ECDSA gem

https://github.com/DavidEGrayson/ruby_ecdsa

David Grayson
Las Vegas Ruby Meetup
2014-04-09

ECDSA

- Elliptic Curve Digital Signature Algorithm

Elliptic Curve Space

- 2-dimensional
- A point has two coordinates, x and y
- x and y are integers between 0 and *p-1,* where *p* is a pre-defined large prime

Elliptic Curve

- Curve equation, modulo prime *p*:

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

Elliptic Curve Group

- Need to define addition of points
- Need to pick a generator point *G*
- The curve group is made up of these points:

$$\infty, G, G+G, G+G+G, \ldots$$

- The words “group” and “curve” are often used interchangeably in ECDSA.

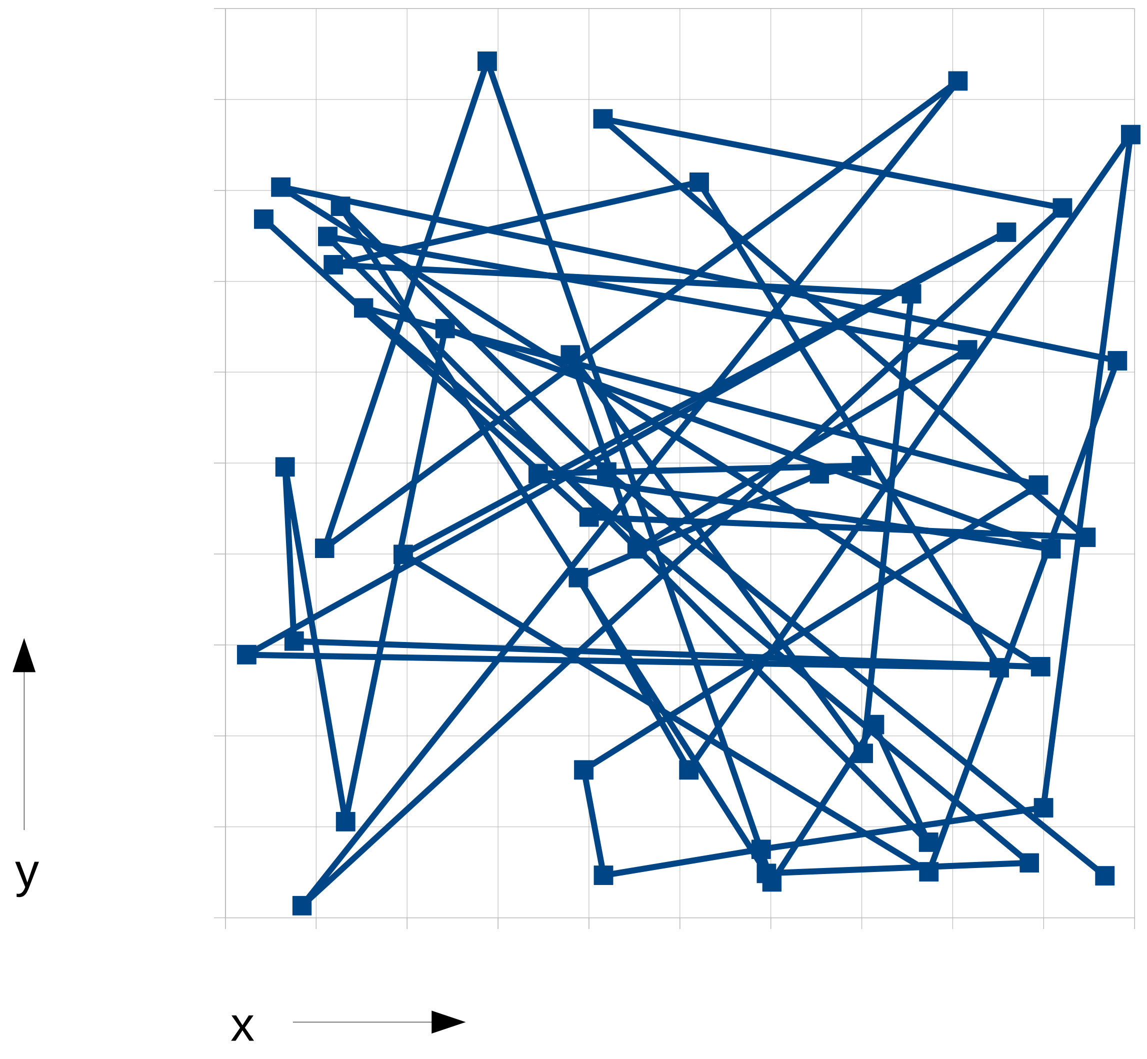
These parameters define an Elliptic curve:

- *p*: prime that defines the range of the coords
- *a*: coefficient in the curve equation
- *b*: coefficient in the curve equation
- *G*: the generator point

Example curve parameters

```
module ECDSA
  class Group
    Secp112r1 = new(
      name: 'secp112r1',
      p: 0xDB7C_2ABF62E3_5E668076_BEAD208B,
      a: 0xDB7C_2ABF62E3_5E668076_BEAD2088,
      b: 0x659E_F8BA0439_16EEDE89_11702B22,
      g: [0x0948_7239995A_5EE76B55_F9C2F098,
          0xA89C_E5AF8724_C0A23E0E_0FF77500],
      n: 0xDB7C_2ABF62E3_5E7628DF_AC6561C5,
      h: 1,
    )
  end
end
```

secp112r1 curve visualization



Private key

- A private key is just an integer *d*

```
require 'ecdsa'
require 'securerandom'
group = ECDSA::Group::Secp256k1
private_key = 1 + SecureRandom.random_number(group.order - 1)
puts 'private key: %#x' % private_key
```

- The public key is *G* added to itself *d* times.

```
public_key = group.generator.multiply_by_scalar(private_key)
puts 'public key: '
puts '  x: %#x' % public_key.x
puts '  y: %#x' % public_key.y
```

Signing and verifying

```
require 'digest/sha2'
message = 'ECDSA is cool.'
digest = Digest::SHA2.digest(message)
signature = nil
while signature.nil?
  temp_key = 1 + SecureRandom.random_number(group.order - 1)
  signature = ECDSA.sign(group, private_key, digest, temp_key)
end
puts 'signature: '
puts '  r: %#x' % signature.r
puts '  s: %#x' % signature.s
```

```
ECDSA.valid_signature?(public_key, digest, signature)
```

ECDSA gem design decisions

- Value education/experimentation over efficiency
  - Cryptographic code, not cryptic
  - Avoid OpenSSL as much as possible
- No randomness
  - Let the user choose a random number generator
- Separation of concerns between cryptography and binary data formatting
  - Cryptographic classes work with Ruby integers

https://github.com/DavidEGrayson/ruby_ecdsa