Lecture notes:
Elliptic curve pairings in cryptography

Lorenz Panny

TU/e, 2DMI10 'Applied Cryptography'

November 29, 2018

1 What?

In the previous lecture[1] I claimed:

> ***Fact.*** *Well-chosen elliptic curves are as close to generic groups as it gets.*

In this lecture, I will convince you of the opposite:[2]

Fact. Well-chosen elliptic curves are *very far* from generic groups.

> *Was interessiert mich mein Geschwätz von gestern?*[3]
>
> — attributed to the first German 'Bundeskanzler' Konrad Adenauer, although he probably never said that

2 Pairings

Throughout this section, let $(G_1, +)$, $(G_2, +)$, and $(T, \cdot)$ be abelian groups.
(Note that T is written multiplicatively because it will be a subgroup of $(\overline{\mathbb{F}}_q^*, \cdot)$ in the constructions we shall see later.)

Definition 1. A map

$$e\colon G_1 \times G_2 \to T$$

is *bilinear* if for all $P, P' \in G_1$ and $Q, Q' \in G_2$

$$e(P + P', Q) = e(P, Q) \cdot e(P', Q);$$
$$e(P, Q + Q') = e(P, Q) \cdot e(P, Q').$$

Definition 2. A map

$$e\colon G_1 \times G_2 \to T$$

is *non-degenerate* if for all $P \in G_1 \setminus \{\mathcal{O}\}$, there exists some $Q \in G_2$ with $e(P, Q) \neq 1$.

Definition 3. A *pairing* is a bilinear and non-degenerate map

$$\hat{e}\colon\; G_1 \times G_2 \longrightarrow T.$$

We are usually interested in the case where $(G_1, +)$, $(G_2, +)$, and $(T, \cdot)$ have prime order ℓ.

[1]See `https://yx7.cc/docs/lectures/ecc_20181127.pdf` for the notes.
[2]Of course, the deception is that 'well-chosen' has different meanings in these two statements.
[3]'What do I care about my blabber from yesterday?'

Remark. When the groups have prime order, the condition of non-degeneracy is equivalent to $\hat{e} \neq 1$, i.e., it is enough to find *one* pair $(P, Q) \in G_1 \times G_2$ which does not map to the identity under the pairing $\hat{e}$. (**Exercise** if you're bored: show this.)

Lemma 1. For a pairing $\hat{e}$ as above, all $(P, Q) \in G_1 \times G_2$, and all integers $a, b \in \mathbb{Z}$,

$$\hat{e}(aP, bQ) = e(P, Q)^{ab}.$$

Proof. First take care of the special cases $a = 0$ and $b = 0$, then reduce to positive a and b, then do induction on a and b. □

2.1 Immediate consequences

In this section, we assume $G_1 = G_2$ and hence simply write G for these groups. The pairing $\hat{e}$ thus has the form

$$\hat{e} \colon G \times G \to T.$$

We shall now discuss a few immediate consequences of such a pairing.

2.1.1 Transfer attacks

Let G be a cyclic group of prime order ℓ with an efficient pairing $\hat{e} \colon G \times G \to T$ and P a generator of G. Suppose an attacker faces an instance of the DLP in G with respect to P, hence is given $Q = [x]P \in G$ and wants to find x. Using $\hat{e}$, he can proceed as follows:

1. Compute $g = \hat{e}(P, P)$ and $h = \hat{e}(P, Q)$.
   Due to bilinearity, we have $h = \hat{e}(P, Q) = \hat{e}(P, [x]P) = \hat{e}(P, P)^x = g^x$.
2. Solve DLP in T to recover x.

This approach is sometimes much more efficient than trying to solve the DLP in G directly (e.g., for supersingular elliptic curves, which we shall see later).

2.1.2 The *decision* Diffie–Hellman problem

The security of many cryptographic systems (including Diffie–Hellman key exchange) relies on the hardness of the following problem:

Reminder. The *decision Diffie–Hellman Problem* (DDH) in a cyclic group G with generator P is, given $[x]P$, $[y]P$, and $Q \in G$, to decide whether $Q = [xy]P$.

Lemma 2. If G admits an efficiently computable pairing, then DDH is easy in G.

Proof. Test whether $\hat{e}([x]P, [y]P) = \hat{e}(P, Q)$. □

Groups in which the *computational* Diffie–Hellman problem (CDH) is hard while DDH is easy are known as '*Gap Diffie–Hellman groups*'.

2.1.3 Signatures from pairings

Let G be a cyclic group of prime order ℓ and $P \in G$ a generator.

Identification from Diffie–Hellman. If Alice has a key pair $(x, [x]P)$, she can prove to anyone that she indeed knows x, as follows.

Bob picks a random $y \in \{0, ..., \ell - 1\}$, computes $S := [y][x]P$, sends $[y]P$ to Alice, and waits. Alice computes $R := [x][y]P$ and sends that to Bob. Bob verifies that $R = S$. This clearly works and the security relies on the hardness of CDH.

To turn this into a signature scheme, we need to get rid of Bob and replace him by a message m. A promising idea is to replace $[y]P$ by a hash of m in G. However, of course the signatures should be publicly verifiable, but *nobody knows* y.[4]

This is where pairings enter the picture, enabling everyone to verify that Alice's response was correct without actually learning y.[4]

BLS signatures [Boneh–Lynn–Shacham '04]**.** We now assume that we have an efficiently computable pairing $\hat{e}\colon G \times G \to T$, and that there exists a cryptographically secure hash function $H\colon \{0,1\}^* \to G$. (There are G for which we can't do this!)
We can then make a pairing-based signature scheme as follows:

- Key generation:
  Alice's private key is a random $x \in \{0, ..., \ell - 1\}$. Her public key is $Q = [x]P$.
- Signing:
  Alice hashes her message m and computes her signature $\sigma = [x]H(m)$.
- Verification:
  Bob verifies that $\hat{e}(\sigma, P) = \hat{e}(Q, H(m))$.

Correctness.

$$\hat{e}(P, \sigma) = \hat{e}(P, [x]H(m)) = \hat{e}(P, H(m))^x = \hat{e}([x]P, H(m)) = \hat{e}(Q, H(m)). \qquad \square$$

Remark. Notice that the reason this works is really only an application of the 'gap' property discussed in the previous section. It does not make any use of the group structure of the target group T at all!

3 Pairings from elliptic curves

From the very beginnings of pairing-based cryptography up until now, elliptic curves are essentially the only source of practical pairing constructions for cryptographic applications.

3.1 Preliminaries

We first need to discuss some more properties of elliptic curves.

3.1.1 Rationality

Consider an elliptic curve E over a field K, for example an Edwards curve

$$E_d\colon\ x^2 + y^2 = 1 + dx^2y^2\,.$$

Just like points in K, we can consider points over any finite extension $L \supseteq K$, or even over the algebraic closure $\overline{K}$: For any extension field L of K, an *L-point* or *L-rational point* or *point defined over* L on E is a tuple $(x, y) \in L^2$ that satisfies the curve equation. The set of

[4]No pun intended.

L-rational points is a group.[5] In the following, we will simply write E for the set $E(\overline{K})$ of points over the algebraic closure.

Reminder. For a finite field $\mathbb{F}_q$, there exists a unique (up to isomorphism) extension field $\mathbb{F}_{q^k}$ for every integer $k \geq 1$. It can be represented as $\mathbb{F}_q[Y]/r(Y)$ for an irreducible polynomial $r \in \mathbb{F}_q[Y]$ of degree k: elements of $\mathbb{F}_{q^k}$ are of the form $c_0 + c_1T + c_2T^2 + \cdots + c_{k-1}Y^{k-1}$ with multiplications and inversions modulo $r(Y)$.

The algebraic closure $\overline{\mathbb{F}}_q$ of $\mathbb{F}_q$ is obtained from $\mathbb{F}_q$ by imposing that all polynomials have a root (and making one up on the spot if they don't), but is trickier to compute with. We only need $\overline{\mathbb{F}}_q$ for theoretical considerations.

3.1.2 Torsion subgroups

Definition 4. Let E be an elliptic curve over a field K and ℓ a positive integer. The *ℓ-torsion subgroup* (or just '*ℓ-torsion*' for short) of E is defined as

$$E[\ell] = \{P \in E : [\ell]P = \mathcal{O}\}.$$

Theorem 1. If ℓ is coprime to $\operatorname{char} K$ (i.e., $\ell \neq 0 \in K$), then

$$E[\ell] \;\cong\; \mathbb{Z}/\ell \times \mathbb{Z}/\ell$$

as abelian groups.

In other words, there exist (non-unique) points $P, Q \in E(\overline{K})$ such that every point $R \in E$ with $[\ell]R = \mathcal{O}$ can be written as $[\alpha]P + [\beta]Q$ for some $\alpha, \beta \in \mathbb{Z}$, and the coefficients α, β are unique modulo ℓ.

(In other other words, $E[\ell]$ is a free $(\mathbb{Z}/\ell)$-module of rank 2 and $\{P, Q\}$ is a basis.)

3.1.3 Functions on elliptic curves

We can consider *functions* from an elliptic curve E to (the algebraic closure of) its base field. Each such function is a rational expression in the coordinates of the input point. For example: On $E_d\colon x^2 + y^2 = 1 + dx^2y^2$ over a field K,

$$\phi = \frac{(x-1)(y+1)}{xy}$$

is a function that takes a point $(\xi, \eta) \in \overline{K}^2$ on E_d to the element $\frac{(\xi-1)(\eta+1)}{\xi\eta} \in \overline{K}$.

Just like rational functions over $\mathbb{R}$ (or any other field), functions on elliptic curves have *zeroes* and *poles*: For example, the function ϕ from above vanishes at $P_1 = (1,0)$ and $P_2 = (0,-1)$ and has poles at $P_3 = (-1,0)$ and $\mathcal{O} = (0,1)$.[6,7] As usual, zeroes and poles have multiplicities (for ϕ: all one), which behave additively under multiplication of functions. Hence, for example, ϕ^3 has triple zeroes at P_1 and P_2 and triple poles at P_3 and $\mathcal{O}$, and $1/\phi = \frac{xy}{(x-1)(y+1)}$ has single zeroes at P_3 and $\mathcal{O}$ and single poles at P_1 and P_2.

[5] One can prove that any single addition formula for an elliptic curve must have exceptional points *over the algebraic closure*. The nice property of Edwards curves that the addition law works for all points defined over the base field of the curve thus does not carry over to extension fields, and one (unfortunately) needs to make case distinctions when adding arbitrary points.

[6] There are quite a few technical complications under the rug: For instance, the given formula for ϕ does not make sense at $(1,0)$ and $(0,-1)$, as both numerator and denominator vanish at that point. The theory of algebraic curves dictates that one has to look at the behaviour in the *local ring* of the curve at the point under consideration, which in practice means rewriting the expression for ϕ using the curve equation until only one of numerator and denominator vanishes:

$$\phi = \frac{(x-1)(y+1)}{xy} = \frac{\left(-\frac{y^2(1-dx^2)}{x+1}\right)(y+1)}{xy} = \frac{-y^2(1-dx^2)(y+1)}{(x+1)xy} = \frac{-y(1-dx^2)(y+1)}{(x+1)x}$$

shows that $\phi((1,0)) = 0$, and a similar calculation works for $\phi((0,-1)) = 0$.

[7] Fun fact for people who know what it means: This shows that $(P_1) + (P_2) - (P_3) - (\mathcal{O})$ is a *principal divisor* on E_d, which translates to the relation $P_1 \oplus P_2 = P_3$ in the elliptic curve group.

Fact. For a function ϕ on an elliptic curve E, the sum of all zeroes and the sum of all poles are equal in the group of points on the elliptic curve.
More precisely, if ϕ vanishes at $P_1, ..., P_n \in E$ with multiplicities $x_1, ..., x_n \in \mathbb{N}$ and has poles at $Q_1, ..., Q_m \in E$ with multiplicities $y_1, ..., y_m \in \mathbb{N}$, then

$$[x_1]P_1 \oplus \cdots \oplus [x_n]P_n = [y_1]Q_1 \oplus \cdots \oplus [y_m]Q_m. \tag{1}$$

A converse is also true: If $x_1 + \cdots + x_n = y_1 + \cdots + y_m$ holds in addition to (1), then there exists a function on E with that prescribed set of zeroes and poles.
(Fun fact: From a rather general and perhaps abstract perspective, this is really the *definition* of the group law on E.)

3.2 The Weil pairing

We will now discuss the historically most significant example of a pairing on elliptic curves. Hence, fix an elliptic curve E over a field K and let ℓ be a prime not equal to $\operatorname{char} K$.[8]

Let μ_ℓ denote the group of ℓ^{th} roots of unity of $\overline{K}$, i.e., $\mu_\ell = \{a \in \overline{K} : a^\ell = 1\}$ with the usual multiplication. (Hence $(\mu_\ell, \cdot) \cong (\mathbb{Z}/\ell, +)$ as groups.)

Theorem 2. There exists a (bilinear, non-degenerate) pairing

$$e_\ell\colon\ E[\ell] \times E[\ell] \longrightarrow \mu_\ell,$$

the *Weil pairing*.

It can be defined as follows: For any point $P \in E[\ell]$, let f_P denote a function on E which vanishes at P with multiplicity ℓ and has an ℓ-fold pole at $\mathcal{O}$.[9] Then $e_\ell(P, Q)$ is defined as

$$e_\ell(P, Q) = \frac{f_P(Q + S)/f_P(S)}{f_Q(P - S)/f_Q(-S)},$$

where $S \in E$ is an arbitrary point outside the subgroup generated by P and Q.

Proof. Not extremely hard, but a bit too complicated for the time we have. The proof makes use of 'divisors' and a result called 'Weil reciprocity'. □

The Weil pairing is closely related to the *determinant pairing*: Fixing a basis $\{P, Q\}$ of $E[\ell] \cong \mathbb{Z}/\ell \times \mathbb{Z}/\ell$, we can define the map

$$\begin{array}{llcl} \det_\ell\colon & E[\ell] \times E[\ell] & \longrightarrow & (\mathbb{Z}/\ell, +) \\ & [\alpha]P + [\beta]Q, [\gamma]P + [\delta]Q & \longmapsto & \det\begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix} = \alpha\delta - \beta\gamma \end{array}$$

which is bilinear and non-degenerate.

Lemma 3. There exists a 'primitive ℓ^{th} root of unity', or in other words, a generator $\zeta \in \mu_\ell$ such that for all $P, Q \in E[\ell]$,

$$e_\ell(P, Q) = \zeta^{\det_\ell(P,Q)}.$$

Although conceptually interesting, this characterization of the Weil pairing is not helpful for computations: Finding $\alpha, \beta \in \mathbb{Z}$ such that $[\alpha]P + [\beta]Q = R$ for a given point $R \in E[\ell]$ requires solving a two-dimensional DLP, which is certainly not easier than the normal DLP itself. The intrinsic definition using functions on the curve is much better in this respect, giving rise to a polynomial-time algorithm for evaluating the Weil pairing.

[8]The assumption that ℓ is prime is too restrictive — just $\ell \neq 0 \in K$ is sufficient for most of the following.
[9]The theory guarantees that f_P always exists, and it is in not hard in practice to find such a function explicitly. One problem one may encounter is that f_P has size exponential in $\log \ell$, but fortunately there is a way around this issue when computing pairings.

Remark. There exists a simpler pairing, the *Tate pairing*, which requires computing only two f_P instead of four as in the Weil pairing above. In practice, essentially all systems use the Tate pairing, but for now we will stick with the Weil pairing for conceptual simplicity and mathematical beauty. Keep in mind that most of the following applies to the Tate pairing in the same way.

3.3 Restricting to cyclic groups

To make a pairing $\hat{e}\colon G \times G \to \mu_\ell$ on a *cyclic* group G of order ℓ, we need to restrict the inputs of the Weil pairing. However, simply taking the same subgroup $\langle P \rangle$ for both inputs yields a degenerate pairing, as the determinant pairing of two linearly dependent vectors is always zero.

Hence one makes use of a basis $\{P, Q\}$ of $E[\ell]$ such that there exists an efficiently computable isomorphism

$$\tau\colon\ \langle P \rangle \longrightarrow \langle Q \rangle;\ P \longmapsto Q.$$

Maps like τ are known as *distortion maps*. We will soon see how to construct a curve E with a distortion map.

The *modified Weil pairing*

$$\hat{e}\colon \langle P \rangle \times \langle P \rangle \to \mu_\ell$$

is then defined by

$$\hat{e}(R, S) = e_\ell(R, \tau(S)).$$

It is non-degenerate: $\hat{e}(P, P) = e_\ell(P, Q) = \zeta^{1\cdot 1 - 0\cdot 0} = \zeta \neq 1$.

3.4 Types of pairings

The fact that one needs an efficiently computable homomorphism in the previous section to make a cryptographic pairing from the Weil pairing has prompted the cryptographic community [Galbraith–Paterson–Smart '06] to classify cryptographic pairings $\hat{e}\colon G_1 \times G_2 \to T$ into (roughly) three 'types':

- **T1**: $G_1 = G_2$.
- **T2**: $G_1 \neq G_2$ but there is an efficiently computable homomorphism $G_2 \to G_1$.
- **T3**: $G_1 \neq G_2$ and there are no efficient homomorphisms between G_1 and G_2.

Note that this list is implicitly intersection-free: A 'type-2' pairing for which there also exists an efficient homomorphism $G_1 \to G_2$ is simply considered a type-1 pairing by identifying G_1 and G_2 through the homomorphisms going in both directions.

These distinctions are sometimes important for applications in constructions, since apparently theoretical cryptographers in the past have occasionally assumed the existence of pairings with all kinds of properties that nobody knew how to construct.

3.5 The embedding degree

For computational purposes, we need to know which field our objects of interest are defined over.

Note that μ_ℓ usually lies in a very large extension: It contains all roots of the degree-ℓ polynomial $U^\ell - 1 \in \mathbb{F}_q[U]$, which means one generally cannot expect the roots to lie in a $\mathbb{F}_{q^k}$ with k significantly smaller than ℓ. Since elements of $\mathbb{F}_{q^k}$ have size linear in k and arithmetic takes time exponential in $\log k$, we must make sure that k is small.

Definition 5. The *embedding degree* of a cyclic group of size ℓ is the smallest positive integer k such that the ℓ^{th} roots of unity in $\overline{\mathbb{F}}_q$ lie in $\mathbb{F}_{q^k}$, in other words, $\mu_\ell \subseteq \mathbb{F}_{q^k}^*$.

Equivalently, k is minimal with $\ell \mid q^k - 1$.

Remark. Since all values of $\hat{e}$ come from the coordinates of the input points in some way, this shows that the input groups of the pairing must as well be defined (at least) over $\mathbb{F}_{q^k}$. Hence we are now doing elliptic-curve arithmetic over fields of several thousand bits, which makes everything comparably slow.
(For comparison: 'Normal' ECC on 'well-chosen' curves requires only ≈ 256 bits.)

3.6 Example: A supersingular curve

Let $p = 4\ell - 1$ be a prime such that ℓ is also prime and consider the elliptic curve $E\colon y^2 = x^3 + x$ defined over $\mathbb{F}_p$. For reasons, this implies that $\#E(\mathbb{F}_p) = p + 1$, which means that $\#E(\mathbb{F}_p)$ has a cyclic subgroup $G = \langle P \rangle$ of size $\ell = (p+1)/4$.

The embedding degree of G with respect to $\mathbb{F}_p$ is the smallest positive integer k such that $\ell \mid p^k - 1$, hence $k = 2$ since $\ell \nmid p - 1 = 4\ell - 2$ but

$$p^2 - 1 = (p-1)(p+1) = (p-1)4\ell.$$

Moreover, the 'distortion map'

$$\tau\colon\ (x, y) \longmapsto (-x, \sqrt{-1} \cdot y)$$

is a group homomorphism, hence maps P to another ℓ-torsion point $Q := \tau(P)$. This point must be linearly independent from P as $Q \notin E(\mathbb{F}_p) \supseteq \langle P \rangle$. Hence $\{P, Q\}$ is a basis of $E[\ell]$. Therefore, we get a non-degenerate pairing from G to $\mu_\ell \subseteq \mathbb{F}_{p^2}^*$ as explained before:

$$\hat{e}(R, S) = e_\ell(R, \tau(S)) \in \mathbb{F}_{p^2}^*.$$

3.7 Miller's algorithm

The definition of the Weil (and Tate) pairing makes use of functions f_P which have an ℓ-fold zero at P and an ℓ-fold pole at $\mathcal{O}$. Clearly such a function must be made up of polynomials of degree about ℓ, which means the size of f_P is exponential in $\log \ell$. This makes computating f_P with cryptographically-sized ℓ impossible!

However, there is hope: Note that to compute pairings, we do not really need the rational-function expression for f_P, but only its value at another point Q. Moreover, note that for two functions g, h, we have the trivial fact

$$(g \cdot h)(Q) = g(Q) \cdot h(Q).$$

Hence if we can decompose f_P into a product with few distinct factors, then computing $f_P(Q)$ is easy by applying the 'trivial fact' above to the decomposed product.

Naïve idea: Decompose f_P as $g_{m-1}^{b_{m-1}} \cdots g_0^{b_0}$, where $\ell = \prod_{i=0}^{m-1} b_i 2^i$, and each g_i is a function with a 2^i-fold zero resp. pole at P resp. $\mathcal{O}$. By the multiplicativity property of zeroes and poles, we are done? **No**: There simply *is no* function on E with, e.g., a single zero resp. pole at P resp. $\mathcal{O}$: Recall that this can only exist if the set of zeroes and poles sum to the same point, which here would imply $P = \mathcal{O}$.

Miller ('86) found a good decomposition:

Lemma 4. For any $R, S \in E$, there exists an (easily computable) function $g_{R,S}$ on E which has a single zero at R and S, and a single pole at $R + S$ and $\mathcal{O}$.

Lemma 5. For fixed P and $k \geq 0$, let f_k denote a function with a k-fold zero at P, a single pole at $[k]P$, and a $(k-1)$-fold pole at $\mathcal{O}$. Such functions may be computed via the following recursive relation:

$$f_{k+k'} = f_k \cdot f_{k'} \cdot g_{[k]P,[k']P}.$$

Proof. Follows easily from the definition of $g_{R,S}$ the fact that zeroes and poles are added under multiplication of functions. □

Note that in this notation, $f_P = f_\ell$. Hence we may decompose f_P into a product of size $\log \ell$ by using the previous lemma in a square-and-multiply manner. We evaluate the functions at each intermediate stage to make sure we are not building a big (exponentially sized) rational expression for f_P. This discussion leads to the following algorithm:

Algorithm 1 (Miller '86)**.**

- **Input:** Two points $P, Q \in E[\ell]$.
- **Output:** The value $f_P(Q)$, where f_P is as defined above.

```
1. Write $\ell = \sum_{i=1}^{m-1} b_i 2^i$ with each $b_i \in \{0, 1\}$.
2. Initialize $f \leftarrow 1$ and $R \leftarrow P$.
3. For $i \in (m-1, m-2, \ldots, 1, 0)$:
    1) Let $f \leftarrow f^2 \cdot g_{R,R}(Q)$ and $R \leftarrow [2]R$.
    2) If $b_i = 1$: Let $f \leftarrow f \cdot g_{R,P}(Q)$ and $R \leftarrow R + P$.
```

- Return f.