Applied Cryptography and Network Security

William Garrison
bill@cs.pitt.edu
6311 Sennott Square

Elliptic-Curve Cryptography

University of Pittsburgh

Overview

Why do we need different public-key schemes?

What are elliptic curves, and how can they help?

- EC definitions
- Finite ECs
- Multiplication on EC points
- Elliptic-curve discrete log problem

What schemes are based on elliptic-curve math?

- Elliptic-Curve Diffie–Hellman
- Elliptic-Curve Digital Signature Algorithm

Recall that the "first-gen" public-key crypto requires large keys

Security strength	Symmetric key size	Hash output size	RSA/DH/DSA modulus size
112	112	224	2048
128	128	256	3072
192	192	384	7680
256	256	512	15360

Why?

- Hint: It's about the mathematical problem(s) that are used

This reflects the difficulty of solving discrete log (for Diffie-Hellman and DSA) or factoring (for RSA)

- Both have sub-exponential solutions, so key sizes grow super-linearly with strength

But… why are large keys a problem?

What are some issues with large key sizes?

- Transmission size increases network (and thus energy) usage
- More importantly: Operating on large values yields increased runtime

Let's turn to mathematics to reduce key size

What are we searching for if we want to improve public-key cryptography efficiency?

- Mathematical problems…
- … in NP…
- … where known solutions are exponential in runtime on conventional computers

An exponential runtime would allow us to use smaller keys with less reduction in security strength

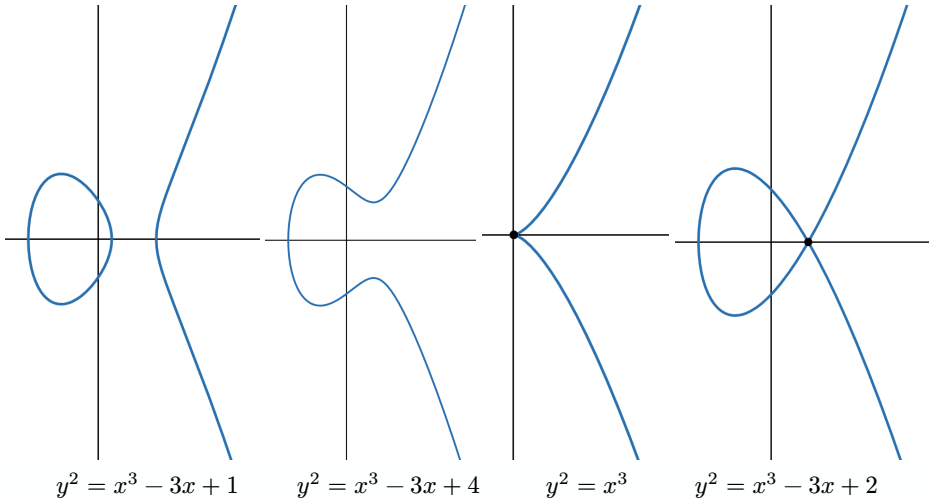
Elliptic curves provide us with a domain where such problems can be found!

An elliptic curve is a set of points on the coordinate plane satisfying:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

For efficiency, it is common to transform the equation to this form:

$$y^2 = x^3 + ax + b$$



We will focus on smooth curves like the first two, not singular curves like the latter two

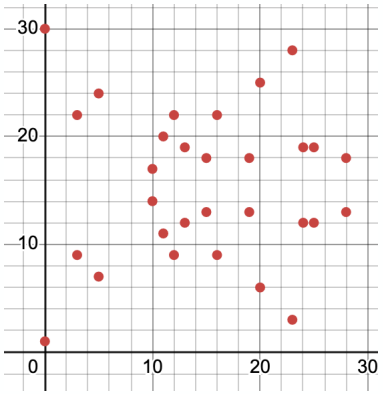
As with exponentiation, we need to adapt these ideas to operate on discrete values

Recall: Diffie-Hellman, RSA, and DSA all use modular arithmetic to restrict values to be integers within a finite range

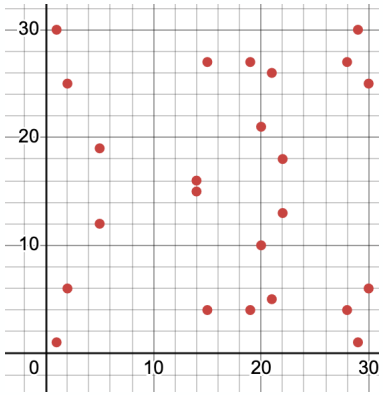
We can do the same with elliptic curves! Two approaches:

- Use a large prime modules ("prime curve")
- Use a modulus of the form 2^n ("binary curve")

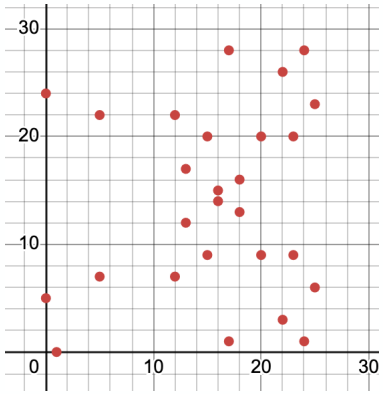
Examples with small values:



$y^2 \equiv x^3 - 3x + 1 \pmod{31}$



$y^2 \equiv x^3 - 3x + 3 \pmod{31}$



$y^2 \equiv x^3 + 3x - 4 \pmod{29}$

Not all elliptic curves are equally useful for cryptography

Curves of the form $y^2 = x^3 + ax + b$ are typical because they balance security and efficiency

- If $\langle x, y \rangle$ is on the curve, then $\langle x, -y \rangle$ is too
  - And there is no other $\langle x, y' \rangle$ on the curve
- Thus, we can specify the x (between 0 and modulus m) and the sign of y
  - This is compact and still uniquely identifies the point

Some example standard curves:

- P-256, $y^2 = x^3 + ax + b$
  - a = 0xffffffff00000001000000000000000000000000fffffffffffffffffffffffc
  - b = 0x5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b
  - p = 0xffffffff00000001000000000000000000000000ffffffffffffffffffffffff
- Curve25519, $y^2 = x^3 + 486662x^2 + x$
  - p = $2^{255} - 19$

A curve defines a sets of values (points); next we must define operations on points

Analogous to multiplying numbers in discrete log, we need a way to "multiply" two points on the curve

- I will use multiplication notation for this operation

We need our operation to have the following properties:

- Closure: If a and b are on the curve, so is $a \times b$
- Commutativity: $a \times b = b \times a$
- Associativity: $(a \times b) \times c = a \times (b \times c)$
- Identity: There is an element (call it $\mathbf{1}$) where $a \times \mathbf{1} = \mathbf{1} \times a = a$
  - We'll define a special element, the point at infinity, to act as $\mathbf{1}$
- Inverses: Each element a has an inverse a^{-1} where $a \times a^{-1} = \mathbf{1}$
  - We'll use vertical reflection

To provide security, the result needs to make "discrete log" hard

- i.e., it should be hard to get x from g^x and g
- This in ECDLP: Elliptic Curve Discrete Logarithm Problem

Defining a multiplication operation for points on an elliptic curve

Note: Some resources call this point addition. This is a change in notation and terminology only. We will continue to call it point multiplication.

Consider $\langle x_1, y_1 \rangle \times \langle x_2, y_2 \rangle = \langle x_3, y_3 \rangle$

if $x_1 \neq x_2$:

- $x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$, and $y_3 = \frac{y_2 - y_1}{x_2 - x_1} \times (x_1 - x_3) - y_1$

if $x_1 = x_2$ and $y_1 = -y_2$:

- These points are inverses, so $\langle x_1, y_1 \rangle \times \langle x_2, y_2 \rangle = \mathbf{1}$

if $x_1 = x_2$ and $y_1 = y_2$:

- $x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1$, where a is from $y^2 = x^3 + ax + b$
- $y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right) \times (x_1 - x_3) - y_1$

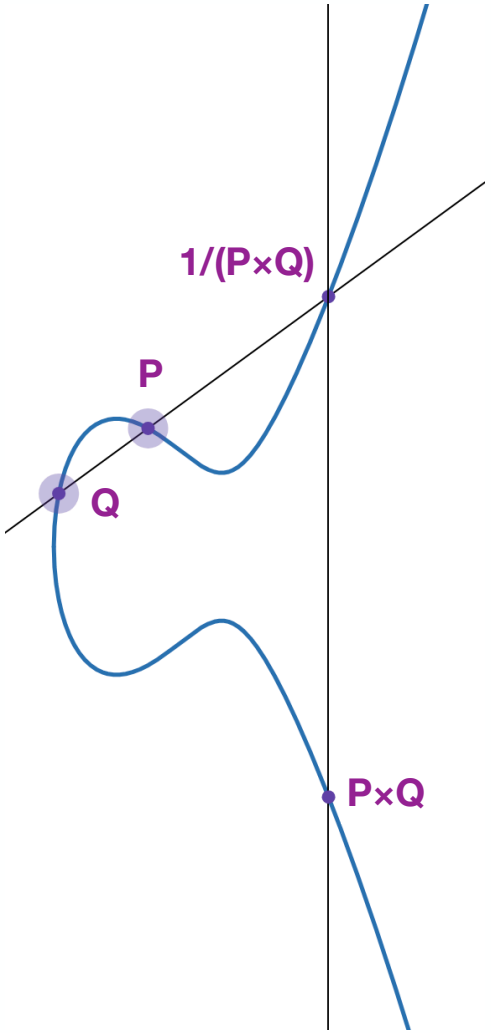
An illustration to help us understand point multiplication more intuitively

To multiply two distinct points:

- Draw the line that goes through both
- Identify the third point where this line intersects the curve
- Invert that third point (reflect vertically)

Some issues with this interpretation:

- What if the two points are inverses?
- What if one of the points is $\mathbf{1}$?
  - Same solution for both: Since there is no true third point, we use the point at infinity, $\mathbf{1}$
- What if the points are equal?
  - Draw the line as the tangent to the curve
- What about the discrete scenario?
  - … This figure unfortunately doesn't generalize from reals to modular arithmetic
    - But the formula still "works"!



A few other practical details…

Let's use exponentiation notation to describe repeated point multiplication

- This looks nothing like integer multiplication…
- … but since it's associative, we can use repeated "squaring" to compute results, as before

We still need to select the generator, g

- In Diffie-Hellman groups, this is an integer where $g^k \bmod p$ could generate all integers less than p
- In ECC, we choose a point and all "reachable" points are of the form g^k
- Unlike in DH, this may not generate all possible values
  - … especially for binary curves
  - We refer to the set of reachable points as the subgroup
  - Subgroups with prime order (cardinality) are most attractive
  - When a subgroup can reach only $\frac{1}{h}$ of the points, h is the co-factor

Elliptic-curve discrete log problem

We finally have all the pieces to define the elliptic-curve discrete log problem (ECDLP)!

ECDLP: Let E be an elliptic curve over a finite field, and let g be a point on E of order n. Given another point p on the curve such that $p = g^k$ for some integer $k \in \{0, 1, \ldots, n-1\}$, determine k.

Does this problem satisfy our original requirements?

- It is easy to compute g^k given g and k, so it's in NP
- The best known algorithms that work on all types of curves are $O(\sqrt{n})$, which is exponential in the size of the values
  - For 128-bit security, we need the order to be 2^{256}—compare to 2^{3072} for regular DLP

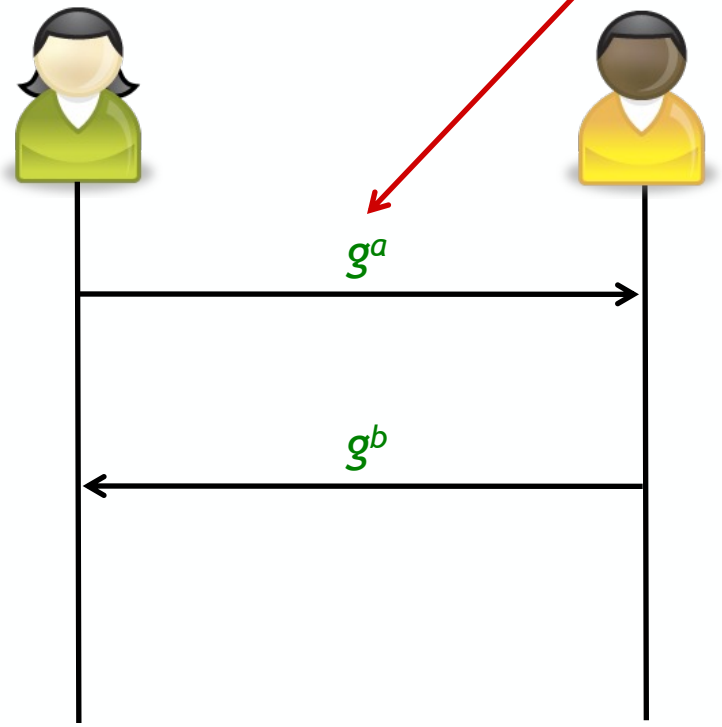
ECDLP is what we've been looking for!

How does elliptic-curve Diffie-Hellman protocol work?

Step 0: Alice and Bob agree on a finite cyclic group G of (large) prime order n, and a generator g for this group. This information is all public.

a is Alice's private key

g^a is Alice's public key



Step 1:

- Randomly choose $a \in \{1, 2, \ldots, n\text{-}1\}$
- Compute g^a
- Send g^a

Step 2:

- Randomly choose $b \in \{1, 2, \ldots, n\text{-}1\}$
- Compute g^b
- Send g^b

Step 3:

- Compute $(g^b)^a = g^{ba} = K_{ab}$

Step 3':

- Compute $(g^a)^b = g^{ab} = K_{ab}$

What's new/different?

Elliptic-Curve Digital Signature Algorithm (ECDSA)

ECDSA uses ECDLP rather than regular DLP

- Let the group include n points that can be expressed as g^k for $k \in \{0,1,2,\ldots,n-1\}$
- A private key is a randomly-chosen integer $2 \leq a \leq n-1$
- The corresponding public key is $A = g^a$

Signing message m:

- $M = \text{hash}(m)$, and t is a per-message secret
- $T = g^t$ (x-coordinate only)
- $S = (M + aT) * t^{-1} \bmod n$
- The signature is $\langle T, S \rangle$
  - This is two integers, not an EC point

Verification:

- Compute $x = M * S^{-1} \bmod n$, $y = T * S^{-1} \bmod n$
- Verify that T is the x-coordinate of $g^X * A^y$

Choosing good curve parameters is hard!

Many curve forms have been proposed: Hessian, Montgomery, (twisted) Edwards, Koblitz…

In 1999, NIST published recommendations for 15 curves:

- 5 prime curves
- 10 binary curves from 2 categories

Their choices have sometimes been criticized for prioritizing efficiency over security

- While they are secure mathematically, they can leak information in practice
- "Safe curves" such as Curve25519 and Curve448 are designed to avoid implementation issues

Some classes of curves are weak and should be avoided

- These have properties that allow us to solve ECDLP more efficiently

Choosing good curve parameters is hard!

In early 2000s, NSA pushed behind-the-scenes to have Dual_EC_DRBG standardized as an RNG option

- ANSI noticed and discussed a possible backdoor depending on how its parameters were generated
- The scheme was standardized and added to common security toolkits
- In 2013, Snowden leaks showed NSA plans "to covertly introduce weaknesses into the encryption standards followed by hardware and software developers around the world" and confirmed that Dual EC DRBG was backdoored
  - NSA paid RSA Security $10 M to include it in BSAFE crypto library

While the issues with this scheme don't seem to extend to other uses of the same curve, they cause distrust

- To combat the fear that curve parameters are chosen to introduce backdoors or other weaknesses, some curves use nothing up my sleeve techniques

Some bad news about elliptic-curve cryptography…

ECC is breakable with Shor's algorithm on a quantum computer

Researchers have estimated how many qubits and Toffoli gates are needed to break the scheme

- For a curve with 256-bit modulus (128-bit security), about 2330 qubits and 126 G Toffoli gates
- For comparison, 2048-bit RSA (112-bit security) would require 4098 qubits and 5.2 T Toffoli gates

This suggests that ECC will be broken by quantum computers sooner than DLP schemes such as RSA and DH

- We'll talk much more about quantum computing and post-quantum cryptography soon

Conclusions

Public-key cryptography based on the discrete log problem can be inefficient

- Solving DLP is sub-exponential, so key sizes grow quickly

Elliptic-curve mathematics enable a new type of discrete log

- We defined finite fields based on EC points and point multiplication
- The best known algorithms for ECDLP are exponential, allowing for smaller keys and more efficient runtimes

ECDLP can be used almost as a drop-in replacement for DLP in DSA and DH

- … forming ECDSA and ECDH

Choosing secure and efficient curves is non-trivial

Next: Quantum computing