char} K \nmid n$ *and* $Q \in E[n]$*, then* $Q = kP$ *for some* k *iff* $e_n(P, Q) = 1$*.*

Proof. For the forwards direction, note $Q = kP \implies e_n(P, kP) = e_n(P, P)^k = 1$.

For the backwards direction, let $\{P, R\}$ be a basis for $E[n]$, and suppose $Q = kP + \ell R$ for some k, ℓ. Then

$$e_n(P, Q) = 1 \implies e_n(P, kP + \ell R) = e_n(P, P)^k e_n(P, R)^\ell = 1 \implies e_n(P, R)^\ell = 1.$$

But the lemma above implies $e_n(P, R)$ is a primitive nth root of unity, hence $n|\ell$ and $Q = kP$. □

Remark 2.16. *To construct the Weil pairing requires the theory of divisors, and to compute the Weil pairing requires efficient algorithms like* ***Miller's algorithm****. However, we won't get into the details here. We will use the Weil pairing later to design an attack on ECC.*

3 Digital signatures

In this section, we introduce DLP as a basis for identification schemes and digital signatures. Then we intuitively build up DSA and ECDSA from identification schemes and the Fiat–Shamir transform. Most of the material is from [1].

3.1 DLP

The discrete log problem in a group is the basis for many encryption schemes. As mentioned in the introduction, DLP in $G = \mathbb{Z}_p^\times$ is: given g and $y = g^x$, find x. It is thought to be hard for large primes p. This extends to any group.

Definition 3.1 (DLP)**.** *In a group* G*, given* g *and* $y = g^x$*, find* x*.*

Remark 3.2. *In the case of* $G = E(K)$ *(or any other abelian group), DLP can be phrased additively: given* P *and* $Q = kP$*, find* k*.*

3.2 Identification schemes

In an **identification scheme**, a prover wants to prove that it knows a private key to a verifier. For example, you might be the prover, and the verifier might be amazon.com. Since this is a

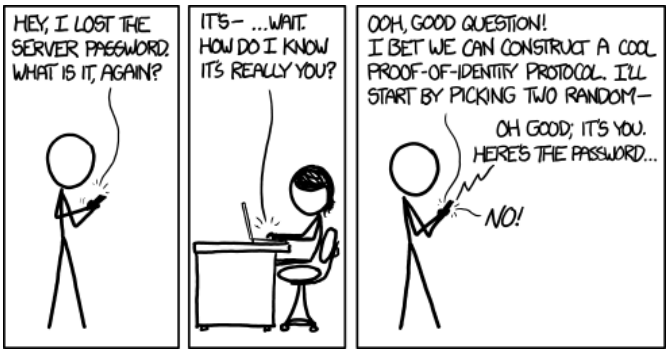


Figure 3: Image from [16]

public key system, the prover has public and private keys pk, sk, respectively. Further, the prover has algorithms P_1, P_2 that will be used in the scheme.

Below, we present an example of a type of 3-round identification scheme. The main idea is as follows: the verifier will use pk to construct a challenge that only someone who knows sk could solve. As we'll see in the following subsections, this challenge will be based on DLP.

Algorithm 3.3: 3-round identification scheme

Input: pk, sk, P_1, P_2, V;
Output: Accept or reject;
Prover sends verifier initial message $I = P_1(sk)$;
Verifier sends prover challenge r determined by pk;
Prover sends verifier response $s = P_2(sk, r)$;
if $V(pk, r, s) = I$ **then**
  Verifier accepts;
end
else
  Verifier rejects;
end

Note that for this protocol to be effective, V should be constructed so that $V(pk, r, P_2(sk, r)) = I$ and $V(pk, r, \tilde{s}) \neq I$ with high probability if $\tilde{s} \neq P_2(sk, r)$.

Below is an illustration of Algorithm 3.3, where arrows indicate messages being sent between the prover and verifier. Here, st is a so-called state and Ω_{pk} is the set of challenges defined by pk. Although this is abstract, we will see specific examples in the coming subsections.

3.3 Digital signatures and the Fiat–Shamir transform

In a digital signature scheme, an individual wants to prove that a given message belongs to him or her [6]. For example, if you digitally sign an email, it allows recipients to check that the email really was from you.

The Fiat–Shamir transform allows us to generate digital signature schemes from identifications

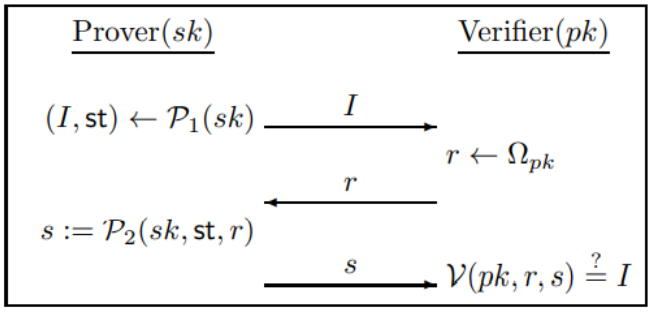


Figure 4: Image from [1], p 452

schemes. Suppose you want to generate a signature for a message m. The main idea is to run the identification scheme above by yourself.

Remark 3.4. *Algorithm 3.5 below uses a hash function to generate the challenge r. A* ***hash function*** *H is a function that takes inputs of some length and compresses them into short, fixed-length outputs. A good hash function is* ***collision resistant****, meaning it is computationally infeasible to find x, x' with $H(x) = H(x')$.*

Algorithm 3.5: Fiat–Shamir transform

Input: m, H;
Output: Signature (r, s);
Generate I;
initial message
$r = H(I, m)$;
challenge for I
$s =$ response to r;
response to challenge
return *(r,s)*;

Remark 3.6. *To verify the signature, a person computes $I = V(pk, r, s)$ and verifies $H(I, m) = r$. Intuitively, this scheme is secure because if an adversary can come up with (r, s), they can impersonate the prover in the original identification scheme. Again, this is abstract, but a modified version of the transform will allow us to explain DSA/ECDSA.*

3.4 Schnorr identification scheme

The **Schnorr identification scheme** is one implementation of the outline given above. It works with a group G in which DLP is hard, and is a stepping stone to DSA/ECDSA in the next section.

In the Schnorr scheme, the public key is $pk = (G, p, g, y)$, where $|G| = p$ is prime, $y = g^x$, and the private key is the exponent $sk = x$. The challenge that the verifier poses to the prover is to solve $g^s \cdot y^{-r} = g^k$ for s, for a random $r \in \mathbb{Z}_p^\times$. This is easy for someone with x, but due to the hardness of DLP, it is difficult for someone without x.

Exercise 3.7. *If you knew x, what would the solution to the above equation be?*

Algorithm 3.8: Schnorr identification scheme

Input: $pk = (G, p, g, y)$ with $|G| = p$ prime and $y = g^x$, $sk = x$;
Output: Accept or reject;
Prover sends verifier initial message $I = g^k$ for $k \in \mathbb{Z}_p^\times$ random;
Verifier sends prover challenge $r \in \mathbb{Z}_p^\times$ at random;
Prover sends verifier response $s \equiv rx + k \pmod{p}$;
if $g^s \cdot y^{-r} = I$ **then**
  Verifier accepts;
end
else
  Verifier rejects;
end

Regarding correctness, note that for $s \equiv rx + k \pmod{p}$, $g^s \cdot y^{-r} = g^{rx+k} \cdot g^{-rx} = g^k = I$, as desired. Here, we used the fact that $g^{|G|} = e \implies g^p = e$ by Lagrange's Theorem in the group G.

Regarding security, if an adversary could efficiently imitate the prover twice without knowing x, then they could efficiently solve DLP. To see this, note that the adversary would be able to fool the verifier on two different challenges r_1, r_2. But this implies the adversary knows

$$g^{s_1} \cdot y^{-r_1} = g^{s_2} \cdot y^{-r_2} \implies \log_g y = (s_1 - s_2)(r_1 - r_2)^{-1},$$

the discrete log of y with respect to g! This contradicts our assumption that DLP is hard in G.

3.5 DSA and ECDSA

DSA/ECDSA both rely on the same identification scheme. This scheme has the same set-up as the Schnorr scheme, with an extra layer of security.

The underlying identification scheme is as follows: the public key is $pk = (G, p, g, y)$, where $|G| = p$ and $y = g^x$, and the private key is the exponent $sk = x$. The challenge that the verifier poses to the prover is to solve $g^{\alpha s^{-1}} \cdot y^{rs^{-1}} = g^k$ for s, for random $r, \alpha \in \mathbb{Z}_p$. This is easy for someone with x, but due to the hardness of DLP, it is difficult for someone without x.

Algorithm 3.9: DSA/ECDSA identification scheme

Input: $pk = (G, p, g, y)$ with $|G| = p$ prime and $y = g^x$, $sk = x$;
Output: Accept or reject;
Prover sends verifier initial message $I = g^k$ for $k \in \mathbb{Z}_p^\times$ random;
Verifier sends prover challenge $r, \alpha \in \mathbb{Z}_p$ at random;
Prover sends verifier response $s \equiv k^{-1}(\alpha + rx) \pmod{p}$;
if $g^{\alpha s^{-1}} \cdot y^{rs^{-1}} = g^k$ **then**
  Verifier accepts;
end
else
  Verifier rejects;
end

Exercise 3.10. *Show that the given s works.*

The proof of security is similar to the security of the Schnorr scheme. To convert this identification scheme to a digital signature scheme, we apply a variant of the Fiat–Shamir transform as follows:

Algorithm 3.11: Modified Fiat–Shamir transform

Input: $m, F : G \to \mathbb{Z}_p, H$;
Output: Signature (r, s);
Generate I;
initial message
$r = F(I)$;
first component of challenge for I
$\alpha = H(m)$;
second component of challenge for I
$s =$ response to r, α;
response to challenge
return *(r,s)*;

Remark 3.12. *The difference between DSA and ECDSA is only the choice of G. DSA uses $G \leqslant \mathbb{Z}_\ell^\times$ for ℓ prime, while ECDSA uses $G \leqslant E(\mathbb{Z}_\ell)$ for ℓ prime. In DSA, the "simple" function F is simply $I \pmod{p}$, while in ECDSA, $F(x, y) = x \pmod{p}$. ECDSA offers much smaller key sizes for the same amount of security. To achieve 128 bits of security, DSA requires a 3072 bit key, while ECDSA only requires a 128 bit key [7].*

Remark 3.13. *The* ***random oracle*** *model of a hash function assumes that its outputs are truly random. It has been shown that the DSA/ECDSA signature schemes are secure under the random oracle model for H and F, but surprisingly, this has not been shown unconditionally. Although it cannot be ensured that F and H are truly random, DSA/ECDSA are both still widely used!*

4 Attacks on DLP

In the last section, we described cryptosystems that utilize the difficulty of DLP. In this section, we describe a few attacks on the DLP. The first works for any group, and the second is specific to elliptic curves. Most of this is from [2].

4.1 General attacks

A few sections ago, we used BSGS to find $\operatorname{ord} P$ given $P \in E(\mathbb{F}_q)$, i.e. M such that $MP = \infty$. Now, we modify it in a more general setting to find k such that $kP = Q$ given a group G, its order N, and $P, Q \in G$.

Algorithm 4.1: BSGS attack

Input: $G, N = |G|, P, Q \in G$;
Output: k such that $Q = kP$;
Initialize k;
Pick $m \geqslant \sqrt{N}$;
$P' = mP$;
$a = [iP \text{ for } i \text{ in } [0, m-1]]$;
for j *in* $[0, m-1]$ **do**
  if $Q - jP' = iP \in a$ **then**
    $k = i + jm \pmod N$;
    return k;
  end
end

Remark 4.2. *There are numerous other general attacks on the DLP, and each has its own advantages. For reference, there are also the index calculus, Pollard ρ, and Pohlig–Hellman methods.*

4.2 Elliptic curve-specific attacks

Introducing a new tool also introduces new vulnerabilities. In the case of ECC, these vulnerabilities are much more complicated than the general attacks of the previous subsection. Here, we will focus on the **Menezes–Okamoto–Vanstone** (**MOV**) **attack**. In short, the MOV attack uses the Weil pairing to reduce DLP in $E(\mathbb{F}_q)$ to DLP in $\mathbb{F}_{q^m}^\times$.

The set-up is: we are given $P, Q \in E(\mathbb{F}_q)$, we know $\operatorname{ord} P = N$, we are guaranteed $\gcd(q, N) = 1$, and we want to find k such that $Q = kP$.

The main idea is to choose $T_1 \in E[N]$ at random. Let $d = \operatorname{ord} T_1$; if $d > 1$ continue, otherwise pick a new T_1. Then, as $Q = kP$, the bilinearity of e_n implies $e_n(Q, T_1) = e_n(P, T_1)^k$. But as e_n maps into μ_d, and since $\mu_d \subset \mathbb{F}_{q^m}^\times \subset \overline{\mathbb{F}}_q$ for some m, this is just DLP in $\mathbb{F}_{q^m}^\times$ for some m. This

yields $k \pmod d$, and repeating yields enough information to recover k via the Chinese Remainder Theorem.

Algorithm 4.3: MOV attack

Input: $q, P, Q \in E(\mathbb{F}_q), N = \operatorname{ord} P$ such that $\gcd(q, N) = 1$;
Output: k such that $Q = kP$;
Declare K;
$D = 1$;
# Repeat until K determined $\pmod N$
while $D < N$ **do**
  # Step 1: Find a random point $T_1 \in E[N]$
  $d = 1$;
  while $d = 1$ **do**
    Pick $T \in E[N]$ randomly;
    $M = \operatorname{ord} T$;
    $d = \gcd(M, N)$;
  end
  $T_1 = \frac{M}{d} T$;
  # Step 2: Find $k \pmod d$
  $\zeta_1 = e_N(P, T_1)$;
  $\zeta_2 = e_N(Q, T_1)$;
  $k = \log_{\zeta_1} \zeta_2 \pmod d$;
  # DLP in $\mathbb{F}_{q^m}^\times$
  $K =$ solution to $x \equiv K \pmod D$ and $x \equiv k \pmod d$ (unless K not initialized); # CRT
  $D = \operatorname{lcm}(D, d)$;
end

Remark 4.4. *One can show that $d = 1$ occurs with low probability, thus the algorithm computes k in a few iterations.*

Remark 4.5. *MOV does not completely break ECC. The size of the resulting finite field $\mathbb{F}_{q^m}$ might be very large, in which case DLP remains hard.*

5 Future of ECC and Further Reading

While ECC has allowed for secure communication for decades, it is vulnerable to an emerging cryptographic menace: **quantum computing**. Quantum computers are much more powerful than classical computers. They gain this power from their use of **qubits**, which can exist in a superposition of states. While classical bits are either 0 or 1, qubits can exist as a mix of the two through the laws of quantum mechanics [12]. **Shor's Algorithm**, discovered in 1994, provides a quantum algorithm to efficiently solve DLP in abelian groups, including $E(\mathbb{F}_q)$ is abelian [14]. As a result, post-quantum cryptography requires new algorithms to secure our communications.

Nevertheless, ECC is still useful for two main reasons. First of all, practical quantum computers remain distant, and ECC will remain method of choice for many years [14]. Second of all, elliptic

curves can be used to design systems that may be quantum secure. One such cryptosystem is **supersingular isogeny key exchange**. It centers around **isogenies**, certain maps between elliptic curves [5].

For further reading, all of the resources in the bibliography are great resources. For this DRP, I mainly used two textbooks: "Rational Points on Elliptic Curves" by Joseph H. Silverman and "Elliptic Curves: Number Theory and Cryptography" by Lawrence C. Washington. When reading Washington, I followed the following course syllabus closely: `http://theory.stanford.edu/~dfreeman/cs259c-f11/syllabus.html`.

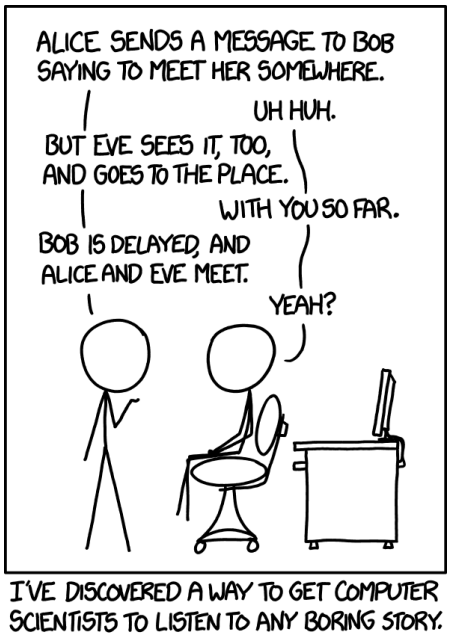


Figure 5: Image from [15]

6 Acknowledgements

I would like to thank my mentor Katharine Woo for guiding me through this DRP, introducing me to so much interesting math, and proofreading this report. I would also like to thank Érico Silva for organizing the DRP this semester.

References

[1] Jonathan Katz **and** Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007. ISBN: 1584885513.

[2] Lawrence C. Washington. *Elliptic Curves: Number Theory and Cryptography, Second Edition*. 2 **edition**. Chapman & Hall/CRC, 2008. ISBN: 9781420071467.

[3] Joseph H Silverman. *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics. Dordrecht: Springer, 2009. DOI: 10.1007/978-0-387-09494-6. URL: https://cds.cern.ch/record/1338326.

[4] Joseph H. Silverman **and** John T. Tate. *Rational Points on Elliptic Curves*. 2nd. Springer Publishing Company, Incorporated, 2015. ISBN: 331918587X.

[5] Luca De Feo. "Mathematics of Isogeny Based Cryptography". **in***CoRR*: abs/1711.04062 (2017). arXiv: 1711.04062. URL: http://arxiv.org/abs/1711.04062.

[6] DocuSign. *Understanding digital signatures*. URL: https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq.

[7] Virag Mody. *Comparing SSH Keys - RSA, DSA, ECDSA, or EdDSA?* URL: https://goteleport.com/blog/comparing-ssh-keys.

[8] Sebastian Pauli. *Diffie Hellman key exchange*. URL: https://mathstats.uncg.edu/sites/pauli/112/HTML/secdiffiehellman.html.

[9] Doug Ravenel. *Elliptic curves: what they are, why they are called elliptic, and why topologists like them, I*. URL: https://people.math.rochester.edu/faculty/doug/mypapers/wayne1.pdf.

[10] Eric Robert. *The History of Cryptography*. URL: https://cs.stanford.edu/people/eroberts/courses/soco/projects/public-key-cryptography/history.html.

[11] Sarah Simpson. *Cryptography in Everyday Life*. URL: http://www.laits.utexas.edu/~anorman/BUS.FOR/course.mat/SSim/life.html.

[12] Veritasium. *How Does a Quantum Computer Work?* URL: https://www.youtube.com/watch?v=g_IaVepNDT4.

[13] Steven Wepster. *Elliptic functions and elliptic curves: 1840-1870*. URL: https://webspace.science.uu.nl/~wepst101/elliptic/elliptic3.pdf.

[14] Jeremy Wohlwend. *Elliptic Curve Cryptography: Pre and Post Quantum*. URL: https://math.mit.edu/~apost/courses/18.204-2016/18.204_Jeremy_Wohlwend_final_paper.pdf.

[15] XKCD. *Protocol*. URL: https://xkcd.com/1323/.

[16] Explain XKCD. *Identity*. URL: https://www.explainxkcd.com/wiki/index.php/1121:_Identity.