# Cryptology (7)

## Public Key Encryption—Elliptic Curve Cryptography

John York, Blue Ridge Community College

http://www.brcc.edu

Much of the information in this course came from “Understanding Cryptography” by Christoff Parr and Jan Pelzl, Springer-Verlag 2010

Much of the classical cryptography material and most Python scripts came from “Cracking Codes with Python” by Al Sweigart, NoStarch Press 2018

Also helpful, “Cryptography Engineering” by Ferguson, Schneier, and Kohno, Wiley Publishing, 2010

## Obligatory XKCD Cartoon


https://imgs.xkcd.com/comics/legal_hacks.png
"It's totally a reasonable modern analogue. Jefferson would have been all about crypto."

If crypto was classified as a weapon, then we could use the second amendment to keep the government from degrading it with back doors.

Cryptography diluted by government has caused problems in the past. There was a time that fully secure cryptology was not allowed to be exported to countries like USSR and China. Instead, vendors were required to use short key lengths (less than or equal to 512 bit keys) for "export-grade" cryptography. This in itself did not cause a problem.

What did cause a problem happened a few years later when a typical computer had the power to break 512 bit keys. Most servers used 1024 or 2048 bit keys, but still supported the legacy export encryption with 512 bit keys. At that point people developed attacks that forced a server to downgrade the key size to 512 bits, and then they could crack the key and decrypt the message. One attack against RSA encryption was the FREAK attack, and an attack against Diffie-Hellman was Logjam.

The problem was support of encryption that was degraded by government past its lifetime. The solution was to make servers stop supporting export grade encryption. FREAK: https://en.wikipedia.org/wiki/FREAK, Logjam: https://weakdh.org/

## Discrete Logarithm Problem (DLP)

$A = B^N = B^* B^* B^* B^* B^* B^* \ldots B$

Multiply N times

- Find N when A and B are known
  - $N = \log_B(A)$ in the continuous world
- Useful in encryption because A jumps around erratically when you use modular arithmetic, hard to predict
- What if we could find an operation other than multiplication that was **even more erratic** in modular arithmetic??

In Diffie-Hellman, the operation we used in the Discrete Logarithm Problem was multiplication. For example, part of Alice's private key was a number we'll call a. She computed part of her public key by using $A = \alpha^a \bmod p$. If Eve wants to crack Alice's private key, she has reverse Alice's equation to compute a. The "mod p" part is what made this difficult. Discrete Logarithm Problem… stated mathematically it is the discrete logarithm over a finite field, which is where the term Finite Field Cryptography (FFC) comes from.

Suppose we want to make the problem even harder for Eve to crack, so that we can use smaller keys and save computation time. One way to do this is to find an operation other than multiplication that's even harder to break.

## Generalized DLP

$A = B^N = \underbrace{B \circ B \circ B \circ B \circ B \circ B \circ \ldots B}$

Some operation N times

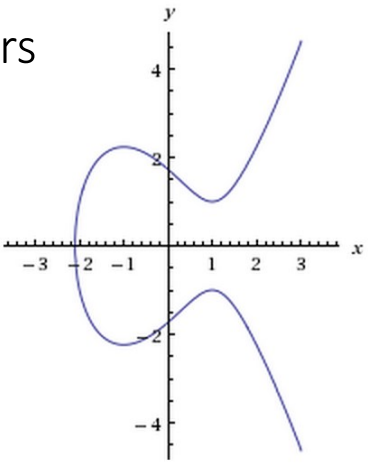
- Create some operation which is <u>really</u> hard to predict
  - Operation and elements must form a finite cyclic group
- One answer: Elliptic Curves
  - Note: The term "elliptic" comes from elliptic integrals, which were originally used to compute arc length along an ellipse. Elliptic Curves don't draw ellipses.

Rather than creating a brand new operation for encryption, mathematicians modified an old one to suit their purposes. The equation for the arc length around an ellipse leads to new functions similar to sin or cos, called elliptic integrals. The math of elliptic curves is complicated, but we are lucky. The part used in cryptography involves fairly simple equations that can be visualized with simple pictures.

The bottom line is that someone found complicated math that they could use to replace multiplication in our DLP. The result is a new type of DLP that allows us to use shorter keys and less computation.

# Elliptic Curve over real numbers

- In real numbers, an elliptic curve satisfies
  - $y^2 = x^3 + ax + b$
  - Discriminant $-16(4a^3 + 27b^2) \neq 0$ to prevent singularities
- We will use a plot over real numbers as an analog for our new operation
- Graphs of several possible shapes in http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/ or https://en.wikipedia.org/wiki/Elliptic_curve

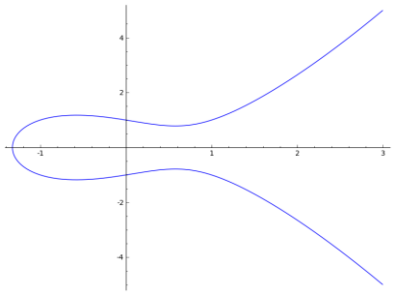



$y^2 = x^3 - 3x + 3$

If you select a curve that looks like this, you can define simple operations using straight lines.

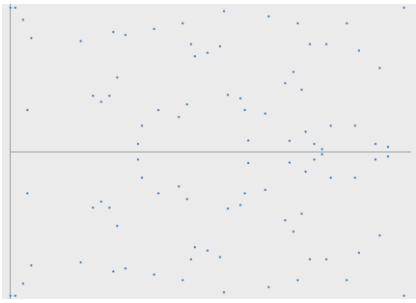
Although the classic equation used in ECC is $y^2 = x^3 + ax + b$, there are now curves that include an $x^2$ term, like $y^2 = x^3 + ax^2 + bx$ or $y^2 = x^3 + ax^2 + bx + c$.

## Elliptic Curve over a finite field (discrete)



$y^2 = x^3 - x + 1$

<= Real
Discrete =>
(mod 97)



Because discrete math uses only whole numbers, and "wraps" when it reaches an edge (modular arithmetic), the discrete curve doesn't look like the real number curve. The analog still works, though.

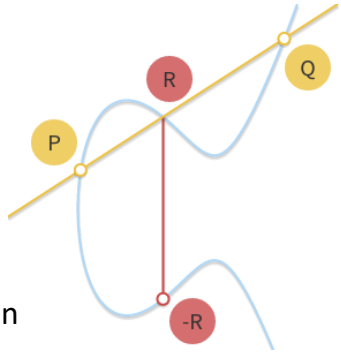
https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/

The graph of the discrete math version of the elliptic curve doesn't look at all like the continuous version. First, the discrete version only includes single points. Second, the modulo component may cause the answer to wrap several times. After all, the modulus is what makes encryption work in the first place.

None the less, graphical methods and equations developed in the continuous version will work well in the discrete world. Best of all, someone else has already found the equations and the algebra of those equations is not difficult.

## Invent an operation on an elliptic curve

- A line intersects curve in three places
  - Let P ◦ Q ◦ R = 0 which means P ◦ Q = -R
- To find P ◦ Q, draw a line between them
- Then find the third point R
- Mirror R across X axis ( y -> -y )
- This is called Point Addition and written as P + Q




http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/

Since the operation is defined as P ◦ Q ◦ R = 0, P ◦ Q = -R. So, if P and Q are the points we want to operate upon, we connect them with a line. The shape of the curve tells us the line will usually intersect the curve in a third spot. We drop (or rise) vertically from the third intersection until we intercept the curve again, which gives us our answer.

From Paar, page 241-244

If we use the curve $y^2 = x^3 + ax + b$, and known points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ we can find $R = (x_3, y_3)$

P and Q define a line and the equation of that line is $y = sx + m$ (the German text uses $y = sx + m$ instead of the $y = mx + b$ you are used to.) Substitute $y = sx + m$ into y2 in the curve, and grind away for 30 min. The solution is

$x_3 = s^2 - x_1 - x_2 \bmod p$ and $y_3 = s(x_1 - x_3) - y_1 \bmod p$

Where $s = (y2 - y1)/(x2 - x2)$ when $P \neq Q$ or

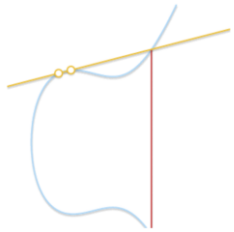
$s = (3x_1^2 + a)/2y_1$ when $P = Q$

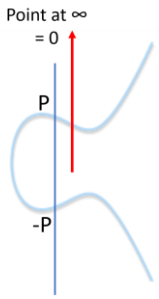
The algebra to derive the equations for $R = (x_3, y_3)$ is messy, but doable. The equations themselves are fairly easy to use.

## Special Cases

- P + Q, but Q = P
  - As P approaches Q, the line approaches a tangent
  - Use the tangent line
  - P + P is also called 2 P
- P + Q, but Q = -P
- Vertical line, only two intersections
  - Create 0 = point at infinity
  - Also gives us an inverse, P + -P = 0
- Repeated addition, add P to itself N times
  - Also called N P

http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/






If we want to do P + Q, or P + P when P = Q, the simplest way is to take the tangent to the curve at P. Then find the other intersection and drop down to -R. Again, someone has already done the math and the equations are incorporated into the ECC algorithm.

If we need to find P + -P, that should equal zero. The line through P and -P will be a vertical line, which appears to be a problem. The designers of the algorithm just said that the third point we need is at infinity, and we'll call that point zero.

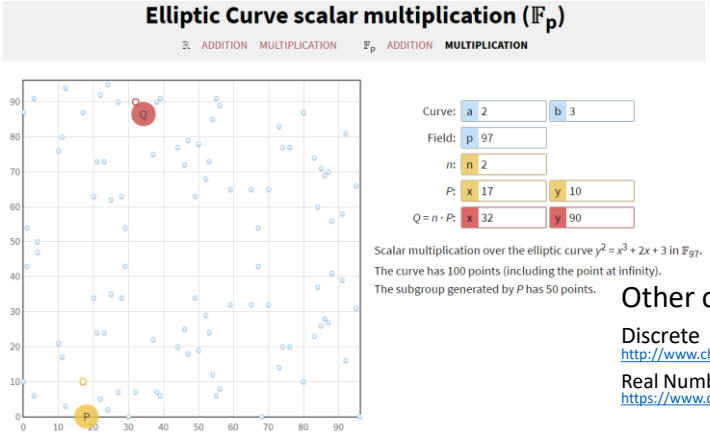
P + P is important, because that is the basis of our DLP. We will need to calculate P added to itself many times, P + P + … + P, or NP.

The solution of the elliptic curve used compute P + Q and P + Q = 2P is fairly simple. See Paar pg 244.

# Elliptic Curve Calculators

Both Real Numbers and Discrete

https://cdn.rawgit.com/andreacorbellini/ecc/920b29a/interactive/modk-add.html




## Other calculators

Discrete
http://www.christelbach.com/eccalculator.aspx

Real Numbers
https://www.desmos.com/calculator/ialhd71we3

We will use the great calculator provided by Andrea Corbellini. We will use the discrete (Fp) portion, with Multiplication. It is the top right option.

Enter a and b for the curve. When a = 2, and b = 3, you are using the curve $y^2 = x^3 + 2x + 3$. The Field box allows you to select the modulus, 97 in this case. The Base Point is P, with x and y coordinates (the page will force you to select a point that is on the curve, and is awkward sometimes.) The last line, "The subgroup generated by *P* has 50 points," tells us the size of the subgroup, often referred to as E+.

## Elliptic Curve Diffie-Hellman (ECDH) Setup

- Choose a curve and modulus
  - (There are now some curves that include an $x^2$ term)
- Choose a point on the curve
  - "primitive element" or base point P = ($x_P$, $y_P$)
- Curve selection based on
  - Security
  - Speed of computation
- Small set of standard curves http://safecurves.cr.yp.to/
- NIST curves thrown out by some, after NSA suspected of putting a back door in NIST's random number generator https://en.wikipedia.org/wiki/Dual_EC_DRBG

Curves and base points are usually selected ahead of time by standards bodies or well known cryptographers. It is interesting to note that curves supported by NIST fell from favor in some circles after the NSA was suspected of placing a back door in NIST's random number generator. Also, the NIST equations contain long complicated constants with no explanation of why they are there. Compare these two curves (from http://safecurves.cr.yp.to )

Curve 25519: y^2 = x^3+486662x^2+x , modulo p = 2^255 - 19

NIST P-256: y^2 = x^3-3x+41058363725152142129326129780047268409114441015993725554835256314039467401291 , modulo p = 2^256 - 2^224 + 2^192 + 2^96 - 1

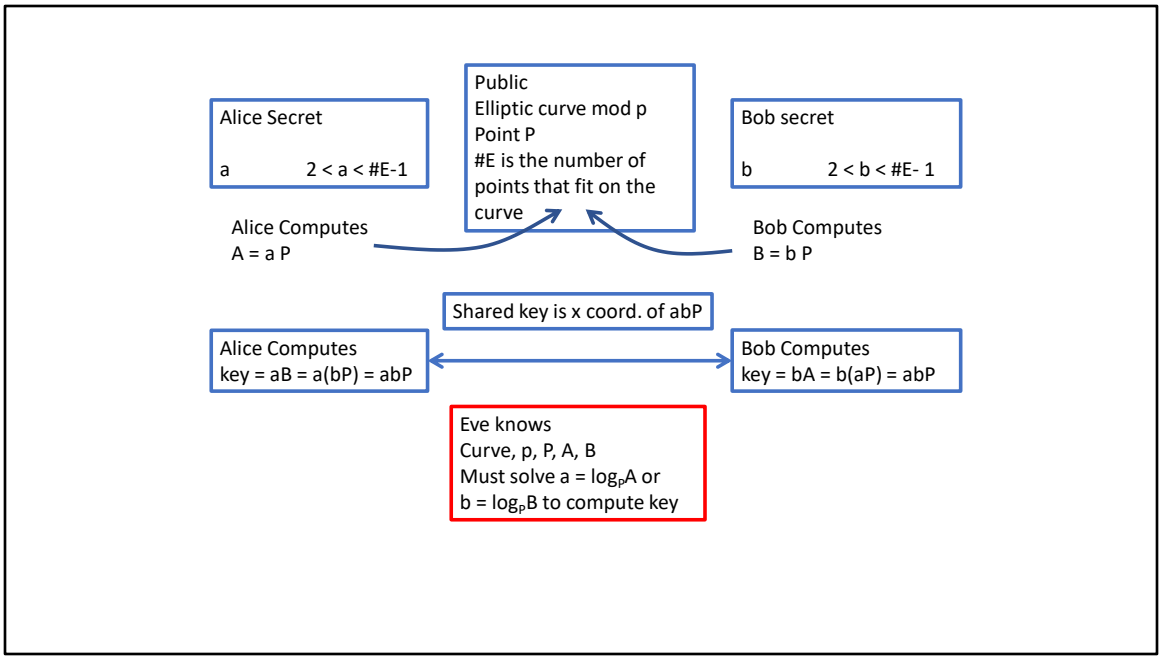
Also, the author of Curve 25519 took pains to ensure his method did not conflict with existing patents on ECC, and listed it as public domain. https://cr.yp.to/ecdh.html

## ECDH Key Exchange

- Alice chooses secret integer, a, gives Bob (public) A = aP
- Bob chooses secret integer, b, gives Alice (public) B = bP
- Alice computes key aB = a(bP) = abP
- Bob computes key bA = b(aP) = abP
- abP is the shared key
- Eve knows curve, P, A and B
  - Must solve $a = \log_A P$, or $b = \log_B P$ to find abP

This procedure is that same that we used in Diffie-Hellman, except that instead of using $A = \alpha^a$, or A = pow(alpha, a, p), we will use the ECC calculator.

Although the syntax makes it look like simple multiplication (A = aP), remember that we are using the operation ○, which we have defined as P + Q = R on an elliptic curve. If you set that aside, our procedure is the same as plain Diffie-Hellman, with one huge exception. Instead of using Python pow(α, x, p), we use an elliptic curve calculator.




Here are the steps to follow for ECC. The curve, modulus, and base point P are known ahead of time, or can be included in the setup messages in an unencrypted form.

Curve _____ Modulus _____

P _____, _____

Alice

Select a _____

Compute A _____ A = a P

Give A to Bob

Bob

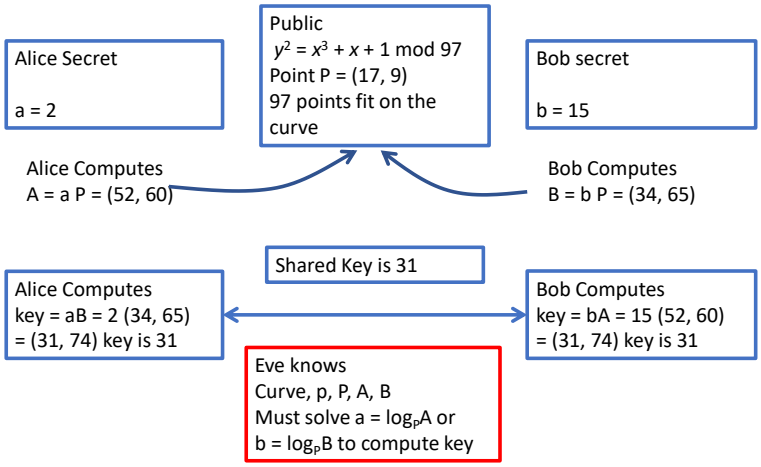
Select b _____

Compute B _____ B = b P

Give B to Alice

Alice computes key = a B (she picked a, Bob gave her B ) _____

Bob computes key = b A (he picked b, Alice gave him A) _____




Verify this using https://cdn.rawgit.com/andreacorbellini/ecc/920b29a/interactive/modk-mul.html

Here are the steps to follow for ECC. The curve, modulus, and base point P are known ahead of time, or can be included in the setup messages in an unencrypted form.

Curve $y^2 = x^3 + 1x + 1$ (a=1, b=1 in the calculator) Field 97 (p in the calc.)
P 17, 9 (P: x and y coordinates in the calc.)

Alice
Select a 2 (n = 2 in the calculator)
Compute A 52, 60 A = a P (Q + n*P: x and y in the calculator)
Give A to Bob

Bob
Select b 15 (n = 2 in the calculator)
Compute B 34, 65 B = b P (Q + n*P: x and y in the calculator)
Give B to Alice

Alice computes key = a B 31, 74 (a is n = 2, B is P: 34, 65 from Bob)
Bob computes key = b A 31, 74 (b is n = 15, A is P: 52, 60 from Alice)
Shared key is 31

## Elliptic Curve Cryptography (ECC) Notes

- The most effective attack against RSA and DH (index-calculus method) does not work against ECC
  - Allows much smaller key sizes, 224 bit versus 2048 (modulus is still 2048 bits)
- Computation is generally faster
  - Preferred for phones and devices with limited CPU
- Adoption was initially slow
  - Community acceptance of new method was slow
  - Patents for some ECC methods slowed implementation
- Expected to become predominant method soon

TLS 1.3 requires that Diffie-Hellman or ECC be used for encryption, RSA is no longer allowed. Since the patent issues related to ECC appear to be resolved, and ECC is faster, ECC will become the dominant public key encryption method.

## Subgroups

- Subgroups occur when the number of elements (#E) in a group is not a prime number
  - Remember subgroups from Diffie-Hellman
- There is one subgroup for each divisor of #E
  - If #E = 30, subgroups with 1, 2, 3, 5, 6, 10, and 15 members
- With point multiplication ( Q = n P ), the only possible values for Q are members of P's subgroup
  - This is a problem if P's subgroup is small
  - Homework will demonstrate this using an Elliptic Curve Calculator
- This is one reason why curves and base points are chosen by standards bodies, and not selected on the fly

We need to be careful that the curve, modulus, and base point we select put us into a large subgroup so that our key has as many possibilities as we can get. Just as in Diffie-Hellman, if we fall into a subgroup that has a small number of elements we will be in trouble.