Math 4527 (Number Theory 2)

Lecture $\#16$ of 38 $\sim$ February 25, 2021

Elliptic Curve Cryptography

- Encoding Messages on Elliptic Curves
- Public-Key Encryption Using Elliptic Curves
- Digital Signatures Using Elliptic Curves

This material represents §7.2.3-7.2.4 from the course notes.

Encoding Messages on Elliptic Curves, I

We can now return to the question of encoding messages on an elliptic curve $E : y^2 = x^3 + Ax + B$ modulo p, where we will now also take $p \equiv 3 \pmod{4}$.

- Since half of the units modulo p are squares, for any given x there should exist a y with $y^2 = x^3 + Ax + B \bmod p$ about half of the time.
- If we try to encode a message directly as the x-coordinate of a point, we therefore should only expect to succeed about half of the time.
- A better procedure is instead to encode a message as part of the x-coordinate of a point, and then try to choose the remaining piece of the x-coordinate in such a way that $x^3 + Ax + B$ is a quadratic residue modulo p.

Encoding Messages on Elliptic Curves, II

Here's one approach:

- Suppose p has $r+k+1$ bits when written in base 2, we break the message into pieces each containing r bits.
- Then, to convert an r-bit message m, we pad the beginning m with $k+1$ bits: a zero followed by k bits $b_1 b_2 \cdots b_k$ that can be arbitrarily chosen, and set x to be the bit string $0b_1 \cdots b_k m$.
- Next, we search through the possible choices of these k bits until we find a solution y to $y^2 = x^3 + Ax + B \pmod{p}$, and pick one of the two possible values of y arbitrarily.
- We then perform our encryption procedure using the point (x, y) on E modulo p.
- To recover the message m from a point (x, y), where $0 \le x < p$, we simply compute x modulo 2^r and write the result as a bit string in base 2.

Encoding Messages on Elliptic Curves, III

We can set the parameters in such a way that it is very likely we can find such a point for any given message piece r.

- Since there are 2^k possible choices for the bit string $b_1 b_2 \cdots b_k$, the probability that none of them yields a quadratic residue $x^3 + Ax + B$ is roughly $1 - 2^{-2^k}$.
- Of course, the probabilities are not entirely independent, but they should be fairly close to independent, certainly enough for a rough calculation like this.
- Even if we merely take $k = 10$, the failure probability is already so vanishingly small ($= 2^{-1024} \approx 1.8 \cdot 10^{-309}$) that it is unlikely a problem would ever occur in practical deployment.

Our calculation is also very efficient if we take $p \equiv 3 \pmod 4$, since then we can compute a square root of $x^3 + Ax + B$ using the proposition from earlier.

Encoding Messages on Elliptic Curves, IV

Example: Encode the message $m = 13 = 1101_2$ as a point on the elliptic curve $y^2 = x^3 + 11x + 17$ modulo $p = 307$ using a message length $r = 4$ bits and a padding length of $k = 4$ bits.

- We note that $p > 256 = 2^8$ so p has 9 bits in base 2.
- We therefore want to search for a bit string $b_1 b_2 b_3 b_4$ such that $x = 0b_1 b_2 b_3 b_4 1101_2$ is a quadratic residue modulo 307.
- The bit string 0000 yields the value $x = 13$, but $x^3 + 11x + 17 \equiv 208 \pmod{307}$ is a quadratic nonresidue as can be confirmed by evaluating $208^{153} \equiv -1 \pmod{307}$.
- The bit string 0001, however, yields $x = 29$, and $x^3 + 11x + 1 \equiv 165 \pmod{307}$ is a quadratic residue as can be confirmed by evaluating $165^{153} \equiv 1 \pmod{307}$.

Encoding Messages on Elliptic Curves, V

Example: Encode the message $m = 13 = 1101_2$ as a point on the elliptic curve $y^2 = x^3 + 11x + 17$ modulo $p = 307$ using a message length $r = 4$ bits and a padding length of $k = 4$ bits.

- To compute the associated value of y, we then compute $x^{(p+1)/4} \equiv 29^{77} \equiv 120 \pmod{307}$, since $p \equiv 3 \pmod{4}$.
- Thus, a point corresponding to the message m on the curve E is $(29, 120)$.
- To recover the message m, we simply extract the x-coordinate and reduce it modulo $2^4 = 16$. This yields the correct original message $13 = 1101_2$.

Encoding Messages on Elliptic Curves, VI

Example: Encode the message $m = 13 = 1101_2$ as a point on the elliptic curve $y^2 = x^3 + 11x + 17$ modulo $p = 307$ using a message length $r = 4$ bits and a padding length of $k = 4$ bits.

- Of course, there are many other points on E that correspond to the same message m: another is the additive inverse $(29, 187)$ of the point we found.
- We could also have searched more randomly for possible bit strings (rather than starting at 0000 and going upward), to try to keep the procedure from being as predictable. The bit string 1110, for example, yields another possible point $(237, 209)$.

Elliptic Curve Cryptography

Now that we can convert messages into points on elliptic curves, we are ready to start creating public-key cryptosystems using elliptic curves.

- We will get into the details next time.
- But you might be surprised to learn that some cryptosystems designed for $\mathbb{Z}/m\mathbb{Z}$ do not really work at all for elliptic curves, while others will.
- We will also talk about how to use elliptic curves for key exchange and digital signature algorithms, since these are fairly closely related to public-key encryption.

Elliptic Curve Cryptography, I

We now discuss the creation of public-key cryptosystems using elliptic curves, which was first proposed by Neal Koblitz and Victor Miller in 1985.

- We will assume throughout that our plaintext is a point (x, y) on a given elliptic curve E.
- We will generally work with the reduction E_p of E modulo a prime p, and N will denote the number of points on E_p.

A natural first guess for how to create a public-key cryptosystem would be to adapt RSA (or a similar method like Rabin encryption) to the elliptic curve setting: however, some difficulties will arise if we try to do this.

Elliptic Curve Cryptography, II

A quick review of RSA encryption:

- First, Bob finds two large primes p and q and computes $N = pq$. Bob also chooses an encryption exponent e that is relatively prime to $\varphi(N)$.
- Bob publishes his public key, which consists of N and e.
- To send Bob a message m, a residue class mod N, she computes $c \equiv m^e \pmod{N}$ and sends c to Bob.
- To decode a ciphertext c, Bob computes $c^d \pmod{N}$, where $d \equiv e^{-1} \pmod{\varphi(N)}$ is a decryption exponent.

The security of the algorithm comes from the fact that computing a decryption exponent d that will work for most messages is (if p, q are chosen properly) essentially equivalent to computing $\varphi(N)$, which in turn is equivalent to factoring N.

Elliptic Curve Cryptography, III

The reason that the RSA encryption-decryption procedure works is Euler's theorem:

- For any residue class m relatively prime to N, Euler's theorem says that $m^{\varphi(n)} \equiv 1 \pmod{m}$.
- Therefore, since $de \equiv 1 \pmod{\varphi(N)}$ by assumption, we have $c^d \equiv m^{de} \equiv m^1 = m \pmod{N}$.
- One may also check that if m is not relatively prime to N (which will be vanishingly unlikely if p and q are large) then the decryption procedure still does return the plaintext m.

Elliptic Curve Cryptography, IV

So now let's try to write down the elliptic curve analogy of RSA:

- Bob would create a public key consisting of an elliptic curve E, a prime p, and an "encryption multiplier" e.
- If Alice wants to encrypt a plaintext message $P = (x, y)$, she computes the ciphertext $C = eP$ on E_p and sends it to Bob.
- To decrypt a ciphertext C, Bob then computes $P = dC$ for an appropriate "decryption" multiplier d.
- In order for everything to work properly, Bob needs $(de - 1)P = \infty$ for every possible message P. From our results on orders, Bob can find such a d by requiring that $de \equiv 1 \pmod{N}$ where N is the number of points on E_p. (Note that this is exactly the same condition we had for the decryption exponent in RSA.)

Elliptic Curve Cryptography, V

So, to compute appropriate values of d and e, Bob needs to compute the number of points on E_p.

- As we have seen, this task is not entirely trivial, although there is a procedure known as Schoof's algorithm can compute the number of points on an elliptic curve modulo p in time approximately equal to $(\log p)^5$. (An improvement due to Elkies and Atkin can heuristically improve it $(\log p)^4$.)
- Roughly speaking, the idea of Schoof's algorithm is to compute the value of N modulo enough small primes that we can find N modulo r for a value of r larger than $4\sqrt{p}$: then the Hasse bound will yield a unique possible value of N.

Elliptic Curve Cryptography, VI

But here's the problem: Eve can run exactly the same computation that Bob ran to find a decryption exponent.

- Ultimately, there does not seem to be a good solution here.
- Suppose we instead try to work with an elliptic curve modulo a nonprime integer $n = pq$: then the addition law will not always work properly. If we ignore that particular issue, the system is essentially using a pair of points (P, Q), one on E_p and one on E_q, and an appropriate pair (e, d) can be found as a solution to the congruence $de \equiv 1 \pmod{N_p N_q}$.
- However, in this case, Eve would be able to break the system by factoring n, since she could then compute the values N_p and N_q using Schoof's algorithm. The usage of elliptic curves here does not add to the security, and merely serves to complicate everything.

Elliptic Curve Cryptography, VII

It does not seem feasible to construct a cryptosystem using the difficulty of inverting modular exponentiation (which is only hard when the modulus is composite).

- So let's instead try to build a system that relies on the difficulty of computing discrete logarithms, which is a more natural problem for elliptic curves modulo a prime p.
- The $\mathbb{Z}/m\mathbb{Z}$ cryptosystem that relies on computing discrete logarithms is called ElGamal encryption, which I'll review for you now.

Elliptic Curve Cryptography, VIII

Here is how the ElGamal public-key cryptosystem works:

- First, Bob chooses a prime p and a primitive root a modulo p such that it is difficult to compute discrete logarithms[1] modulo p. He also chooses an integer d with $0 < d < p-1$ and computes $b = a^d \pmod{p}$.
- Bob then publishes his public key (p, a, b).
- To send Bob a residue class m modulo p, Alice chooses a random integer k with $0 < k < p-1$ and computes $r = a^k \pmod{p}$ and $t = b^k m \pmod{p}$, and sends (r, t) to Bob.
- To decrypt a ciphertext (r, t), Bob computes $t \cdot r^{-d} \pmod{p}$.

The decryption works because
$t \cdot r^{-d} \equiv (b^k m)(a^{-kd}) \equiv (a^{kd} m)(a^{-kd}) \equiv m \pmod{p}$.

[1]Typically, this is done by ensuring $p-1$ has a large prime divisor

Elliptic Curve Cryptography, IX

Like with RSA, the only steps required to implement ElGamal are modular exponentiation and inversion (to compute r^{-d}) which are both very fast, but it is less obvious why the procedure is secure.

- Suppose Eve intercepts the transmitted information: she will obtain p, a, b, r, and t, and she wants to compute $m = t \cdot b^{-k} \equiv t \cdot a^{-dk} \equiv t \cdot r^{-d}$ modulo p.
- If Eve knows d then she can decrypt using the same procedure Bob uses. However, in order to find d from Bob's public key, Eve would need to compute the discrete logarithm $\log_a b$, which we assume she cannot do.

Elliptic Curve Cryptography, X

Furthermore, since Alice chooses k randomly, $r = a^k$ will be a random integer modulo p, as will $t = b^k m$ (since b^k is likewise random) provided $m \neq 0$.

- Knowing r alone does not help, because in order to compute k Eve would need to evaluate the discrete logarithm $\log_a r$.
- Knowing t does not help much either, because in order for Eve to compute m she would have to know the value of b^k, which in turn would require knowing the value of k.

In order to compute any one of the desired quantities to decrypt an ElGamal ciphertext, it seems that Eve would essentially have to evaluate a discrete logarithm.

- This is not a proof, of course, and it is not actually known whether breaking ElGamal encryption is equivalent to evaluating discrete logarithms.

Elliptic Curve Cryptography, XI

We can now write down the elliptic-curve version of ElGamal encryption. First, Bob must create his public key.

- To do this, he chooses an elliptic curve E, a prime p, and a point Q_a on E whose order is large.
- Ideally, Bob should choose the point Q_a to have an order whose value is a large prime roughly equal to the number of points on the curve E_p, but this can be a bit hard to arrange.
- In our description of ElGamal, Bob chose a value a that was a primitive root modulo p. It is not actually necessary to choose a primitive root: the system works essentially as well when a is any value whose order is sufficiently large that computing discrete logarithms to the base a is difficult.
- Bob can search for such a Q_a by computing $(M!)Q_a$ for a reasonably large value of M and making sure that it is not equal to ∞.

Elliptic Curve Cryptography, XII

Alternatively, Bob could try to find a curve E having a prime number of points on it: then any point other than ∞ will have order N.

- Bob then chooses a positive integer d that is less than the number of points on E_p (he does not actually need to compute the number of points itself, since he can just choose d to be less than $p - 2\sqrt{p}$) and computes the point $Q_b = dQ_a$.
- However he makes his selection, Bob then publishes (E, p, Q_a, Q_b), which serve as his public key.

Elliptic Curve Cryptography, XIII

Now suppose that Alice wants to send Bob a message $P = (x, y)$.

- Alice chooses a random integer k less than the number of points on E_p (again, she could simply choose a random integer less than $p - 2\sqrt{p}$) and computes $Q_r = kQ_a$ and $Q_s = kQ_b + P$ on E_p.
- She then sends the pair (Q_r, Q_s) to Bob.

Bob receives a ciphertext pair (Q_r, Q_s), and then wishes to recover the value of m.

- To do this, Bob simply computes $Q_s - dQ_r = (kQ_b + P) - d(kQ_a) = P + kdQ_a - dkQ_a = P$.
- Note of course that Bob would compute the subtraction as $Q_s + d(-Q_r)$, where $-Q_r$ is the additive inverse of Q_r.

Elliptic Curve Cryptography, XIV

Example: If Bob uses elliptic-curve ElGamal with $E : y^2 = x^3 + 7x + 1$, $p = 44927$, $Q_a = (7772, 14369)$, and $d = 22105$, find Bob's public key.

- First, Bob computes $Q_b = dQ_a = (39061, 4109)$ using successive doubling.
- His public key then consists of the quadruple (E, p, Q_a, Q_b).
- Remark: For the given parameters, the curve E_p turns out to have a prime number of points (44651) so P in fact has order 44651 on this curve.

Elliptic Curve Cryptography, XV

Example: If Bob uses elliptic-curve ElGamal with $E : y^2 = x^3 + 7x + 1$, $p = 44927$, $Q_a = (7772, 14369)$, and $d = 22105$. Encode the message $P = (14605, 29833)$, and then decode the associated ciphertext.

- If Alice wants to encode the message P, she chooses a random integer k less than $p - 2\sqrt{p} \approx 44503.08$. Imagine she chooses $k = 23207$.
- She then computes $Q_r = kQ_a = (30566, 37885)$ and $Q_s = kQ_b + P = (35487, 8262) + P = (40194, 40273)$ and sends them to Bob.
- Bob receives the ciphertext pair Q_r, Q_s, and then decrypts by evaluating $Q_s - dQ_r = Q_s + (35487, 36665) = (14605, 29833)$, which is indeed the correct plaintext.

Elliptic Curve Cryptography, XV

The only steps required to implement elliptic curve ElGamal are the point operations on the elliptic curve, which can be done comparatively quickly using the successive doubling algorithm. However, it is less obvious why the procedure is secure.

- Suppose Eve intercepts the transmitted information: she will obtain (E, p) along with Q_a, Q_b, Q_r, and Q_s. She wants to compute $P = Q_s - kQ_b = Q_s - dkQ_a = Q_s - dQ_r$ on E_p.
- If Eve knows d then she can decrypt using the same procedure Bob uses. However, in order to find d from Bob's public key, Eve would need to determine the value d for which $dQ_a = Q_b$, which is the elliptic curve analogue of computing a discrete logarithm.

Elliptic Curve Cryptography, XVI

Furthermore, since Alice chooses k randomly, $Q_r = kQ_a$ will essentially be a random point on the curve E_p (technically, it will be a random multiple of Q_a, but this does not tell Eve very much if Q_a has a large order).

- Likewise, $Q_s = kQ_b + P$ will be essentially random.
- Knowing Q_r alone does not help, because in order to compute k Eve would again need to compute an elliptic-curve discrete logarithm.
- Knowing Q_s does not help much either, because in order for Eve to compute P she would have to know the value of kQ_b, which in turn would require knowing the value of k.
- Ultimately, like with the modular version of ElGamal, the only obvious method of attack is to compute a discrete logarithm.

Elliptic Curve Cryptography, XVII

Ultimately, like with the modular version of ElGamal, the only obvious method of attack is to compute a discrete logarithm.

- Pleasantly, it appears to be much harder to compute elliptic curve discrete logarithms than modular discrete logarithms.
- There is a version of the Pohlig-Hellman algorithm (which operates on a similar principle to Pollard's $(p-1)$-algorithm) for elliptic curves that is effective when the number of points N on E_p has only small prime divisors.
- In this case, N plays the role of $p-1$ in the algorithm, and the number of steps is on the order of the largest prime divisor of N.
- This situation is easy to avoid if the curve E is chosen properly (i.e., so that it has a point of large order on it, meaning that N has at least one large prime factor).

Elliptic Curve Cryptography, XVIII

There is also a so-called "baby-step giant-step" method whose procedure requires approximately $p^{1/2}$ steps to compute a discrete logarithm. Here is the method:

- To find a solution to $dQ_a = Q_b$ on an elliptic curve E_p modulo p, choose an integer M and then compute two lists: the points xQ_a for all $0 \leq x \leq M-1$ and the points $Q_b - MyQ_a$ for all $0 \leq y \leq M-1$.
- Then compare the two lists to find an element that is on both lists: if $xQ_a = Q_b - MyQ_a$, we get a solution $d = x + My$.
- By (more or less) the pigeonhole principle, we would expect to get a collision between the two lists when $M^2 \geq N$.
- Thus, since we compute two lists of size M, where $M \approx p^{1/2}$, the number of steps is on the order of $p^{1/2}$.

Elliptic Curve Cryptography, XIX

However, there does not appear to be any natural analogue of any of the sieving algorithms.

- The basic reason is that the sieving algorithms all rely on an easily-computed notion of "smallness" of a residue class modulo n that remains consistent under modular multiplication (i.e., the product of two small numbers modulo n remains small modulo n).
- The idea is then to try to obtain a large number of relations among small primes and use them to compute the discrete logarithms of enough small primes to allow new discrete logarithms to be computed rapidly.

Elliptic Curve Cryptography, XX

However, there is no analogous notion of size that is easy to compute on an elliptic curve modulo p.

- For one thing, even if the x-coordinate of a point is small, the y-coordinate will look more or less random and very often will be large.
- Also, even if all the coordinates of particular points are both small, their sum may have very large coordinates due to the modular divisions in the addition law.
- Finally, even if we were to declare that a point is "small" if it had a small x-coordinate, there is no easy way to see how a large point can be written as a sum of small points that is analogous to the way we can easily factor a big integer that is a product of small primes.

Elliptic Curve Cryptography, XXI

Since the sieving algorithms do not carry over, and there do not seem to be any other natural algorithms that are comparable, we can achieve a level of security comparable to that of RSA using an elliptic curve cryptosystem with much smaller key sizes.

- It is estimated, based on the speed of integer factorization algorithms versus the speed of elliptic curve discrete logarithm algorithms, that an elliptic curve cryptosystem with a key size of 256 bits provides security roughly comparable to that of RSA with a key size of approximately 3000 bits.
- The smaller key size leads to significant savings in computation time, even after accounting for the additional complexity of doing elliptic curve addition versus modular multiplication: indeed, many websites (e.g., Google, as of when I wrote this slide) are currently using elliptic-curve public-key encryption as part of https.

Elliptic Curve Cryptography, XXII

In practice, since it is hard to count the number of points on a given elliptic curve, many elliptic curve protocols specify a curve published by an independent authority, such as NIST, that has done the point-counting ahead of time and certifies it as secure.

- Of course, this requires a degree of trust[2][3] that the authority has not intentionally chosen a curve that has some kind of nonobvious "backdoor" (i.e., some clever way of computing discrete logarithms quickly), though in practice it seems unlikely such a backdoor would exist for a nonsingular curve.
- As a last remark, many of these implementations use elliptic curves over large finite fields of characteristic 2 (since the resulting binary arithmetic is more efficient).

[2]And for some authorities, possibly an unwise degree of trust; cf., the backdoor that was allegedly[3] placed in the Dual_EC_DRBG algorithm....

[3]I probably have to say "allegedly" here for legal reasons.

Elliptic Curve Diffie-Hellman, I

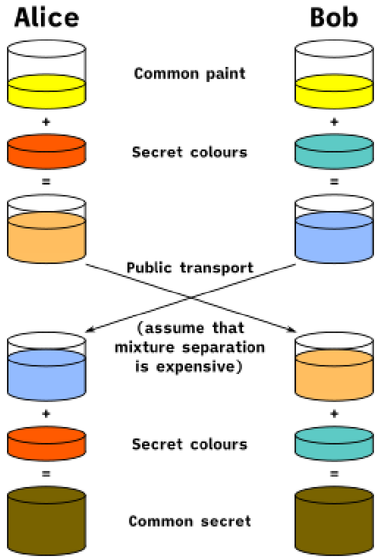
Public-key protocols are fast for small messages, but if Alice needs to send Bob megabytes (or gigabytes or terabytes) of encrypted data, even a very fast implementation of public-key encryption will take an unreasonably long time to encode and decode.

- Symmetric cryptosystems generally do not require nearly as much computation and can be done comparatively efficiently even for large amounts of data.
- Thus, in practice, most efficient cryptographic protocols will require some sort of "key exchange", wherein Alice and Bob must somehow decide what encryption key to use for their symmetric cryptosystem.
- One way to do this is to use an asymmetric cryptosystem to send the key: Alice chooses a key, encrypt it using Bob's public key, and send it to Bob: then Bob can decrypt the message and obtain the key.

Elliptic Curve Diffie-Hellman, II

We will now describe a different procedure for key exchange that is an elliptic-curve version of the Diffie-Hellman key exchange procedure.

The idea of Diffie-Hellman is quite simple, and is contained in this image from wikipedia:



Elliptic Curve Diffie-Hellman, III

The standard implementation of Diffie-Hellman using $\mathbb{Z}/m\mathbb{Z}$ works as follows:

- First, Alice and Bob jointly choose a large prime number p where it is hard to compute discrete logarithms, along with a primitive root g modulo p.
- Alice chooses a secret integer a, and sends Bob $g^a \pmod{p}$.
- Bob chooses a secret integer b, and sends Alice $g^b \pmod{p}$.
- Then the secret key s is given by $g^{ab} \pmod{p}$, which both of them can compute.
- Alice knows a, and has the value of g^b from Bob, so she needs only raise g^b to the ath power.
- Similarly, Bob knows b and has the value of g^a from Alice, so he needs only raise g^a to the bth power.

Elliptic Curve Diffie-Hellman, IV

If Eve is eavesdropping on the conversation, she will have the values of p, along with g, g^a, and g^b modulo p, and she wants to compute the secret key $g^{ab} \pmod{p}$.

- In order to do this, Eve would essentially need to compute one of the exponents a and b; since g is a primitive root, this is equivalent to calculating the discrete logarithm $\log_g(g^a)$ or $\log_g(g^b)$ modulo $p-1$.
- This discrete logarithm calculation is believed to be hard in general, though both integer factorization and discrete logarithm calculations can be performed in polynomial time using Shor's algorithm on a quantum computer.

Elliptic Curve Diffie-Hellman, V

It is not hard to construct an elliptic-curve version of Diffie-Hellman key exchange for elliptic curves using the same ideas.

- First, Alice and Bob jointly choose a large prime p, an elliptic curve E_p modulo p, and a point P on E having large order.
- Alice chooses a secret integer $a < \mathrm{ord}(P)$, and sends Bob $Q_a = aP$.
- Bob chooses a secret integer $b < \mathrm{ord}(P)$, and sends Alice $Q_b = bP$.
- Then the secret key s is given by $Q_{ab} = (ab)P$, which both of them can compute: Alice evaluates $a(bP)$ while Bob evaluates $b(aP)$.

Elliptic Curve Diffie-Hellman, VI

Example: Use elliptic-curve Diffie-Hellman to construct a secret shared key using $E : y^2 = x^3 + 7x + 1$, $p = 44927$, and $P = (27844, 29401)$, where Alice's secret number is $a = 40006$ and Bob's secret number is $b = 18846$.

- Alice computes $Q_a = aP = (3454, 34367)$ and sends it to Bob. Bob computes $Q_b = bP = (22472, 6971)$ and sends it to Alice.
- Alice then recovers $Q_{ab} = aP_b = (2147, 22480)$ and Bob recovers $Q_{ab} = bQ_a = (2147, 22480)$.
- Bob and Alice now have a secret shared key $Q_{ab} = (2147, 22480)$ that they can use for further communications (e.g., with a symmetric-key cryptosystem).

Elliptic Curve Diffie-Hellman, VII

If Eve is eavesdropping on the conversation, she will know E_p along with P, Q_a, and Q_b , and she wants to compute Q_{ab}.

- In order to do this, Eve would essentially need to compute one of the multipliers a and b. Since P is assumed to have large order, the only reasonable way to do this is for her to evaluate a discrete logarithm on E_p.
- Again, as we have already discussed, computation of discrete logarithms on elliptic curves appears to be very difficult.
- It is of course possible that there is some way to combine the information in P, Q_a, Q_b to find Q_{ab}, but this seems unlikely since the operations of scaling a point by a and scaling a point by b are essentially independent.

Elliptic Curve Diffie-Hellman, VIII

Both the modular and elliptic-curve Diffie-Hellman protocols we have described have no authentication, and are susceptible to a "man-in-the-middle" attack.

- In this attack, Mallory impersonates Alice to Bob and simultaneously impersonates Bob to Alice, and performs a simultaneous key exchange with both of them.
- Then, Mallory will be able to decode messages sent from Alice, and then re-encrypt them to send to Bob.
- As far as Alice and Bob can tell, they are communicating with each other, since their messages are received correctly, at least as long as Mallory is in the middle decoding and re-encoding the messages.

Elliptic Curve Diffie-Hellman, VIII

The problem is that the basic Diffie-Hellman protocol does not authenticate Alice and Bob to one another before creating the key.

- One way to include an authentication step would be for both of Alice and Bob to put a digital signature on their communications during the key creation process, so that the other person feels confident that Mallory is not impersonating either of them.
- We can also use elliptic curves to create digital signatures, which we now describe.

Elliptic Curve Digital Signatures, I

A digital signature must be created in such a way that binds it both to its creator (so that Bob knows Alice and not Eve was the signer and the sender) and to its associated message in a way that cannot easily be altered (so that Bob knows Eve didn't change the message).

- The goal when designing a digital signature algorithm is not to keep the message from being deciphered, but rather to prevent the signature from being easily decoupled from Alice's identity or from Alice's original message.
- Ultimately, however, these ideas are similar enough that we can adapt public-key cryptosystems to create digital signature algorithms.

Elliptic Curve Digital Signatures, II

Here is a digital signature algorithm based off of the ElGamal cryptosystem.

- Alice first creates an ElGamal public key (p, a, b), where p is a large prime for which it is hard to compute discrete logarithms, a is a primitive root mod p, and $b \equiv a^d \pmod{p}$ for her secret choice d with $0 < d < p - 1$.
- If Alice now wants to sign a message m, she first chooses a random integer k relatively prime to $p - 1$.
- She then computes $r \equiv a^k \pmod{p}$ and $s \equiv k^{-1}(m - dr) \pmod{p-1}$, and her signature is the triple (m, r, s).
- If Bob wants to verify that Alice really signed the message m, he checks whether $b^r r^s$ is congruent to $a^m \pmod{p}$. If so, then he accepts the signature as valid, and if not he rejects it.
- This works $b^r r^s \equiv (a^d)^r a^{ks} \equiv a^{dr} a^{m-dr} \equiv a^m \pmod{p}$.

Elliptic Curve Digital Signatures, III

Suppose now that Eve has intercepted a message pair (m, r, s) that Alice has signed and wants to forge Alice's signature on a new message w.

- Obviously, Eve cannot simply use the signature pair (w, r, s), since Bob will compute $b^r r^s \equiv a^m \not\equiv a^w \pmod{N}$ and reject the signature as invalid.
- In order to find a valid signature z for her message w, she needs to find (r, s) that are solutions to the congruence $b^r r^s \equiv a^w \pmod{N}$.
- If Eve picks a particular r and searches for s, she is attempting to solve $r^s \equiv a^w b^{-r} \pmod{N}$, which is equivalent to computing the discrete logarithm $\log_r(a^w b^{-r})$.

Elliptic Curve Digital Signatures, IV

Another possibility is for Eve to try to choose the value of s first, but this requires solving an even more unusual congruence $b^r r^s \equiv a^w \pmod{N}$, which is a combination of a discrete-log and root-extraction problem.

- It may be possible to choose r and s together in some more efficient manner, but it is not obvious how such a procedure would work.
- Ultimately, if we believe it is difficult to compute discrete logarithms modulo p, then it should also be difficult to forge Alice's ElGamal signature.

Elliptic Curve Digital Signatures, V

We will now describe how to adapt the ElGamal signature algorithm to the elliptic curve setting.

- Some details of the algorithm differ slightly from the modular case since we are dealing with points rather than individual numbers.
- Alice first creates an elliptic-curve ElGamal public key (p, E, Q_a, Q_b) where p is a large prime, E is an elliptic curve modulo p on which it is hard to compute discrete logarithms, Q_a is a point on E whose order has only large prime factors, and $Q_b = dQ_a$ for Alice's secret number d.
- Alice also calculates the number of points N on E_p.

Elliptic Curve Digital Signatures, VI

So, Alice has an elliptic-curve ElGamal public key (p, E, Q_a, Q_b) where Q_a is a point on E whose order has only large prime factors, and $Q_b = dQ_a$ for Alice's secret number d, and E has N points.

- To sign a message m (an integer modulo N), Alice first chooses a random positive integer k relatively prime to N.
- She then computes $Q_r = kQ_a = (x, y)$ and $s = k^{-1}(m - dx) \pmod{N}$, and sends Bob her signed message (m, Q_r, s).
- Bob verifies that Alice's signature is correct by computing $xQ_b + sQ_r$ and comparing it to mQ_a. If the results are equal, he accepts the signature, and otherwise he rejects it.
- The verification works because $xQ_b + sQ_r = x(dQ_a) + s(kQ_a) = (m - dx)Q_a = xdQ_a + mQ_a - dxQ_a = mQ_a$, where we are using the fact that $sk \equiv m - dx \pmod{N}$ to deduce that $ksQ_a = (m - dx)Q_a$ since the order of Q_a necessarily divides N.

Elliptic Curve Digital Signatures, VII

As with the elliptic-curve ElGamal encryption scheme, the security of this procedure ultimately relies on the difficulty of computing a discrete logarithm and the fact that k is randomly chosen.

- It does not depend on the difficulty of computing the number of points on the curve N, which could even be published as part of the public key if desired.

Elliptic Curve Digital Signatures, VIII

Example: Alice publishes her elliptic-curve ElGamal signature key with $E : y^2 = x^3 + 7x + 1$, $p = 44927$, $Q_a = (3174, 1067)$, and $Q_b = dQ_a = (38921, 25436)$ with her secret $d = 25661$. Bob then sends her the message $m = 17781$. Generate a signature for this message with $k = 33050$ and verify that it is correct.

- Alice computes the number of points on the curve, $N = 44651$, which happens to be prime.
- She then computes $Q_r = kQ_a = (11123, 34794) = (x, y)$ and $s = k^{-1}(m - dx) \equiv 42665 \pmod{N}$.
- She then sends the pair (Q_r, s) to Bob, who then evaluates $xQ_b + sQ_r = (29063, 26534) + (36219, 42811) = (35670, 7590)$ and compares it to $mQ_a = (35670, 7590)$.
- The results are equal, so Bob accepts the signature.

Summary

We discussed public-key encryption using elliptic curves.

We discussed elliptic-curve Diffie-Hellman key exchange.

We introduced elliptic-curve digital signature algorithms.

Next lecture: Rational points on elliptic curves.