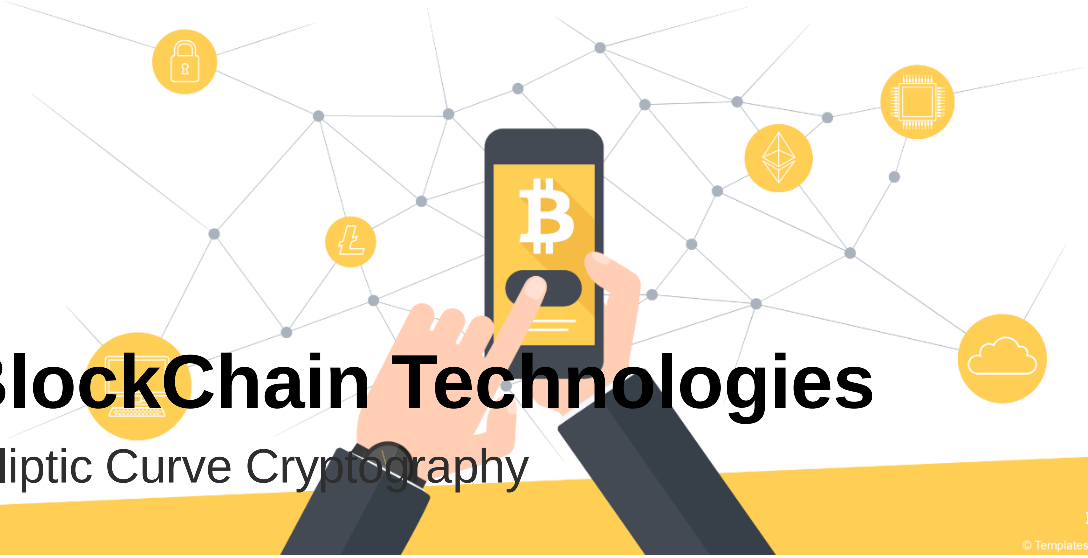


BlockChain Technologies

Elliptic Curve Cryptography

WHAT'S WRONG WITH RSA?

RSA is based upon the ‘belief’ that factoring is ‘difficult’ – never been proven

Prime numbers are getting too large

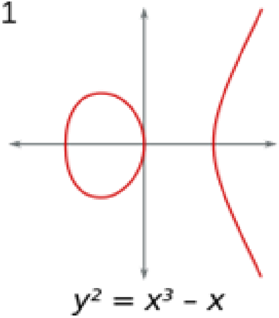
ELLIPTIC CURVE CRYPTOGRAPHY

➢ General mathematical form (Weierstraus equation):

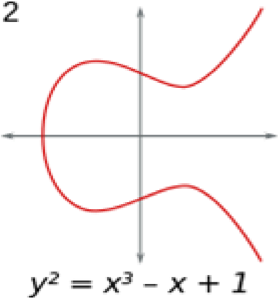
$$y^2 = x^3 + \boldsymbol{a}x + \boldsymbol{b}$$

for some $\boldsymbol{a}, \boldsymbol{b}$ (curve parameters)

1



2



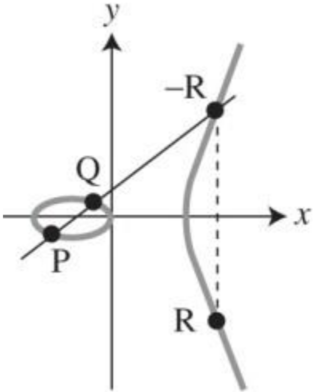
ELLIPTIC CURVE ENCRYPTION

- Encryption: Transforming points on curve (P, K_{PU}) into other point on same curve (C)

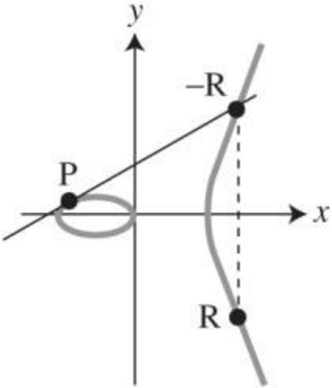
- Main idea (Abelian group): Need a definition of "+" so that "sum" of two points on a curve is also on the same curve:
  - $R = P + Q$ where $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$, $R = (x_R, y_R)$

- $R =$ "0" (additive identity)
  - Point at infinity: ∞
  - $0 = -0$
  - $P + (-P) = 0$

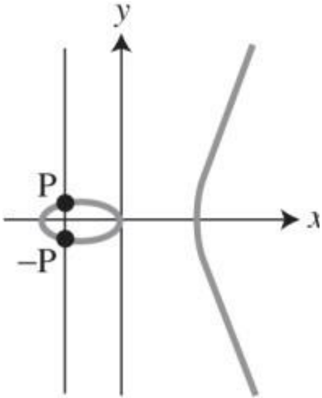
ELLIPTIC CURVE ADDITION CASES



Case 1: $P \neq Q$
$(x_P \neq x_Q, y_P \neq y_Q)$

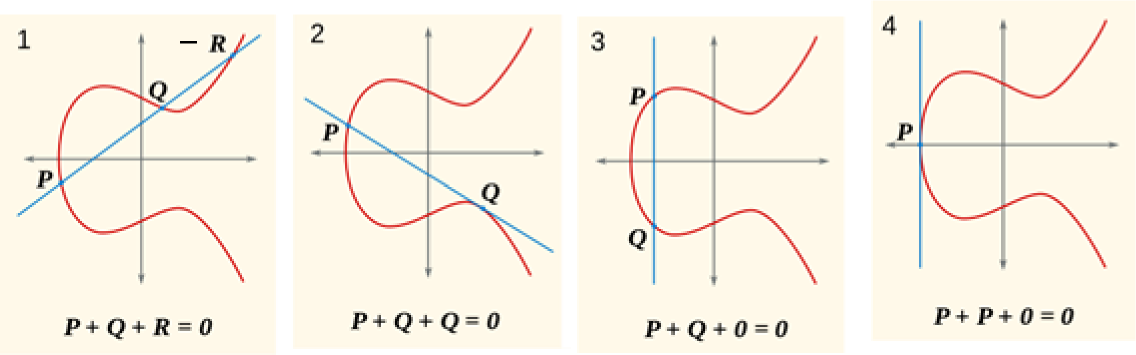


Case 2: $P = Q$



Case 3: $P = -Q$
$(x_P = x_Q, y_P \neq y_Q)$

ELLIPTIC CURVE ADDITION



- Equations for $P \neq Q$ (case 1):

$$\Delta = (y_Q - y_P)/(x_Q - x_P)$$
$$x_R = \Delta^2 - x_P - x_Q$$
$$y_R = \Delta(x_P - x_R) - y_P$$

Elliptic Curves over Zp

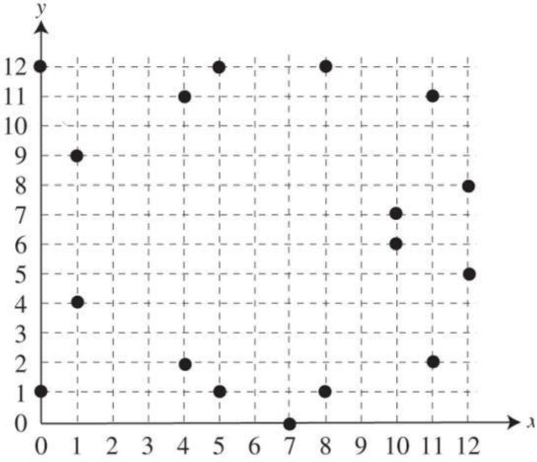
- Encryption requires modular arithmetic
  - Must be difficult to recover original points from **R**.
  - Modular arithmetic prevents "working backward", as in RSA
- Define "curve" as $E_p(a, b)$ where p is the modulus, a, b are the coefficients of $y^2 = x^3 + ax + b$
- Looking for (x, y) such that $y^2 = (x^3 + ax + b)$ mod p
  - Note: "points" on curve are integers
  - Example ($a = b = 1\ ,\ p = 13$): $x = 0 \rightarrow y^2 \bmod 13 = 1 \bmod 13$
  - $y = \pm 1 \bmod 13 \rightarrow y = 1, 12$
  - Two points: (0,1) and (0,12)

FINDING POINTS ON A Zp CURVE

- Points on elliptic curve $y^2 = x^3 + x + 1$ over p(13):

(0, 1)	(0, 12)
(1, 4)	(1, 9)
(4, 2)	(4, 11)
(5, 1)	(5, 12)
(7, 0)	(7, 0)
(8, 1)	(8, 12)
(10, 6)	(10, 7)
(11, 2)	(11, 11)

Points



Graph

EXAMPLE

- Let's examine the following elliptic curve as an example:

$$y^2 = x^3 + x + 6 \text{ over } \mathbb{Z}_{11}$$

X	0	1	2	3	4	5	6	7	8	9	10
$x^3 + x + 6$ mod 11	6	8	5	3	8	4	8	4	9	7	4
Y			4,7	5,6		2,9		2,9	3,8		2,9

ELLIPTIC CURVE MATHEMATICS

- Computing $(x_R, y_R) = (x_P, y_P) + (x_Q, y_Q)$
  - Necessary to turn two points corresponding to key and plaintext into point corresponding to ciphertext

- Use same rules for "+" as curves in space

- Main ideas:
  - Addition/subtraction/multiplication in mod p
  - Division = multiplication by inverse mod p

EXAMPLE: (4, 2) + (10, 6) ON E13(1, 1)

- step 1: compute $\Delta = (y_Q - y_P) / (x_Q - x_P)$

  $\Delta = (6 - 2) \times (10 - 4)^{-1} \bmod 13$

  - $= 4 \times 6\text{-}1 \bmod 13$ ($6^{-1} \bmod 13 = 11$)
  - $= 4 \times 11 \bmod 13 = 5$

13 = 2*6 + 1
1 = 13 – 2*6
-2 mod 13 =
11

- step 2: compute $x_R = \Delta^2 - x_P - x_Q$
  - xR = (25 – 4 – 10) mod 13 = 11
- step 3: compute $y_R = \Delta(x_P - x_R) - y_P$
  - $y_R = (5 * (4 - 11) - 2\ ) \bmod 13 = \mathbf{2}$

(4, 2) + (10, 6) = (11, 2) → note: also on curve!

MULTIPLICATION ON AN ELLIPTIC CURVE

- Multiplication = addition several times
  - Necessary for some forms of elliptic curve cryptography
  - Must use formula where P = Q for first addition

- Example: 3×(1, 4) on $E_{13}(1, 1)$
  - 3 x (1, 4) = (1, 4) + ((1, 4) + (1, 4)) = (1, 4) + (8, 12) = **(0,12)**
- Elliptic curve encryption is generally based on using multiplication on elliptic curves in place of exponentiation in existing public key algorithm.

$$g^k \rightarrow k \times G$$

DIFFIE-HELLMAN KEY AGREEMENT

Alice | **Bob**

Alice selects random α $\xrightarrow{g^{\alpha} \bmod p}$

$\xleftarrow{g^{\beta} \bmod p}$ Bob selects random β

Alice computes $\left(g^{\beta}\right)^{\alpha} = g^{\alpha\beta} \bmod p$ as the shared key (**session key**)

Bob computes $\left(g^{\alpha}\right)^{\beta} = g^{\alpha\beta} \bmod p$ as the shared key (**session key**)

ELLIPTIC CURVE DIFFIE-HELLMAN

- Alice and Bob agree on global parameters:
  - $\boldsymbol{E_p(a,b)}$: Elliptic curve mod $\boldsymbol{p}$ (prime) with parameters $\boldsymbol{a}$ and $\boldsymbol{b}$
  - $\boldsymbol{G}$: “Generator” point on that elliptic curve
    - For all points $\boldsymbol{R}$ on the curve, there exists some $\boldsymbol{n}$ such that $\boldsymbol{n} \times \boldsymbol{G} = \boldsymbol{R}$
      - Example: $\boldsymbol{P} = 211, \boldsymbol{E_p}(0, -4)$: the curve $y^2 = x^3 - 4$, $\boldsymbol{G} = (2, 2)$

- Alice and Bob select own private $\boldsymbol{x}$ and $\boldsymbol{y}$
- They each generate a public $\boldsymbol{R_1}$ and $\boldsymbol{R_2}$ as: $\boldsymbol{R_1} = \boldsymbol{x} \times \boldsymbol{G}$ and $\boldsymbol{R_2} = \boldsymbol{y} \times \boldsymbol{G}$
- They exchange these values

EXAMPLE

- Let's examine the following elliptic curve as an example:

$$y^2 = x^3 + x + 6 \text{ over } \mathbb{Z}_{11}$$

X	0	1	2	3	4	5	6	7	8	9	10
$x^3 + x + 6$ mod 11	6	8	5	3	8	4	8	4	9	7	4
Y			4,7	5,6		2,9		2,9	3,8		2,9

THE GROUP

$y^2 = x^3 + x + 6$ over $\mathbb{Z}11$

We can generate this by using the rules of addition we defined earlier where $2\alpha = \alpha + \alpha$

$\boldsymbol{G} = (2,7)$	$2\,\boldsymbol{G} = (5,2)$	$3\,\boldsymbol{G} = (8,3)$
$4\,\boldsymbol{G} = (10,2)$	$5\,\boldsymbol{G} = (3,6)$	$6\,\boldsymbol{G} = (7,9)$
$7\,\boldsymbol{G} = (7,2)$	$8\,\boldsymbol{G} = (3,5)$	$9\,\boldsymbol{G} = (10,9)$
$10\,\boldsymbol{G} = (8,8)$	$11\,\boldsymbol{G} = (5,9)$	$12\boldsymbol{G} = (2,4)$

EXAMPLE

Example: $\boldsymbol{P} = 211$, $\boldsymbol{E_p}(0, -4)$: the curve $y^2 = x^3 - 4$, $\boldsymbol{G} = (2, 2)$

- $\boldsymbol{x} = 121 \rightarrow \boldsymbol{R_1} = 121 \times (2, 2) = (115, 48)$
- $\boldsymbol{y} = 203 \rightarrow \boldsymbol{R_2} = 203 \times (2, 2) = (130, 203)$

$\xrightarrow{(115,\ 48)}$

$\xleftarrow[(130,\ 203)]{}$

$$121 \times (130, 203) = 203 \times (115, 48) = (\mathbf{161}, \mathbf{69})$$

ELLIPTIC CURVE DIFFIE-HELLMAN

- Alice and Bob generate the same key $\boldsymbol{k}$

  Alice: $\boldsymbol{k} = \boldsymbol{R_2} \times \boldsymbol{x}$
  Bob: $\boldsymbol{k} = \boldsymbol{R_1} \times \boldsymbol{y}$

- Proof:

$$\boldsymbol{R_2} \times \boldsymbol{x} = (\boldsymbol{G} \times \boldsymbol{y}) \times \boldsymbol{x}$$
$$\boldsymbol{R_1} \times \boldsymbol{y} = (\boldsymbol{G} \times \boldsymbol{x}) \times \boldsymbol{y}$$

SAFE ELLIPTIC CURVES

- The Curve25519 function:
  - Uses the prime number $2^{255} - 19$
  - Uses the elliptic curve $y^2 = x^3 + 486662x^2 + x$
  - Starting in 2014, OpenSSH defaults to Curve25519-based ECDH.
- The NIST P-256 curve:
  - Uses a prime $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ chosen for efficiency
  - Uses curve shape $y^2 = x^3 - 3x + b$
  - The NIST's P curve constants led to concerns that the NSA had chosen values that gave them an advantage in factoring public keys.
  - Dual Elliptic Curve Deterministic Random Bit Generation (or Dual_EC_DRBG) is a NIST national standard, which had included a deliberate weakness in the algorithm and the recommended elliptic curve.
- See https://safecurves.cr.yp.to/ for a list of safe elliptic curves.

ECDSA

- Elliptic Curve Digital Signature Algorithm (ECDSA) is an update of DSA algorithm adapted to use elliptic curves.
- Bitcoin uses ECDSA over the standard elliptic curve sec256k1 which provides 128 bit of security:
  - The equation: $y^2 = x^3 + 7$
  - The prime: $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$
- While sec256k1 is a published standard, it is rarely used outside of Bitcoin
- Possible reason for choosing sec256k1:
  - It is often more than 30% faster than other curves if the implementation is sufficiently optimized.
  - It is less likely to have a backdoor.

SECURITY AND SPEED OF ECC

- Why is this secure?
  - Same type of inverse modular problem (elliptic curve discrete logarithm problem or ECDLP)
  - If we have: $(\boldsymbol{x_2}, \boldsymbol{y_2}) = \boldsymbol{d} \times (\boldsymbol{x_1}, \boldsymbol{y_1})$, there is no simple way to determine $\boldsymbol{d}$ from $(\boldsymbol{x_1}, \boldsymbol{y_1})$ and $(\boldsymbol{x_2}, \boldsymbol{y_2})$ without trying all possible values
  - Computationally secure as long as $\boldsymbol{p}$ large enough (e.g. 160 bits) to prevent exhaustive search
- Why is this fast?
  - Only uses addition and multiplication – no exponents!
  - Smaller key sizes
    - 160 bit ECC key equivalent to 1024 bit RSA key
- Widely used on smart cards.

USING ELLIPTIC CURVES IN CRYPTOGRAPHY

- The central part of any cryptosystem involving elliptic curves is the **elliptic group**.
- All public-key cryptosystems have some underlying mathematical operation.
  - RSA has exponentiation (raising the message or ciphertext to the public or private values)
  - ECC has point multiplication (repeated addition of two points).

GENERIC PROCEDURES OF ECC

- Both parties agree to some publicly-known data items
  - The **elliptic curve equation**
    - values of ***a*** and ***b***
    - prime, ***p***
  - The **elliptic group** computed from the elliptic curve equation
  - A **base point**, G, taken from the elliptic group
    - Similar to the generator used in current cryptosystems
- Each user generates their public/private key pair
  - Private Key = an integer, x_A, selected from the interval [1, p-1]
  - Public Key = product, Y_A, of private key and base point
    - (Y_A = Pm*G)

EXAMPLE

- Suppose Alice wants to send to Bob an encrypted message.
- Both agree on a base point, G.
- Alice and Bob create public/private keys.
  - Alice
    - Private Key = X_A
    - Public Key = $Y_A = X_A * G$
  - Bob
    - Private Key = X_B
    - Public Key = $Y_B = X_B * G$
- Alice takes plaintext message, M, and encodes it onto a point, P_M, from the elliptic group

EXAMPLE CONT.

- Alice chooses another random integer, k from the interval [1, p-1]
- The ciphertext is a pair of points
  - $P_C = [\ (kG), (P_M + kY_B)\ ]$
- To decrypt, Bob computes the product of the first point from P_C and his private key, b
  - $X_B * (kG)$
- Bob then takes this product and subtracts it from the second point from P_C
  - $(P_M + kY_B) – [X_B(kG)] = P_M + k(X_BG) – X_B(kB) = P_M$
- Bob then decodes P_M to get the message, M.

ENCRYPTION RULES

- $y^2 = x^3 + x + 6$ over $\mathbb{Z}13$
- Suppose we let $G = (2,7)$ and choose the private key to be $XA= 7$
- Then $Y_A = 7G = (7,2)$
- Encryption:

$$e_K(x,k) = (k(G), \mathbf{P_M} + k(YA))$$

$$e_K(x,k) = (k(2,7), \mathbf{P_M} + k(7,2)) ,$$

where $x \in E$ and $0 \leq k \leq 12$

DECRYPTION RULE

- Decryption:

$d_K(y_1,y_2) = y_2 - xAy_1$ =>x is private kay

$d_K(y_1,y_2) = y_2 - 7y_1$

USING THIS SCHEME

- Suppose Alice wants to send a message to Bob.
- Plaintext is $\mathbf{P_M}$ = (10,9) which is a point in *E*
- Choose a random value for k, k = 3
- So now calculate (y_1,y_2):
- $y_1 = 3(2,7) = (8,3)$
- $y_2 = (10,9) + 3(7,2) = (10,9) + (3,5) = (10,2)$
- Alice transmits y = ((8,3),(10,2))

BOB DECRYPTS

- Bob receives y = ((8,3),(10,2))
- Calculates

$$
\begin{aligned}
\mathbf{P_M} &= (10,2) - 7(8,3) \\
&= (10,2) - (3,5) \\
&= (10,2) + (3,6) \\
&= (10,9)
\end{aligned}
$$

Which was the plaintext