COS 533: Advanced Cryptography — Princeton University
Lecture 11 (October 18, 2017)
Lecturer: Mark Zhandry — Scribe: Fermi Ma

Notes for Lecture 11

1 Elliptic Curve Cryptography (continued)

Three Party Key Agreement. Previously, we saw a two party key agreement protocol. Given a group G with generator g, Alice generates a random a and publishes g^a. Similarly, Bob generates a random b and publishes g^b. To agree on key $k = g^{ab}$, Alice raises g^b to the a, and Bob raises g^a to the b. The Decisional Diffie Hellman (DDH) assumption for the group G is that (g, g^a, g^b, g^{ab}) for randomly chosen a, b is computationally indistinguishable from (g, g^a, g^b, g^d) for randomly chosen a, b, d.

How do we extend this to handle three parties? A first attempt would be to have Alice and Bob do the same thing as above, and include a third user Charlie who generates a random c and outputs g^c. However, if the key is g^{abc}, it's unclear how any of the users can actually compute this. In fact, we can show that if Alice can compute g^{abc}, she can serve as an adversary to break the DDH assumption. Alice could simply multiply g^{abc} by g^{-a} to recover g^{bc}.

This motivates pairings, a tool which will make three party key agreement possible.

Pairings. A pairing e, also known as a bilinear map, maps $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_2$.

We will use different notation when for elements in $\mathbb{G}$ than for elements in $\mathbb{G}_2$, to be consistent with the literature. Generally, $\mathbb{G}$ will be an elliptic curve group, while $\mathbb{G}_2$ will be some multiplicative group. Thus, we'll use addition as the group operation for $\mathbb{G}$, but multiplication for the group operation in $\mathbb{G}_2$.

We will require some properties of e to make this map interesting:

- (Non-degeneracy) e could simply map everything to the identity in $\mathbb{G}_2$. Such an e will satisfy the remaining properties, but would also be useless for building crypto. We rule out such e.
- (Bilinearity) We require the map be linear in each input. This means that

$$e(P_1 + P_2, Q) = e(P_1, Q) \cdot e(P_2, Q),$$

and

$$e(P, Q_1 + Q_2) = e(P, Q_1) \cdot e(P, Q_2).$$

In almost all cases we are interested in, $\mathbb{G}$ will actually be a cyclic group. The bilinearity property then implies that

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

This identity will be most relevant to our discussion today.

Given these properties, we can see how to construct three-party key agreement. Suppose we have a pairing e as defined above, where P is some generator for G. Alice generates a random a and sends aP, Bob generates a random b and sends bP, and Charlie generates a random c and sends cP. Alice can compute $e(bP, cP) = e(P, P)^{bc}$, and then raise this to the a power to get $e(P, P)^{abc}$. Bob and Charlie can follow the same template to compute $e(P, P)^{abc}$. Non-degeneracy of the pairing is important here, as we don't want the key to always be the identity.

Number Theory. We'll do a brief recap of number theory before moving on. Let $\mathbb{F}$ be a field. It will sometimes be instructive to think of this as the real numbers or the complex numbers, but in crypto we'll be dealing with finite fields. An example of a finite field would be $\mathbb{Z}_p$, consisting of the integers from 0 to $p-1$ and arithmetic is done mod p.

An extension field is a field $\mathbb{F}' \supseteq \mathbb{F}$. In words, this is obtained by affixing some roots of some polynomial that does not have roots over $\mathbb{F}$ (in other words, a polynomial irreducible over $\mathbb{F}/$ cannot be factored over $\mathbb{F}$). A typical irreducible polynomial over $\mathbb{R}$ is the polynomial $x^2 + 1$. We can think of the complex numbers $\mathbb{C}$ as numbers $\{a + bi\}$ where i is a solution to the polynomial $x^2 + 1$.

Another example is over $\mathbb{F}_5$ (integers from 0 to 4 with multiplication done mod 5). Here an irreducible polynomial is x^2+x+1. For degree two polynomials irreducibility is equivalent to saying this polynomial has no solutions over $\mathbb{F}_5$ (note that for higher degree polynomials, irreducibility is a stronger condition than not having solutions over $\mathbb{F}$, since there exist reducible polynomials with no solutions). We can define an extension field with 25 elements $\mathbb{F}_{25} = \{a + bx : a, b \in \mathbb{F}_5\}$ where $x^2 = -x - 1$. What this means is that whenever we have an x^2 term, we can reduce it to $-x - 1$.

Theorem 1 *For any prime power q, there exists a unique finite field $\mathbb{F}_q$ on q elements, and moreover for any k, $\mathbb{F}_{q^k}$ is an extension field of $\mathbb{F}_q$.*

For example, if I have $\mathbb{F}_{5^4}$ this theorem tells me that is an extension of both $\mathbb{F}_5$ and $\mathbb{F}_{25}$.

Roughly speaking, we define the closure $\bar{\mathbb{F}}$ of a field $\mathbb{F}$ to be the field that results from iteratively adding more irreducible polynomials forever. For example, the closure of $\mathbb{R}$ is just $\mathbb{C}$. Once we introduce i, now all polynomials of degree 2 or higher are reducible.

For finite fields, the above theorem implies that this process never ends. Given $\mathbb{F}_q$ for some prime power q, I can always consider the extension field $\mathbb{F}_{q^2}$. In this case, we can write

$$\bar{\mathbb{F}}_q = \bigcup_{k \geq 1} F_{q^k}.$$

Lastly, we define the characteristic of a field $\mathbb{F}$ to be the smallest $n > 0$ such that $n \cdot 1 = 0$. In other words, it's the minimum number of times we need to add 1 to itself to "wrap around" to 0. For $\mathbb{F}_{p^k}$ for a prime p, the characteristic is p. For the real numbers, the characteristic is taken to be infinity.

Elliptic Curves. Let E be an elliptic curve $y^2 = x^3 + ax + b$ with a, b in some underlying field $\mathbb{F}$. While the coefficients defining this curve lie in $\mathbb{F}$, the solutions may live in an extension field of $\mathbb{F}$. For any extension field $\mathbb{F}' \supseteq \mathbb{F}$, we let $E(\mathbb{F}')$ denote the set of solutions over $\mathbb{F}'$. It turns out that $E(\mathbb{F}')$ is still a group under the elliptic curve group laws discussed in the previous lecture.

We define a torsion point to be a point with finite order. In other words, a torsion point is a point P such that $nP = 0$. This means that if I add P to itself n times under the elliptic curve group law, I end up at the point at infinity.

Note that if we're using parenthesis, $E(\mathbb{F}')$ denotes the set of solutions over $\mathbb{F}'$. We'll use brackets to denote the set of torsion points of order n over $\bar{\mathbb{F}}$ as $E[n]$.

It's not hard to show that the set of torsion points actually forms a group. Basically you just need to show that the torsion points are closed under addition, which turns out to be straightforward.

Theorem 2 *If $char(\mathbb{F}) \nmid n$, or is 0, then $E[n]$ is isomorphic to $\mathbb{Z}_n \oplus \mathbb{Z}_n$ (the set of pairs of integers mod n, with pointwise addition and multiplication). in particular, $E[n]$ is finite.*

The isomorphism might not be finite, but the important point is that this theorem tells us the size of $E[n]$.

Finally, we define $\mu_n = \{x \in \bar{\mathbb{F}} : x^n = 1\}$, also known as the set of nth roots of unity in $\mathbb{F}$. We have $|\mu_n| = n$.

Weil Pairing. We'll be using the Weil Pairing as our pairing operation. For n such that $char(F) \nmid n$, there exists $e_n : E[n] \times E[n] \to \mu_n$, with a number of desirable properties. For this lecture, we won't delve into what e_n actually is, and instead just focus on its properties / how to build crypto from such an e_n.

- (Non-degeneracy) For any Q, $e_n(P, Q) = 1 \forall P$ if and only if $Q = 0$. Clearly this implies that e_n is not degenerate.

- (Bilinearity) e_n satisfies bilinearity, as defined earlier.
- (Antisymmetry) This property will actually hurt us later. It turns out that
$$e_n(P, Q) = e_n(Q, P)^{-1}.$$
- e_n satisfies some other properties, but they won't be relevant here.

For cryptography, we want our groups to be cyclic. But as we said, the set of torsion points for a fixed n is isomorphic to a sum of two cyclic groups, and isn't itself cyclic. The most obvious fix is to pick some $P \in E[n]$ and let $G = \langle P \rangle$, the subgroup generated by P. Then we can restrict e_n to this G. The issue with this idea is that the resulting e_n is actually degenerate. To see this, we look at $e(Q, R)$ and plug in $Q = aP$ and $R = bP$. Using bilinearity and antisymmetry, we get that $e(Q, R)$ is its own inverse. This means it's either 1 or -1, but it turns out that the other properties of e_n that we're hiding will imply that it's 1.

The next attempt is to pick $P, Q \in E[n]$ such that $e(P, Q) \neq 1$. From the non-degeneracy condition, such points exist. We let $G = \langle P \rangle$. We define a second group $G' = \langle Q \rangle$. So now our pairing is $e_n : G \times G' \to \mu_n$. We'll have non-degeneracy since $e(P, Q) \neq 1$.

But now our inputs come from two different groups, which makes this an "asymmetric pairing". This is in contrast to a symmetric pairing, where the inputs come from the same group. It turns out that this is going to make our lives a little harder in terms of designing crypto, but for most applications, we can do some tweak to make it work with asymmetric pairings.

One distinction is that n the symmetric case, you can pair an element by itself. In the asymmetric case, you can't pair an element by itself anymore. If we try to do it for this particular pairing, the element by itself will always give 1.

Three Party Key Agreement from an Asymmetric Pairing. If we blindly try to apply the Weil pairing to our construction where the key is $e(P, P)^{abc}$, all the keys will be 1, so this will be degenerate. But if we use the asymmetric pairing, we'll have to change the construction. Now instead of Alice just outputting aP, Alice will also output aQ (and similarly for Bob and Charlie). This gives enough to compute $e(P, Q)^{abc}$. We won't talk about security right now.

Further Considerations. Recall from last lecture (Hasse's Theorem) that the size of the elliptic curve group over some finite field $\mathbb{F}$ is $|E(\mathbb{F})| \approx |\mathbb{F}|$. The hardness of discrete log grows with the size of the largest cyclic subgroups, so if this group were the sum of two smaller groups, say two cyclic groups of square root size, suddenly discrete log is easier.

So we usually want $E(\mathbb{F})$ to be cyclic. For pairings, we'll usually take P such that $\langle P \rangle = E(\mathbb{F})$. But this creates a problem, since Q can't lie in $\langle P \rangle$. So Q must live in some extension field $E(\mathbb{F}')$. But this creates another problem, since now Q is an element of $\mathbb{F}' \times \mathbb{F}'$ (recall that elliptic curve points are pairs of field elements). This means Q takes more space to write down than P, which gives an asymmetry where operations in $G = \langle P \rangle$ are very fast, but operations in $G' = \langle Q \rangle$ are much slower (and elements take many more bits to write down).

When designing protocols using pairings, you should be careful to minimize operations in G' and have as much as possible happen in G. We may return to this later when we describe more advanced protocols.

There is a theorem that tells us how far we the field extension needs to go before we can select a Q.

Theorem 3 *Suppose that the base field is $\mathbb{F}_q$. Then $\mathbb{F}_{q^k}$, where k is the smallest integer greater than 0 such that $n \mid (q^k - 1)$, is the smallest extension field of $\mathbb{F}_q$ such that $E[n] \subseteq E(\mathbb{F}_{q^k})$.*

Equivalently, the condition on k states that k is the smallest integer greater than 0 such that the set of nth roots of unity $\mu_n \in \mathbb{F}_{q^k}$.

For security against 2^{128} time adversaries, we need $n \approx 2^{256}$. This means we need to set $|\mathbb{F}| \approx 2^{256}$. To store an element in $\mathbb{G}$, we need one extra bit. Note that we don't need to store y, since we can just compute it given x, but we do store a bit indicating if we're using the positive or negative coordinate.

Now elements in $\mathbb{G}'$ takes $k \times 256$ bits to write down, and elements in $G_2 = \mathbb{F}_{q^k}$ also take $k \times 256$ bits to write down. So we don't want k to be very large, but we can't make it too small either. The MOV attack is an attack that reduces computing discrete log in G to computing discrete log in $\mathbb{F}_{q^k}$. It tries to compute a given (P, aP) by first pairing to get $(e(P,Q), e(P,Q)^a)$, and then solving discrete log in $\mathbb{F}_{q^k}$. There are known methods for solving discrete log slightly faster on $\mathbb{F}_{q^k}$, and to defend against these attacks we roughly need $q^k > q^{3000}$. This means we need $k \approx 12$.