An introduction to pairing-based elliptic curve cryptography

David Kurniadi Angdinata

Friday, 13 September 2019

Adjoint

1 Introduction

Cryptography is the study of computational techniques to allow for secure communication of information across public platforms in the presence of third party eavesdroppers. Historically, this is based on the unidirectional computational intractability of classical mathematical problems, such as *integer factorisation* and *discrete logarithms*. For example, while the multiplication of two large integers are relatively easy on a modern computer, decomposing a huge integer into its constituents is significantly more difficult. This complexity problem is the basis of conventional *Rivest–Shamir–Adleman (RSA)* encryption schemes.

However, as computers become increasingly powerful, many of these conventional schemes become less secure by being more vulnerable from external attacks. As a result, many cryptosystems require huge key sizes, which in turn enforce huge storage and transmission requirements. The advent of *elliptic curve cryptography (ECC)* resolves this predicament by allowing for significantly reduced key sizes, while maintaining an equivalent measure of security, albeit with slightly more theory required for its implementations. For instance, it is estimated that a key size of 4096 bits in RSA is equivalent to 300 bits in ECC.

While there are certainly a myriad of powerful cryptosystems in ECC, a relatively modern approach known as *pairing-based cryptography (PBC)* forms the backbone of several important cryptographic protocols, some of which allow for multi-party key agreements. The mathematical theory of PBC naturally extends from that of conventional ECC, but relies on several hardness assumptions that differ from the classical discrete logarithm problem used in ECC, which is older and more understood by cryptanalysts. As such, PBC has gained wide attention in cryptology for its versatility and security.

These notes will serve as a brief introduction into the mathematical ideas involved in the engineering of ECC, in the flavour of PBC, as well as some of the difficulties encountered and optimisations considered in the implementation process. Unless otherwise specified, all ideas have been developed under the powerful type system of Haskell across several polymorphic libraries.

The general pairing-based ECC stack is built upon several layers of mathematical abstraction. In ascending order of complexity, these layers can be summarised as follows.

- Galois field arithmetic. In this layer, the basic operations in arithmetic are defined for a particular finite system of numbers that have primality-related properties.
- Elliptic curve operations. In this layer, the definition of an *elliptic curve* is given in terms of a *Galois field*, and several operations acting on its points are recorded in a series of compact formulae.
- Bilinear pairing algorithms. In this layer, curve points are paired up with each other in accordance to specialised algorithms, producing another point on the curve satisfying certain properties.

These layers will be described in further detail over the following sections. Note that, at the lowest layer, Galois field arithmetic would be compiled into machine code, where further low-level optimisations can be made. In contrast, at the highest layer, pairings are merely primitives in various cryptographic protocols, which will in turn be used in concrete applications. These will not be touched upon in the upcoming sections.

2 Galois field arithmetic

Before defining a Galois field, the definition of an abelian group will make things much simpler.

Definition. *An **group** is a set G with an operation $+ : G \times G \to G$ such that*

- *(identity) there exists a unique 0 in G such that for all a in G, we have $0 + a = a + 0 = a$,*
- *(inverses) for all a in G, there exists a unique $-a$ in G such that $a + (-a) = (-a) + a = 0$, and*
- *(associativity) for all a, b, and c in G, we have $(a + b) + c = a + (b + c)$.*

*Moreover, G is **abelian** if for all a and b in G, we have $a + b = b + a$ (commutativity).*

For example, the set of integers with the usual addition is an abelian group, but the set of natural numbers with the usual addition is not an abelian group since there are never any inverses that are positive. With this in mind, the definition of a Galois field can be given succinctly.

Definition. *A **field** is a set K with operations $+ : G \times G \to G$ and $\cdot : G \times G \to G$ such that*

- *K with $+$ is an abelian group,*
- *K minus 0 with $\cdot$ is an abelian group, and*
- *(distributivity) for all a, b, and c in K, we have $a \cdot (b + c) = a \cdot b + a \cdot c$.*

*Moreover, K is **Galois** if K is finite.*

For example, the set of real numbers with the usual addition and multiplication is a field, albeit not Galois. An important result in *Galois theory* characterises all possible Galois fields.

Theorem. *A Galois field is defined inductively as follows.*

- *For any prime number p, there exists a unique **prime field** $\mathbb{F}_p$, a Galois field with p elements. Its elements are typically represented by the integers from 0 to $p - 1$, and its $+$ and $\cdot$ are addition and multiplication of integers modulo p respectively. For example, $\mathbb{F}_2$ is a prime field containing the integers 0 and 1, and its operations are tabulated as follows.*

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

- *For any prime field $\mathbb{F}_p$ and any natural number n, there exists an **extension field** $\mathbb{F}_{p^n}$, a Galois field with p^n integer elements, defined over a polynomial $f(X)$ and extended from $\mathbb{F}_p$. Its elements are typically represented as polynomials of degree less than n with coefficients in $\mathbb{F}_p$, and its $+$ and $\cdot$ are addition and multiplication of polynomials modulo $f(X)$. For example, $\mathbb{F}_4$ defined over the polynomial $X^2 + X + 1$ is an extension field extended from $\mathbb{F}_2$ containing the integers 0 and 1 as well as the polynomials X and $X + 1$, and its operations are tabulated as follows.*

$+$	0	1	X	$X+1$
0	0	1	X	$X+1$
1	1	0	$X+1$	X
X	X	$X+1$	0	1
$X+1$	$X+1$	X	1	0

$\cdot$	0	1	X	$X+1$
0	0	0	0	0
1	0	1	X	$X+1$
X	0	X	$X+1$	1
$X+1$	0	$X+1$	1	X

While subtraction and division are not explicitly tabulated here, they can be inferred indirectly from addition and multiplication, albeit not so easily for division. In particular, computing divisions in extension fields involve the *extended Euclidean algorithm*, which makes it an order of magnitude slower than the other operations, and as such are generally avoided in ECC whenever possible.

On the other hand, there are certain slow functions on Galois fields that are unavoidable in PBC that are called sparingly. These include computing *modular square roots* with the *Tonelli–Shanks algorithm* and checking if a field element is a *primitive root of unity*, both of which takes some effort in its implementation.

As all layers in ECC depend on Galois field arithmetic, they have to be optimised for maximal efficiency. In particular, most Galois fields used in ECC are prime fields, which are made incredibly fast by low-level optimisations, or extension fields of the form $\mathbb{F}_{2^m}$, which can be optimised considerably with bit operations.

3 Elliptic curve operations

Elliptic curves are the main objects considered in ECC, and a formal definition depends on that of a general field, and involves several concepts in the mathematical field of *algebraic geometry*.

Definition. *A **projective plane curve** of degree n over a field K is the set of triples of solutions to a polynomial equation in three variables such that each monomial has a total degree of n. A projective plane curve is **smooth** if its partial derivatives with respect to its three variables are not all zero. An **elliptic curve** over a field K is then a smooth projective plane curve of degree three with a specified base point.*

For example, projective plane curves of degree two over $\mathbb{F}_2$ are exactly the sets of the form

$$\{(X, Y, Z) \in K : aX^2 + bY^2 + cZ^2 + dXY + eXZ + fYZ = 0\},$$

where the coefficients a to f are either 0 or 1, and these are smooth whenever at least one of

$$2aX + dY + eZ, \qquad 2bY + dX + fZ, \qquad 2cZ + eX + fY,$$

their partial derivatives with respect to X, Y, and Z, is not zero. In the degree three case, there is powerful result in algebraic geometry known as the *Riemann–Roch theorem* that characterises all possible elliptic curves over any field where $1 + 1 \neq 0$ and $1 + 1 + 1 \neq 0$. From now on, all fields will be implicitly assumed to satisfy these conditions to simplify explanations, but the general theory will still hold regardless.

Theorem. *An elliptic curve over a field K is a set of the form*

$$\{(x, y) \in K^2 : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\},$$

where the coefficients a and b are in K and satisfy $4a^3 + 27b^2 \neq 0$, and $\mathcal{O}$ is a fixed specified symbol.

Here, the number of variables has been decreased by one due to a reversible process known as *dehomogenisation*, giving a smooth **affine plane curve** of two variables that can be plotted in a Euclidean plane, as well as a newly introduced **point at infinity** $\mathcal{O}$ capturing the remaining loss of information.

Theorem. *An elliptic curve over any field is an abelian group as follows.*

- *The identity point is the point at infinity $\mathcal{O}$.*
- *The inverse point of a point is obtained by reflecting the point about the x-axis.*
- *The addition of two points is obtained by inverting the third point of intersection between the curve and the line joining the two points.*

It turns out that this definition of an abelian group is indeed well-defined, but in certain edge cases, the inverse of a point or the addition of two points may not exist in the plane, and in these cases they will be defined as the point at infinity $\mathcal{O}$. It will be remarked that the group axiom of associativity is by far the most difficult to verify, and several general geometric techniques can been developed just to tackle this, but this is indeed true and can be implicitly assumed in the uses in ECC.

Given the definition of the group law above, it is possible to give explicit formulae for the various operations using the *chord and tangent* rule, possibly having many cases to consider in each operation. For example, addition of points in the affine part of the elliptic curve depend on whether the points are the same, in which case the tangent at that point will be drawn for *point doubling*, or otherwise, in which case an obvious line can be drawn between them. In any case, these explicit formulae are always the same and depend on the coefficients a and b, and as such are usually recorded in online ECC databases.

With all of the basic group operations being defined, a slightly higher level operation known as *scalar multiplication* can also be defined in terms of these, which involves taking an affine point (x, y) and efficiently computing $n(x, y) = (x, y) + \cdots + (x, y)$ with *exponentiation by squaring*. Instead of performing group operations on all of the points of the curve, a subset is usually chosen carefully such that any point is attainable from any other point from scalar multiplication with itself sufficiently many times, and this subset will again have a prime number of points so that the arithmetic of a prime field can be reused. Scalar multiplication is used very often in ECC, and it is the basis for the intractability of the elliptic curve discrete logarithm problem, namely the problem of computing the integer n given the affine points (x, y) and $n(x, y)$.

In ECC, elliptic curves are mostly defined over the Galois fields $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$, since cryptographic operations tend to be finitary. In many use cases, the elliptic curve itself is usually fixed and publicly available as a set of *recommended domain parameters*, possibly with a popular name such as *JubJub*. These parameters are designed carefully, often with *nothing-up-my-sleeve numbers*, such that for instance, fewer multiplications are needed in the explicit formulae and the size of the special subset is a prime number.

There are several other considerations when implementing an efficient ECC library. As aforementioned, while division is an expensive Galois field operation, it is used several times in every addition of affine points. This could be circumvented by reconsidering projective points that omits the point at infinity completely, and would have slightly different explicit formulae that do not involve division but with a markedly increase number of multiplications. There will hence be a trade-off between the higher number of multiplications and the higher cost of divisions, and this can only be determined with benchmarking experiments.

Finally, an important consideration would be the usability and extensibility of an ECC library in larger contexts. As there is a wide range of protocols that use different elliptic curves, it would be beneficial for a library to include as many known curves as possible, and allow for a flexible polymorphism between them. For instance, in a battle to resist timing attacks while maintaining optimal efficiency, the recent development of *Edwards curves*, which have very different *complete addition formulae* that do not distinguish between the different addition cases, surged in popularity in contrast to the conventional *Weierstrass curves* given above. These exotic curves should ideally be present in any readily extensible elliptic curve library.

4 Bilinear pairing algorithms

Bilinear pairings are the main functions considered in PBC and pairing-based ECC, and a formal definition is rather simple. Let G_1 and G_2 be two abelian groups of size q, and let G_T be an abelian group of size q dependent on G_1 and G_2. The latter group is conventionally written multiplicatively, so that its identity is 1 and scalar multiplication becomes scalar exponentiation, but this will not affect the overall computation.

Definition. *A **pairing** of G_1 and G_2 is a map $\langle -, - \rangle : G_1 \times G_2 \to G_T$ such that*

- *(bilinearity) for all P in G_1 and Q in G_2, and for all non-zero a and b in $\mathbb{F}_q$ we have $\langle aP, bQ \rangle = \langle P, Q \rangle^{ab}$,*
- *(non-degeneracy) there exist P in G_1 and Q in G_2 such that $\langle P, Q \rangle \neq 1$, and*
- *(computability) there exists an efficient algorithm to compute $\langle -, - \rangle$.*

These abelian groups may be Galois fields under multiplication or elliptic curve point groups under addition. An efficient pairing defined for a particular family of elliptic curves will be the main example.

Definition. *A **Barreto–Naehrig curve** is an elliptic curve over a field $\mathbb{F}_{p^{12}}$ of the form*

$$E(\mathbb{F}_{p^{12}}) = \{(x, y) \in \mathbb{F}_{p^{12}}^2 : y^2 = x^3 + b\} \cup \{\mathcal{O}\},$$

where the coefficient b is in $\mathbb{F}_p$, and the size of the group $E(\mathbb{F}_p)$ is a prime number q defined by

$$p = 36t^4 + 36t^3 + 24t^2 + 6t + 1, \qquad q = 36t^4 + 36t^3 + 18t^2 + 6t + 1,$$

*where the **Barreto–Naehrig parameter** t is a specified integer.*

These elliptic curves are *pairing-friendly*, which means that a suitably nice pairing algorithm can be defined over its *subgroups*, subsets that are groups themselves. In particular, the subgroup

$$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p^2 : y^2 = x^3 + b\} \cup \{\mathcal{O}\}$$

will play the role of G_1, and a size q subgroup of the *twisted* elliptic curve point group

$$E'(\mathbb{F}_{p^2}) = \{(x, y) \in \mathbb{F}_p^2 : y^2 = x^3 + b/\xi\} \cup \{\mathcal{O}\}$$

will play the role of G_2, where the coefficient ξ is in $\mathbb{F}_p^2$. The target group G_T will then be inhabited by a size q multiplicative subgroup of $\mathbb{F}_{p^{12}}$, of which each element in the group is a q-th root of 1.

Definition. *The **optimal ate pairing** over a Barreto–Naehrig curve $E(\mathbb{F}_{p^{12}})$ is a map*

$$\langle -, - \rangle : E(\mathbb{F}_p) \times E'(\mathbb{F}_{p^2}) \to \mathbb{F}_{p^{12}}$$

defined by the following algorithm.

```
Input: P in E(Fp) and Q in E(Fp2)
Output: <P, Q> in Fp12
 1. Write s = 6t + 2 as a sum of signed binary powers {s_i}_i for i in [0 .. l - 1]
 2. (T, f) <- (Q, 1)
 3. for i in [l - 2 .. 0] do
 4.   (T, f) <- (T + T, f^2 * Line(T, T, P))
 5.   if s_i = -1 then
 6.     (T, f) <- (T - Q, f * Line(T, -Q, P))
 7.   else if s_i = 1 then
 8.     (T, f) <- (T + Q, f * Line(T, Q, P))
 9.   end if
10. end for
11. Q1 <- Frob(Q), Q2 <- Frob(Q1)
12. (T, f) <- (T + Q1, f * Line(T, Q1, P))
13. f <- f * Line(T, -Q2, P)
14. f <- f^((p^12 - 1) / q)
15. return f
```

`Line(T, Q, P)` is the equation of the line joining T and Q, evaluated at P and embedded into $\mathbb{F}_{p^{12}}$ via

$$\begin{aligned} E'(\mathbb{F}_{p^2}) &\longrightarrow E(\mathbb{F}_{p^{12}}) \\ (x, y) &\longmapsto (x\xi^{1/2}, y\xi^{1/3}), \end{aligned}$$

and `Frob(Q)` is the Frobenius endomorphism

$$\begin{aligned} E'(\mathbb{F}_{p^2}) &\longrightarrow E'(\mathbb{F}_{p^2}) \\ (x, y) &\longmapsto (x^p, y^p). \end{aligned}$$

The optimal ate pairing is a variant of the more well-known *Tate pairing* used in general *number theory* but is optimised for efficiency with a shorter main loop. While the algorithm here is explicit, its proof of correctness using the notion of *divisors* can be found in the literature and will be omitted here for brevity.

Theorem. *The optimal ate pairing over a Barreto–Naehrig curve is computable non-degenerate bilinear.*

Several obvious optimisations can be made to the algorithm itself, such as skipping a potentially expensive line function computation during the assignment of $T + Q$ and $l(T, Q, P)$ by storing certain values. More importantly, there are heavy field-specific optimisations for computing Frobenius endomorphisms and the final exponentiation step, both of which take up a majority of the computation time in PBC.

As for ECC protocols, different PBC protocols may use differing pairing-friendly curve families, and as such it is desirable to have some form of polymorphism between them. For instance, several *zkSNARK* actively make use of the *Barreto–Lynn–Scott* curve family, which is markedly different in structure to the Barreto–Naehrig curve family but with an equivalently defined optimal ate pairing algorithm.

5 Conclusion

While it is in the best interest of theoretical mathematics to abstract ideas into general features, the trade-off with actual performance in cryptographic protocols always needs to be considered. Fortunately, the type system of Haskell is sufficiently powerful to achieve the best of both worlds, allowing for a clean polymorphic abstraction while maintaining relatively good performance.