Elliptic Curve Cryptography

Matthew Bielskas, Ben Hughes

Elliptic Curve Cryptography is a public-key cryptographic approach which relies upon the difficulty of the discrete logarithm problem (DLP) on groups defined by elliptic curves over finite fields. The scheme can be generally used in public-key settings such as key exchange, encyrption, and tagging. Corresponding to these security games are Elliptic Curve Diffie-Hellman (ECDH), Elliptic Curve Integrated Encryption Scheme (ECIES), and Curve Digital Signature Algorithm (ECDSA). Other protocols involving elliptic curves also exist, but in this paper we will focus on describing ECDH and ECDSA. We will also develop the preliminary theory required to understand elliptic curves, their strengths, and some problems.

1 Introduction

With protocols like RSA and Diffie-Hellman already providing assurances of security, why introduce the complication of elliptic curves? The answer is efficiency. The most efficient known algorithm to factor a large n, the general number field sieve (GNFS), runs in about $O((\log n)^{1/3})$ time- this super-polynomial algorithm forces us to choose very large primes to get comparable security in the private key setting. The best algorithms to compute DLP on elliptic curves, however, run in $O(\sqrt{n})$, where n is the size of the group. This means that achieving 128-bit security in the private key setting, say with AES-128, requires only 256-bit keys with elliptic curves, whereas RSA would require 3072-bit key! The difference in size alone accounts for much of the speed-up, but there are also other optimizations to consider. First, we must establish some theory.

2 Preliminaries

Definition 2.0.1. *A **group** is a set with a binary operation, which we'll denote with "+", and has the following properties:*

- ***Closed:** if $a, b \in G$, then $a + b \in G$*
- ***Identity:** there exists a $e \in G$ such that for all $a \in G$ $a + e = e + a = a$*
- ***Inverses:** for every $a \in G$ there exists $b \in G$ such that $a + b = e$. We denote b = -a*
- ***Associative:** $(a + b) + c = a + (b + c)$ for all $a, b, c \in G$*

*The group may also be **commutative** if it satisfies:*

- *For all $a, b \in G$, $a + b = b + a$*

*This makes it an **abelian** group.*

The addition notation makes sense when dealing with an abelian group. In general, multiplicative notation is used, so we have $a + a + ... + a = n \cdot a$ for an abelian group and $a \cdot a \cdots a = a^n$ in general. Later on we will need more group structure, namely that of cyclic groups:

Definition 2.0.2. *We say H is a **subgroup** of $(G, \cdot)$ if $H \subseteq G$ and H is a group with respect to $\cdot$. H is a cyclic subgroup if there exists $a \in G$ such that $H = \{a^n | n \in \mathbb{Z}\}$. G itself is cyclic if it is equal to a cyclic subgroup (i.e. there is an element which **generates** the entire group).*

For any group G, we denote the number of elements in G as $|G|$. If G is a cyclic group such that a generates G, then $a^n = e$ for some integer n, since it must also be able to generate the identity element.

Fact 2.0.1. *If a generates G and n is the smallest positive integer such that $a^n = e$, then $|G| = n$*

Lastly, we will need the well-known Langrange Theorem. For the sake of completeness, we will briefly define left cosets.

Definition 2.0.3. *Let H be a subgroup of G. Given $a \in G$, $aH = \{x \in G | x = ah, h \in H\}$ is a left coset of H in G.*

Certainly some elements will result in the same coset. For example, if $a \in G$ and $a \in H$ then aH = H. The number of distinct left cosets we call the index of H in G and denote it by $[G : H]$

Theorem 2.1 (Lagrange's Theorem)**.** *If G is a finite group and H a subgroup of G, then*

$$|G| = |G| \cdot [G : H]$$

We care about the following consequences of this theorem:

Corollary 2.1.1. *The order of any subgroup H of G must always divide the order of G. What's more, $[G : H]$ is also a positive integer.*

We will also have need of one more elementary fact:

Fact 2.1.1. *For any $a \in G$, if $|G| = N$, then $a^N = e$*

We will soon talk about groups over fields. Fields are sets with two binary operations, an addition and multiplication, which "play nice" together. In essence, fields are abelian groups with respect to addition, have a commutative multiplication, have an identity element for the multiplication, have inverses for all nonzero elements, and satisfy simple distributive laws. The set of real numbers $\mathbb{R}$ is an infinite field, but there exist finite fields. In fact, all finite fields have order p^e, where p is a prime and e an integer greater than zero, so we may unambiguously write $\mathbb{F}_{p^e}$ to denote them. When we talk of a group over a field, it means the variables of that group accept values only from that field.

3 Construction of $E(\mathbb{F}_q)$

Let us talk about elliptic curves on a more general level before we go through with the construction regarding $\mathbb{F}_q$.

3.1 Elliptic Curves in the $\mathbb{R}^2$ setting

Definition 3.0.1. *An elliptic curve E in $\mathbb{R}^2$ represents an equation of the Weierstrass form:*

$$y^2 = x^3 + ax + b,\ a, b \in \mathbb{R}$$

One might expect this to hold for all real-valued pairs $(x, y) \in \mathbb{R}^2$. But there are some a and b values where this equation is degenerative due to a singularity. Specifically, we require that $4a^3 + 27b^2 \neq 0$ to avoid cusps and singularities which will not work in our construction. Now let's add infinity $0 = \infty$ for the y-axis of R^2; this is so we take into account that vertical lines have slope ∞.

Goal: To express an arbitrary elliptic curve in this space as a group. **Why?** If we know two (not necessarily unique) points on an elliptic curve, we can find another. In other words, take the point or two picked out and you can always align them in a line that connects to another point in E. All points mentioned are aligned so this is a binary operation. By assuring that the curve forms a group, the properties from Definition 2.0.1 give a structure to this process. Let us define the group for clarity.

Theorem 3.1. *Denote this process of 'finding' an aligned point R in E as addition. Then with points $P, Q, R \in E$:*

- ***Identity:*** *There exists $0 \in E$ such that for all $P \in G$, $P + 0 = 0 + P = P$*
- ***Inverses:*** *for every $P \in E$ there exists $-P \in E$ such that P + (-P) = 0 and -P is symmetric in the x-axis*
- ***Commutative:*** *For all $P, Q \in E$, $P + Q = Q + P$*
- ***Associative:*** *$(P + Q) + R = P + (Q + R)$ for all $P, Q, R \in E$*
- ***Closed:*** *If $P, Q \in E$, then $P + Q = -R \in E$. This is so that $P + Q + R = 0$.*

So E must be an abelian group.

As the details will not be proved here, what you should get out of this is that the operation is designed around achieving $P + Q + R = 0$ (which signifies that P, Q, and R are aligned in a line). The existence of an inverse takes advantage of the fact that points on an elliptic curve are symmetric to each other over the x-axis. So always $P + Q = -R$, and then $-R + R = 0$ must be true as the line connecting them is perpendicular to the x-axis and thus vertical. Associativity and commutativity imply that we can interpret the last sentence similarly regardless of what point among P, Q, R we find from addition.

Fact 3.1.1. *$P = Q$ (one point case), then you can form a **tangent** line that is aligned with some $R \in E$*

Let us show how we can perform group addition using algebra. With $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ the following two methods only differ by how you calculate slope s.

1. **Cord Method:** We use this when $P \neq Q$ where we can use point-slope form to find s. So $s = \frac{y_2 - y_1}{x_2 - x_1}$ and we can calculate $P + Q = (x_3, y_3) = (s^2 - x_1 - x_2, s(x_1 - x_3) - y_1)$.
2. **Tangent Method:** We use this when $P = Q \neq 0$ (tangency point) with slope equation $s = \frac{3x_1^2 + a}{2y_1}$. So we can calculate $P + Q = (x_3, y_3) = (s^2 - 2_x 1, s(x_1 - x_3) - y_1)$.

Note that the $P = -Q$ case trivially results in $P + Q = 0$.

3.2 Elliptic Curves over $\mathbb{F}_q$

Recall that $\mathbb{F}_q$ is a finite field of order $q = p^e$ for prime p and natural number e. When $e = 1$, we are dealing with a prime field where one could always imagine the elements as integers $\{0, 1, ..., p-1\}$. In fact we have already used $\mathbb{Z}_p$ as a prime field. When $e > 1$, we are dealing with an extension field. Take for example $\mathbb{F}_{2^2}$: thinking of the elements in a similar way as before, you may notice that 2 is a nonzero element yet $2 * 2 = 0 \neq 1$ (it has no multiplicative inverse). Extension fields work differently and can get difficult, so for simplicity let us work with $\mathbb{F}_p$.

Fact 3.1.2. *In a prime field $\mathbb{F}_p$ with point $(x, y) \in \mathbb{F}_p^2$ and $a, b \in \mathbb{F}_p$, an elliptic curve represents an equation of the form:*

$$y^2 \cong x^3 + ax + b(modp), 4a^3 + 27b^2 \ncong 0(modp)$$

CONSEQUENCE: The group addition algebraic methods hold with modulo p, and thus we are able to have **Theorem 3.1** hold when E is over $\mathbb{F}_p$.
Since we are dealing with a finite field, we would of course expect the subset of points that make up $E(\mathbb{F}_p)$ to be finite. However the order for the group is difficult to determine.

Fact 3.1.3. ***Schoof's Algorithm** is the first deterministic polynomial-time algorithm that could calculate the order of group $E(\mathbb{F}_p)$. Its runtime has a complexity that has been able to be reduced down to $O(log^5 q)$.*

Subgroups: Some interesting properties emerge when we consider the subgroup H that a point P generates in the curve group $E(\mathbb{F}_p)$. Note that $|E(\mathbb{F}_p)| = N$ is not necessarily prime. One result that can be made is if we find a large enough prime n so that $nP = n * P = 0$, then H is of **prime** order and also **cyclic**. Thus H can be represented as $\{0, P, 2P, ..., (n-1)P\}$. Now recall Lagrange's Theorem from Section 2. So for an order n subgroup of our order N group, it must hold that n divides N. In particular, $N/n = h$ which is an integer representing the index of H in $E(\mathbb{F}_p)$. Some consequences are that $NP = 0$ and having prime subgroup order n implies $(nh)P = n(hP) = 0$. This information is all useful because it will help us

obtain a random first point of the curve!!

Theorem 3.2. ***Base Point Algorithm:*** *Assume* $|E(\mathbb{F}_p)| = N$, $|H| = n$ *with* n *prime, and index* h.

1. *Choose a random* $P \in E(\mathbb{F}_p)$
2. *Compute* hP
3. *If* $hP = 0$ *go back to step 1*
4. *Because* $(nh)P = n(hP) = 0$ *with prime* n*, and* $hp = 0$*, it must hold that* H *is a cyclic subgroup of order* n *that is generated by* hP
5. *Set the base point* $G = hP$.

Now we are ready to utilize elliptic curves.

4 Applications to Cryptography

Let us set the stage for using Elliptic Key Cryptography.

- **Discrete Log Problem:** Say a point P in the elliptic curve generates a prime order (cyclic) subgroup. Then we can pick a private key $\alpha \in \{0, P, 2P, ..., (n-1)P\}$ so that public key $Q = \alpha P$. The Discrete Log Problem is figuring out α given P and Q, and for security we assume it is hard.
- **Construction:** The parties involved agree upon a tuple of domain parameters (p, a, b, G, n, h). Let p denote the size of the field, a and b elliptic curve parameters, G the base point of the cyclic subgroup $E(\mathbb{F}_p)$ so that $nG = 0$, n is a sufficiently large prime, and h is the index of such subgroup.

4.1 ECDH

Recall that the Diffie-Hellman key-agreement (**lecture 8**) relies on the Discrete Log Problem being difficult. The expected result is that Alice and Bob share a key which an adversary cannot learn anything significant about over a public channel.

1. Alice and Bob agree upon (p, a, b, G, n, h), perhaps using randomness to generate a and b.
2. Alice picks $x \leftarrow \{0, 1, ..., n-1\}$ and Bob picks $y \leftarrow \{0, 1, ..., n-1\}$ at random
3. Alice sends $Q_a = xG$ to Bob and Bob sends $Q_b = yG$ to Alice
4. Alice takes $xQ_b = xyG = k$ and Bob takes $yQ_a = yxG = k$ as the key

Now Alice and Bob share key k. Note that unlike typical DH, here we have multiplicative notation instead of exponential notation. Thus k is easier for Alice and Bob to individually compute.

4.2 ECDSA

Suppose Alice wants to authenticate a message m to Bob. The following protocol uses a "Hash and Sign" methodology, after the parties have agreed upon (p, a, b, G, n, h) and a secure hash algorithm. Alice choose a random $sk \leftarrow \{1, 2, ..., p-1\}$ for a signing key and publishes $vk = sk \cdot G$ as the verification key. To sign the message, Alice does the following:

1. Calculate Hash(m), and letting z be the first n-bits of the result
2. Choose $k \leftarrow \{1, 2, ..., p-1\}$ randomly
3. Calculate Q = (x_1, y_1) = kG
4. Calculate r = x_1 (mod n). If r = 0, go back to step 2.
5. Calculate s = $k^{-1}(z + r \cdot sk)$. If s = 0, go back to step 2.

(r,s) becomes the signature. Bob, running the verification algorithm, would do the following:

1. Calculate z by taking the first n bits of Hash(m)
2. Calculate $u_1 = s^{-1}z$ (mod n)
3. Calculate $u_2 = s^{-1}s$ (mod n)
4. Calculate $(x_1, y_1) = u_1G + u_2(vk)$
5. Accept if $r \cong x_1$ (mod n), otherwise reject

Proving the correctness of the above scheme requires several computational substitutions, but the main idea is that both sides end up calculating the point Q and Alice proves she calculated this point by signing it in step 5. Built into this step is part of the hash of the message, and as Bob essentially computes the reverse of this last step he will only accept if both the hash is correct (which he recomputes himself) *and* his verification key corresponds to Alice's signing key.

5 Optimizations

Adding two points on an elliptic curve in Weierstrass form takes 2 multiplications, 1 squaring, and 1 inversion. All these operations are fast, but still an inversion takes anywhere between 9 to 40 times as long as multiplication, and a squaring takes about 0.8 multiplications. When resources are limited, such as on Internet of Things devices, the impact can be felt. Luckily,

this form is not the only way to represent elliptic curves, and other representations can lead to faster implementations.
A curve is in **Montgomery Form** if it satisfies

$$by^2 = x^3 + ax^2 + x, b(a^2 - 4) \neq 0$$

Addition in this form requires 3 multiplications and 2 squarings. It is estimated that computing curves in this form is faster than Weierstrass curves by about 10%. A curve is in **Edwards form** if

$$x^2 + y^2 = 1 + dx^2y^2, d \neq 0, 1$$

Addition here takes 10 multiplications and 1 squaring. Addition in this form for P = (x_1, y_1) and Q = (x_2, y_2) takes on a particularly succinct definition:

$$P + Q := \left(\frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2}\right)$$

Because of this simplicity, curves in this form can be particularly resilient against *timing attacks*, where an adversary tries to gain information on a private key by timing how long a computation with that private key might take.

Mersenne primes, of the form $2^n - 1$ can also be used for faster implementations when chosen as the field size. This is because such primes can be broken up into factors of two which lend themselves to bit-wise operations like shifts.

6 Issues with Elliptic Curve Cryptography

There are certain setbacks we must deal with if we want to use Elliptic Curve Cryptography. The most obvious one is that because there is so much variety in how these kinds of curves can appear, some just make it easier for the Discrete Log Problem to be solved. We can refer to them as weak elliptic curves.
EXAMPLES:

- $|E(\mathbb{F}_p)| = q$ where q is a natural number power of p
- $|E(\mathbb{F}_p)|$ divides $p^r - 1$, where r is small
- Assuming $|F| = q$ again, curves over $\mathbb{F}_{2^m}$ where m is composite

The U.S National Institute of Standards and Technology (NIST) has an approved list of curves, but there are also methods for generating safe random curves. This randomization can be proven using hashes, so the other party knows that the curve **was not specially tailored to be crackable**. We know that NIST has experienced vulnerabilities in the recent years. In 2013 a leaked memo from Edward Snowden suggested at least one backdoor in an elliptic-curve pseduorandom generator from NIST (withdrawn in 2014). Thus it might be possible that the people working there standardized a set of weak elliptical curves.

Another very recent problem for Elliptic Curve Cryptography is **Shor's algorithm**. This is because Shor's is a quantum algorithm and can solve computationally hard problems like the Discrete Log Problem in polynomial time. So using Elliptic Curves in the way we defined will not be secure post-Quantum.

References

[1] Boneh, Dan. Shoup, Victor. *A Graduate Course in Applied Cryptography*. Version 0.4, 2017.

[2] Corbellini, Andrea. *Elliptic Curve Cryptography: a gentle introduction*. 2015. `http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/#elliptic-curves`

[3] Friedl, Stefan. *An Elementary Proof of the Group Law for Elliptic Curves*. `http://math.rice.edu/~friedl/papers/AAELLIPTIC.PDF`

[4] Okeya, Katsuyuki, et. al. *Elliptic Curves with the Montgomery-Form and Their Cryptographic Applications*. 2000. `https://link.springer.com/content/pdf/10.1007/978-3-540-46588-1_17.pdf`

[5] Olenski, Julie *ECC 101: What is ECC and why would I want to use it?*. 2015. `https://www.globalsign.com/en/blog/elliptic-curve-cryptography/`

[6] Washington, Lawrence C. *Elliptic Curves: Number Theory and Cryptography*. Second Edition. Taylor & Francis Group, Boca Raton, Florida, 2008.

[7] Wikipedia. *Elliptic-curve cryptography*. `https://en.wikipedia.org/wiki/Elliptic-curve_cryptography`

[8] Wolfram. *Finite Field* `http://mathworld.wolfram.com/FiniteField.html`