Elliptic Curve Cryptography in Bitcoin

Saravanan Vijayakumaran
sarva@ee.iitb.ac.in

Department of Electrical Engineering
Indian Institute of Technology Bombay

January 29, 2024

Some Context on Diophantine Equations

Diophantine Equations

- Polynomial equations with integer coefficients
- Solutions in the integers or rational numbers are of interest
- Named after Greek mathematician Diophantus of Alexandria, who lived before the 3rd century AD
- *Example:* Fermat's Last Theorem

  $$X^n + Y^n = Z^n$$

  where $n \in \mathbb{Z}, n \geq 3$

  - Has no solutions in the non-zero integers X, Y, Z
  - Stated in 17th century; proved in 1995 by Andrew Wiles

Bachet's Equation

- Consider the problem of writing an integer as the difference of a square and a cube

$$y^2 - x^3 = c \text{ for some fixed } c \in \mathbb{Z}$$

- In 1621, Bachet discovered a duplication formula
- If (x, y) is a solution in $\mathbb{Q}^2$ and $y \neq 0$, then the following is also a solution

$$\left(\frac{x^4 - 8cx}{4y^2}, \frac{-x^6 - 20cx^3 + 8c^2}{8y^3} \right)$$

- Later, it was proved that if $xy \neq 0$ and $c \notin \{1, -432\}$, then the duplication formula gives infinitely many distinct solutions
- It turns out that the duplication solution is the intersection of the tangent at (x, y) with the curve $y^2 - x^3 = c$
    - An instance of geometry being used to settle algebraic questions

Bivariate Diophantine Equations

- Consider the equation

$$f(x, y) = 0$$

  where f is a polynomial with rational coefficients
- Mathematicians were interested in the following questions
  1. Are there any solutions in the integers?
  2. Are there any solutions in the rational numbers?
  3. Are there infinitely many solutions in the integers?
  4. Are there infinitely many solutions in the rational numbers?
- The answers are easy for linear polynomials of the form

$$ax + by + c = 0$$

  where $a, b, c \in \mathbb{Z}$
  - No integer solutions if $\gcd(a, b) \nmid c$
  - Infinitely many integer solutions if $\gcd(a, b) \mid c$
  - Infinitely many rational solutions always exist

Rational Conics

- A rational conic is given by a quadratic equation

  $$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

  where the coefficients are rational numbers
- If the conic has one rational point $\mathcal{O}$, then it has infinitely many rational points
- Substituting the equation of a line passing through $\mathcal{O}$ into the conic gives a quadratic equation in x
- Since the quadratic has rational coefficients, the sum of its roots is rational
- Since the x-coordinate of $\mathcal{O}$ is one of the roots, the other root is also rational

Rational Cubics

- A rational cubic is given by a cubic equation

$$ax^3 + bx^2y + cxy^2 + dy^3 + ex^2 + fxy + gY^2 + hx + iy + j = 0$$

  where the coefficients are rational numbers
- We cannot use the technique we used for conics as a line generally meets a cubic in three points
- But if we have two rational points on a cubic, we can use the line joining them to find the third point
- In the 1920s, Siegel proved that cubic equation has only finitely many integer solutions
- In 1922, Mordell proved that a **non-singular** rational cubic curve has a finite set of rational points that "generate" all other rational points
  - The generation process involves drawing lines through points and considering intersections
- Elliptic curves are curves of genus one with a specified point
- Every elliptic curve can be specified by a cubic equation in the affine plane with specified point mapped to $[0, 1, 0]$

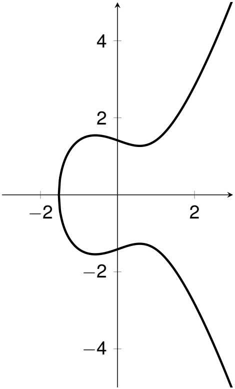
Elliptic Curves Over Real Numbers

Elliptic Curves over Reals

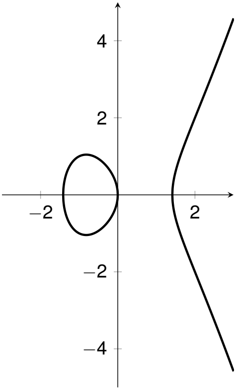
The set E of real solutions (x, y) of

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

along with a “point of infinity” $\mathcal{O}$. Here $4a^3 + 27b^2 \neq 0$.

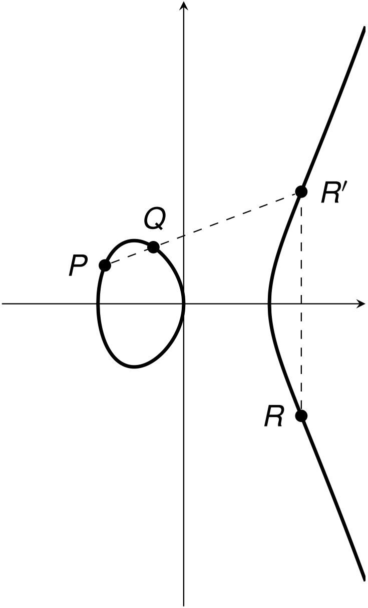


$y^2 = x^3 - x + 2$



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

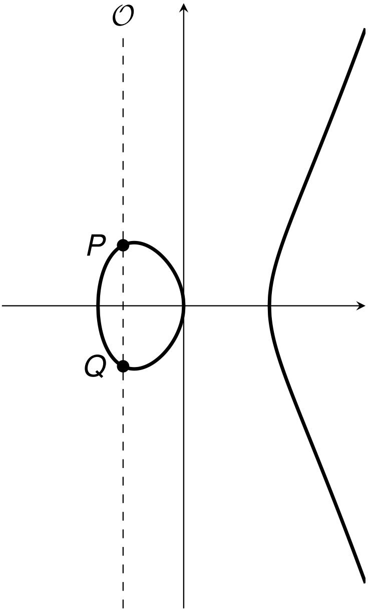
Point Addition (1/3)



$$P = (x_1, y_1), Q = (x_2, y_2)$$
$$x_1 \neq x_2$$
$$P + Q = R$$

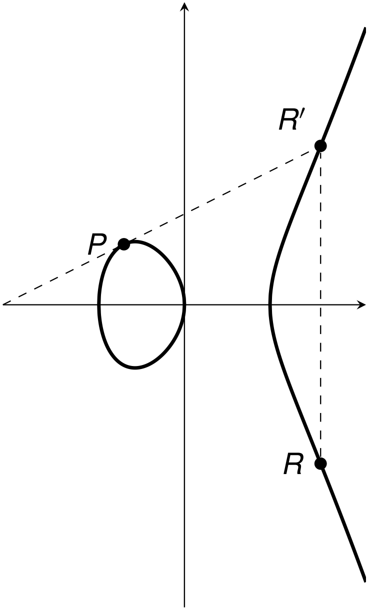
$$R = (x_3, y_3)$$
$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$$
$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$$

Point Addition (2/3)



$$P = (x_1, y_1), Q = (x_2, y_2)$$
$$x_1 = x_2, y_1 = -y_2$$
$$P + Q = \mathcal{O}$$

Point Addition (3/3)



$$P = (x_1, y_1), Q = (x_2, y_2)$$
$$x_1 = x_2, y_1 = y_2 \neq 0$$
$$P + Q = R$$

$$R = (x_3, y_3)$$
$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1$$
$$y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1$$

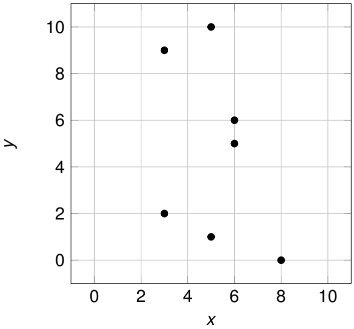
Elliptic Curves Over Finite Fields

Elliptic Curves over Finite Fields

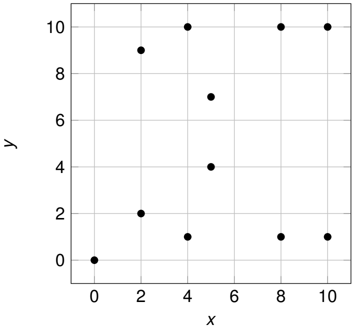
For char$(F) \neq 2, 3$, the set E of solutions (x, y) in F^2 of

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

along with a “point of infinity” $\mathcal{O}$. Here $4a^3 + 27b^2 \neq 0$.



$y^2 = x^3 + 10x + 2$ over $\mathbb{F}_{11}$



$y^2 = x^3 + 9x$ over $\mathbb{F}_{11}$

Point Addition for Finite Field Curves

- Point addition formulas derived for reals are used
- Example: $y^2 = x^3 + 10x + 2$ over $\mathbb{F}_{11}$

$+$	$\mathcal{O}$	$(3,2)$	$(3,9)$	$(5,1)$	$(5,10)$	$(6,5)$	$(6,6)$	$(8,0)$
$\mathcal{O}$	$\mathcal{O}$	$(3,2)$	$(3,9)$	$(5,1)$	$(5,10)$	$(6,5)$	$(6,6)$	$(8,0)$
$(3,2)$	$(3,2)$	$(6,6)$	$\mathcal{O}$	$(6,5)$	$(8,0)$	$(3,9)$	$(5,10)$	$(5,1)$
$(3,9)$	$(3,9)$	$\mathcal{O}$	$(6,5)$	$(8,0)$	$(6,6)$	$(5,1)$	$(3,2)$	$(5,10)$
$(5,1)$	$(5,1)$	$(6,5)$	$(8,0)$	$(6,6)$	$\mathcal{O}$	$(5,10)$	$(3,9)$	$(3,2)$
$(5,10)$	$(5,10)$	$(8,0)$	$(6,6)$	$\mathcal{O}$	$(6,5)$	$(3,2)$	$(5,1)$	$(3,9)$
$(6,5)$	$(6,5)$	$(3,9)$	$(5,1)$	$(5,10)$	$(3,2)$	$(8,0)$	$\mathcal{O}$	$(6,6)$
$(6,6)$	$(6,6)$	$(5,10)$	$(3,2)$	$(3,9)$	$(5,1)$	$\mathcal{O}$	$(8,0)$	$(6,5)$
$(8,0)$	$(8,0)$	$(5,1)$	$(5,10)$	$(3,2)$	$(3,9)$	$(6,6)$	$(6,5)$	$\mathcal{O}$

- The set $E \cup \mathcal{O}$ is closed under addition
- In fact, its a group

Bitcoin's Elliptic Curve: `secp256k1`

- $y^2 = x^3 + 7$ over $\mathbb{F}_p$ where

$$\begin{aligned} p &= \underbrace{\texttt{FFFFFFFF}\ \cdots\ \texttt{FFFFFFFF}}_{\text{48 hexadecimal digits}}\ \texttt{FFFFFFFE}\ \texttt{FFFFFC2F} \\ &= 2^{256} - 2^{32} - 2^{9} - 2^{8} - 2^{7} - 2^{6} - 2^{4} - 1 \end{aligned}$$

- $E \cup \mathcal{O}$ has cardinality n where

$$\begin{aligned} n = {} & \texttt{FFFFFFFF}\ \texttt{FFFFFFFF}\ \texttt{FFFFFFFF}\ \texttt{FFFFFFFE} \\ & \texttt{BAAEDCE6}\ \texttt{AF48A03B}\ \texttt{BFD25E8C}\ \texttt{D0364141} \end{aligned}$$

- Private key is $k \in \{1, 2, \ldots, n-1\}$
- Public key is kP where $P = (x, y)$

$$\begin{aligned} x ={} & \texttt{79BE667E}\ \texttt{F9DCBBAC}\ \texttt{55A06295}\ \texttt{CE870B07} \\ & \texttt{029BFCDB}\ \texttt{2DCE28D9}\ \texttt{59F2815B}\ \texttt{16F81798}, \\ y ={} & \texttt{483ADA77}\ \texttt{26A3C465}\ \texttt{5DA4FBFC}\ \texttt{0E1108A8} \\ & \texttt{FD17B448}\ \texttt{A6855419}\ \texttt{9C47D08F}\ \texttt{FB10D4B8}. \end{aligned}$$

Point Multiplication using Double-and-Add

- Point multiplication: kP calculation from k and P
- Let $k = k_0 + 2k_1 + 2^2k_2 + \cdots + 2^m k_m$ where $k_i \in \{0, 1\}$
- Double-and-Add algorithm
    - Set $N = P$ and $Q = \mathcal{O}$
    - for $i = 0, 1, \ldots, m$
        - if $k_i = 1$, set $Q \leftarrow Q + N$
        - Set $N \leftarrow 2N$
    - Return Q

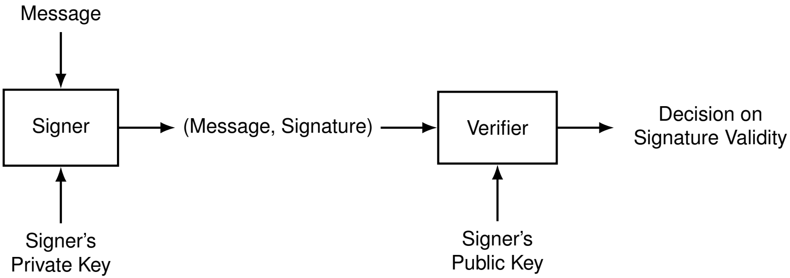
Why ECC?

- For elliptic curves $E(\mathbb{F}_q)$, best DL algorithms are exponential in $n = \lceil \log_2 q \rceil$
$$C_{EC}(n) = 2^{n/2}$$
- In $\mathbb{F}_p^*$, best DL algorithms are sub-exponential in $N = \lceil \log_2 p \rceil$
  - $L_p(v, c) = \exp\left(c(\log p)^v (\log\log p)^{(1-v)}\right)$ with $0 < v < 1$
- Using GNFS method, DLs can be found in $L_p(1/3, c_0)$ in $\mathbb{F}_p^*$
$$C_{CONV}(N) = \exp\left(c_0 N^{1/3} \left(\log\left(N \log 2\right)\right)^{2/3}\right)$$
- Best algorithms for factorization have same asymptotic complexity
- For similar security levels
$$n = \beta N^{1/3} \left(\log\left(N \log 2\right)\right)^{2/3}$$
- Key size in ECC grows slightly faster than cube root of conventional key size
  - 173 bits instead of 1024 bits, 373 bits instead of 4096 bits

Elliptic Curve Digital Signature Algorithm

Digital Signatures

- Digital signatures prove that the signer knows private key



Schnorr Identification Scheme

- Let G be a cyclic group of order q with generator g
- Identity corresponds to knowledge of private key x where $h = g^x$
- A prover wants to prove that she knows x to a verifier without revealing it
  1. Prover picks $k \leftarrow \mathbb{Z}_q$ and sends initial message $I = g^k$
  2. Verifier sends a challenge $r \leftarrow \mathbb{Z}_q$
  3. Prover sends $s = rx + k \bmod q$
  4. Verifier checks $g^s \cdot h^{-r} \overset{?}{=} I$
- Passive eavesdropping does not reveal x for uniform r
  - (I, r) is uniform on $G \times \mathbb{Z}_q$ and $s = \log_g(I \cdot h^r)$
  - Transcripts with same distribution can be simulated without knowing x
  - Choose r, s uniformly from $\mathbb{Z}_q$ and set $I = g^s \cdot h^{-r}$
- We can prove that a prover which generates correct proofs must know x by constructing an extractor for x
  - Section 19.1 of Boneh-Shoup

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**
  1. On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q$
  2. Sets $I := g^k$
  3. Computes $r := H(I, m)$
  4. Computes $s = rx + k \bmod q$
  5. Outputs (r, s) as signature for m
- **Verifier**
  1. On input m and (r, s)
  2. Compute $I := g^s \cdot h^{-r}$
  3. Signature valid if $H(I, m) \overset{?}{=} r$
- Example of Fiat-Shamir transform
- Patented by Claus Schnorr in 1988

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response
  5. Verifier accepts if $s \neq 0$ and

$$g^{\alpha s^{-1}} \cdot h^{rs^{-1}} \overset{?}{=} I$$

- Digital Signature Algorithm
  1. Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
  2. Let $F : G \mapsto \mathbb{Z}_q$ be a function, not necessarily CHF
  3. **Signer:**
     - 3.1 On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q^*$ and sets $r := F(g^k)$
     - 3.2 Computes $s := \left[k^{-1} \cdot (H(m) + xr)\right] \bmod q$
     - 3.3 If $r = 0$ or $s = 0$, choose k again
     - 3.4 Outputs (r, s) as signature for m
  4. **Verifier**
     - 4.1 On input m and (r, s) with $r \neq 0, s \neq 0$ checks

$$F\left(g^{H(m)s^{-1}} h^{rs^{-1}}\right) \overset{?}{=} r$$

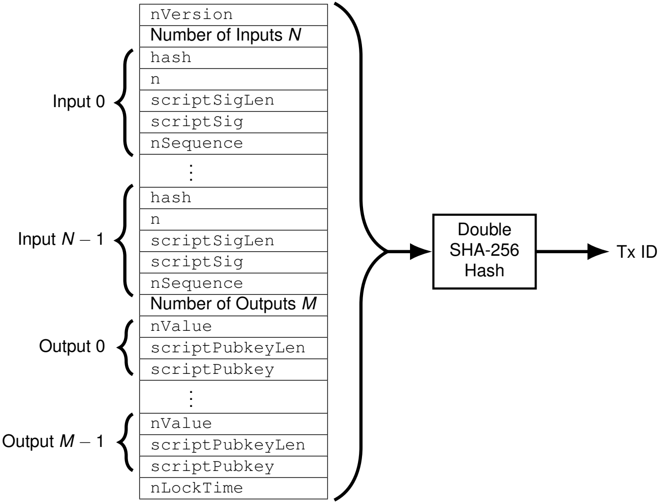
ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{F}_n^*$
  3. Compute $jP = (x, y)$
  4. Calculate $r = x \bmod n$. If $r = 0$, go to step 2.
  5. Calculate $s = j^{-1}(e + kr) \bmod n$. If $s = 0$, go to step 2.
  6. Output (r, s) as signature for m
- **Verifier:** Has public key kP, message m, and signature (r, s)
  1. Calculate $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Calculate $j_1 = es^{-1} \bmod n$ and $j_2 = rs^{-1} \bmod n$
  3. Calculate the point $Q = j_1 P + j_2(kP)$
  4. If $Q = \mathcal{O}$, then the signature is invalid.
  5. If $Q \neq \mathcal{O}$, then let $Q = (x, y) \in \mathbb{F}_p^2$. Calculate $t = x \bmod n$. If $t = r$, the signature is valid.
- As n is a 256-bit integer, signatures are 512 bits long
- As j is randomly chosen, ECDSA output is random for same m

Transaction Malleability

Transaction ID

Regular Transaction



Input 0

Input $N-1$

Output 0

Output $M-1$

`nVersion`

Number of Inputs N

`hash`

`n`

`scriptSigLen`

`scriptSig`

`nSequence`

⋮

`hash`

`n`

`scriptSigLen`

`scriptSig`

`nSequence`

Number of Outputs M

`nValue`

`scriptPubkeyLen`

`scriptPubkey`

⋮

`nValue`

`scriptPubkeyLen`

`scriptPubkey`

`nLockTime`

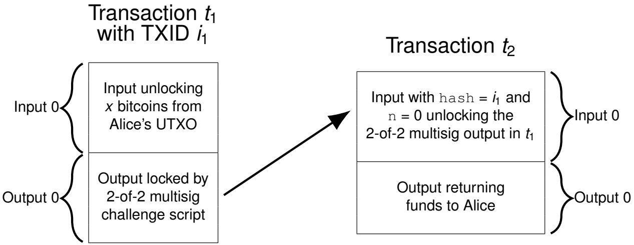
Double SHA-256 Hash

Tx ID

Refund Protocol

- Alice wants to teach Bob about transactions
- Bob does not own any bitcoins
- Alice decides to transfer some bitcoins to Bob
- Alice does not trust Bob
- She wants to ensure refund

Refund Protocol



Transaction t_1 with TXID i_1

Input 0

Input unlocking *x* bitcoins from Alice's UTXO

Output 0

Output locked by 2-of-2 multisig challenge script

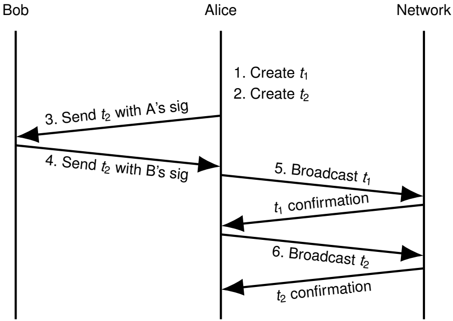
Transaction t_2

Input with `hash` = i_1 and `n` = 0 unlocking the 2-of-2 multisig output in t_1

Input 0

Output returning funds to Alice

Output 0



Bob

Alice

Network

1. Create t_1

2. Create t_2

3. Send t_2 with A's sig

4. Send t_2 with B's sig

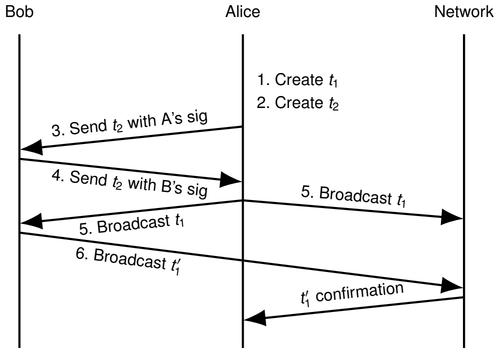
5. Broadcast t_1

t_1 confirmation

6. Broadcast t_2

t_2 confirmation

Exploiting Transaction Malleability



- If (r, s) is a valid ECDSA signature, so is $(r, n - s)$
- The t_1' transaction cannot be spent by t_2
- SegWit = Segregated Witness
  - Activated in August 2017
  - Solves problems arising from transaction malleability

References

- Chapter 1 of *Rational Points on Elliptic Curves*, Joseph H. Silverman, John T. Tate, 2nd Edition, 2015
- Sections 9.3 of *Introduction to Modern Cryptography*, J. Katz, Y. Lindell, 2nd edition
- Chapters 2, 5 of *An Introduction to Bitcoin*, S. Vijayakumaran, `www.ee.iitb.ac.in/~sarva/bitcoin.html`
- Section 19.1 of *A Graduate Course in Applied Cryptography*, D. Boneh, V. Shoup, `www.cryptobook.us`