Elliptic Curve Cryptography in Bitcoin

Saravanan Vijayakumaran
sarva@ee.iitb.ac.in

Department of Electrical Engineering
Indian Institute of Technology Bombay

January 23, 2026

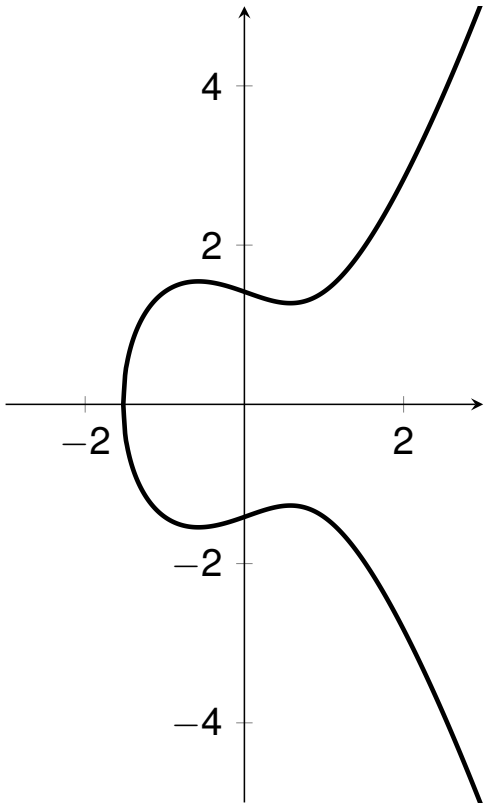
Elliptic Curves Over Real Numbers

Elliptic Curves over Reals

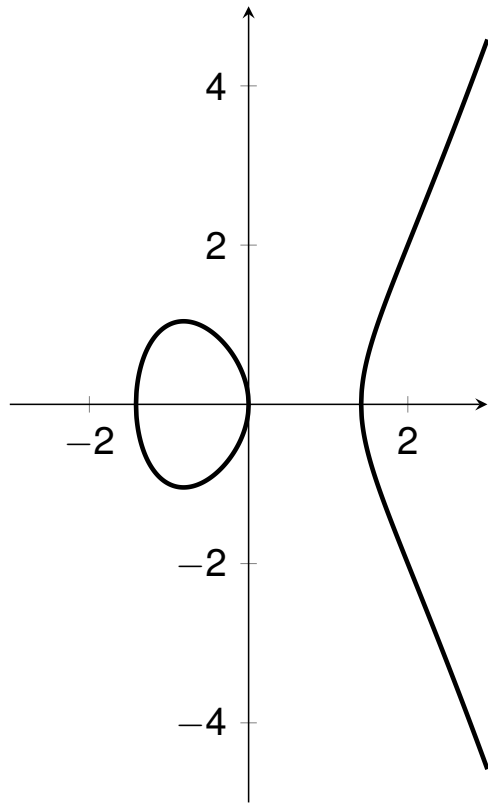
The set E of real solutions (x, y) of

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

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

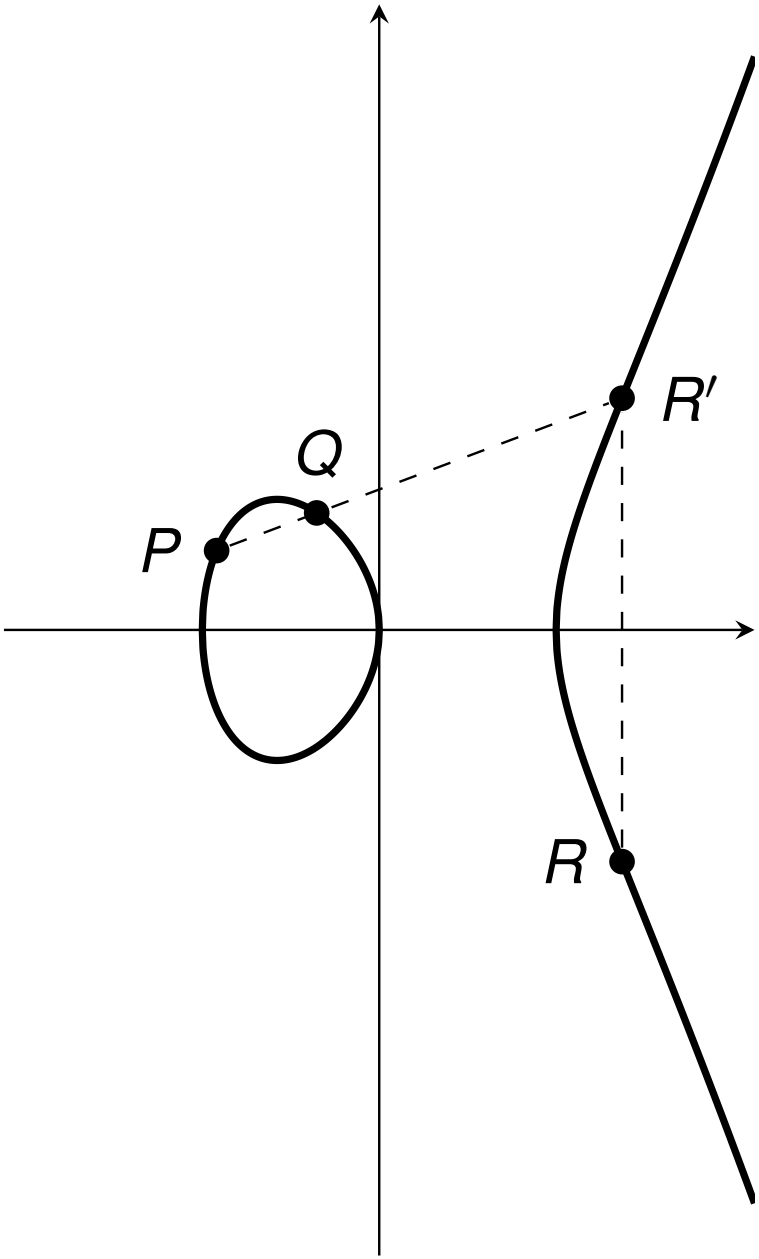


$y^2 = x^3 - x + 2$



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

Point Addition (1/3)



$$P = (x_1, y_1), Q = (x_2, y_2)$$

$$x_1 \neq x_2$$

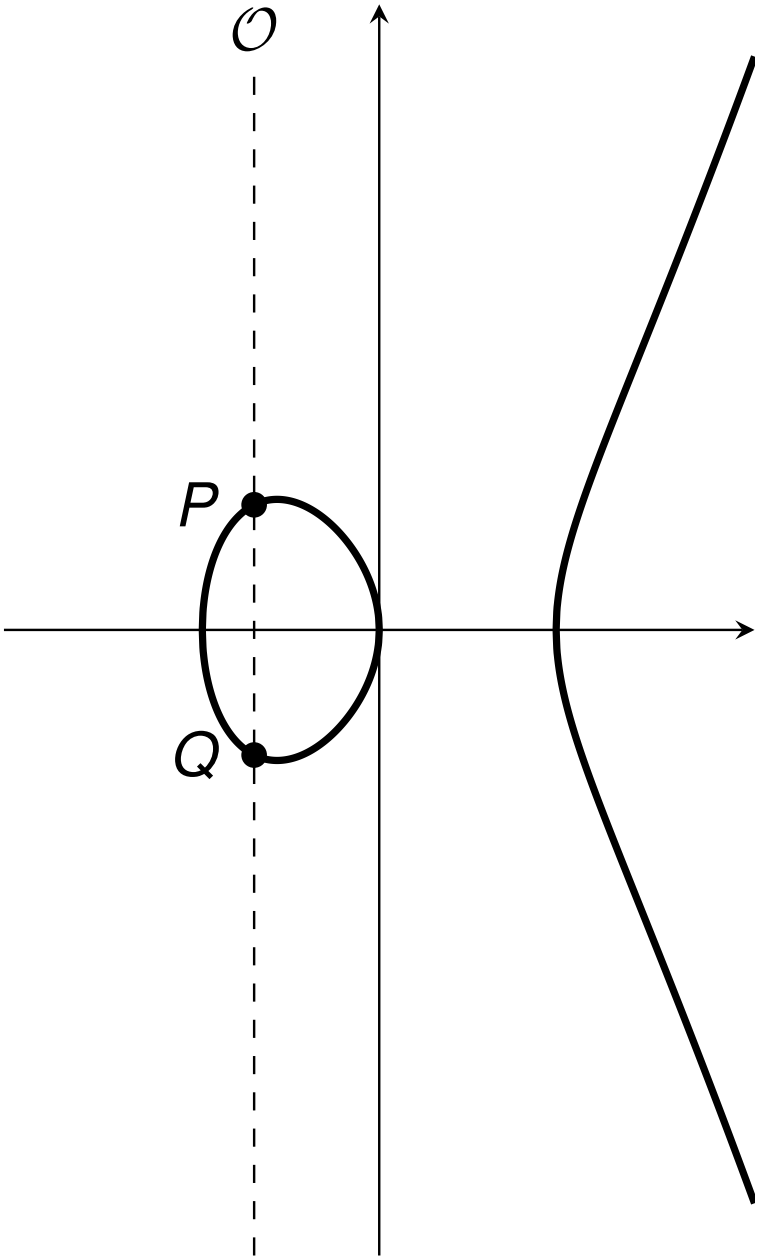
$$P + Q = R$$

$$R = (x_3, y_3)$$

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

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

Point Addition (2/3)

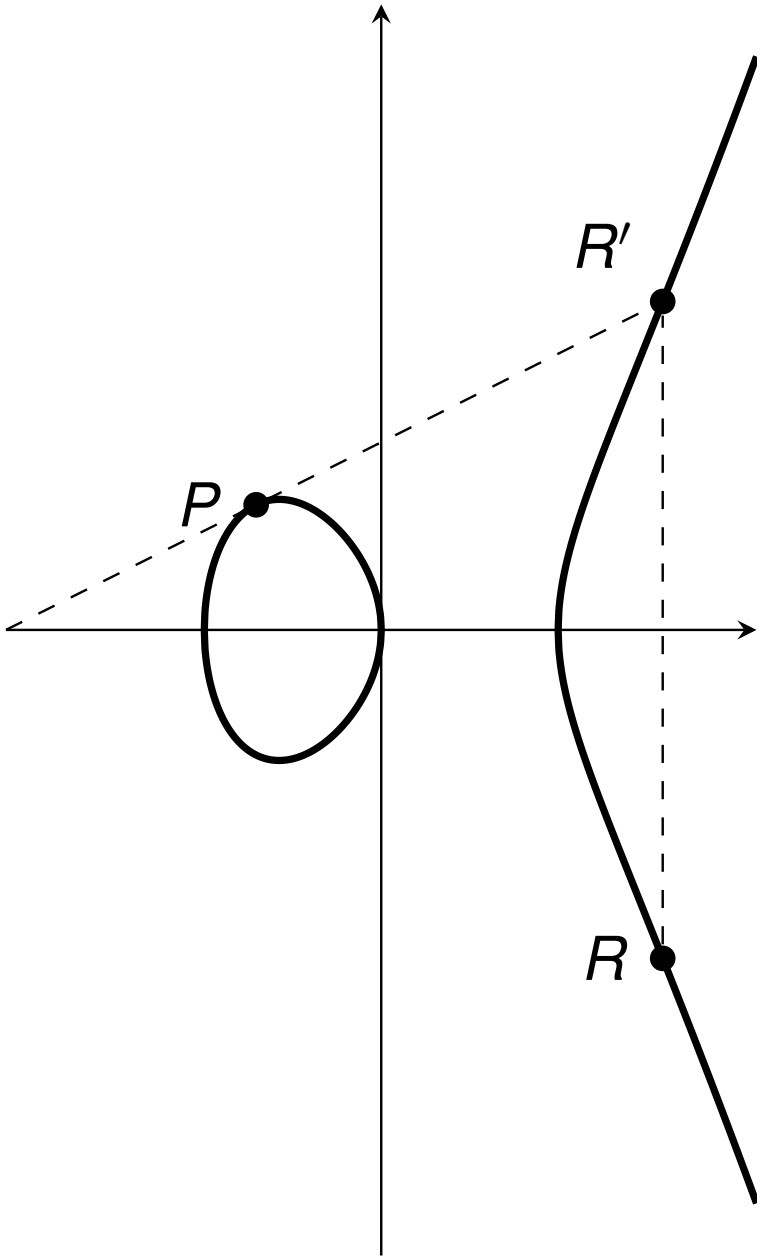


$$P = (x_1, y_1), Q = (x_2, y_2)$$

$$x_1 = x_2, y_1 = -y_2$$

$$P + Q = \mathcal{O}$$

Point Addition (3/3)



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

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

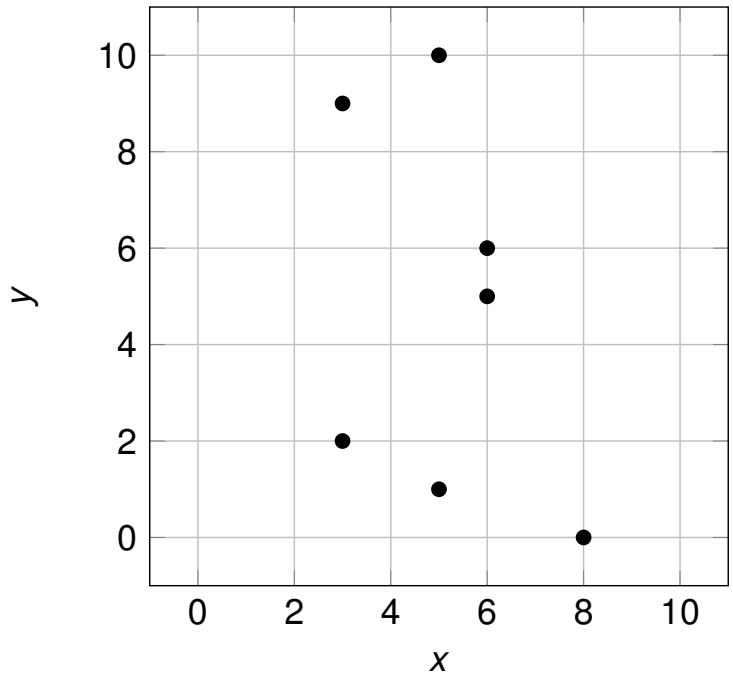
Elliptic Curves Over Finite Fields

Elliptic Curves over Finite Fields

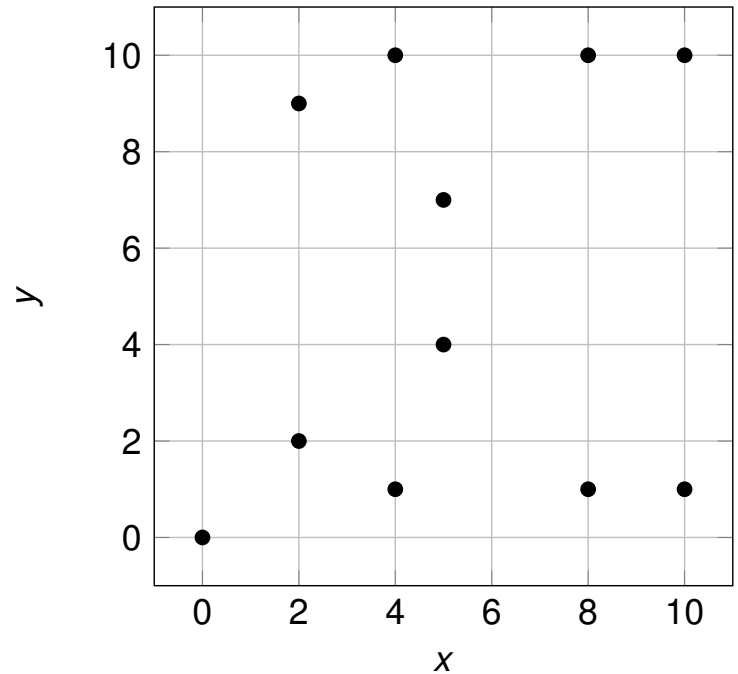
For $\mathrm{char}(F) \neq 2, 3$, the set E of solutions (x, y) in F^2 of

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

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

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

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

$$n = \texttt{FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE}$$
$$\texttt{BAAEDCE6 AF48A03B BFD25E8C D0364141}$$

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

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

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

The Discrete Logarithm Problem

- Given public key kP, finding private key k requires solving the discrete logarithm problem
- **Definition:** If G is a cyclic group of order q with generator g, then for $h \in G$ the unique $x \in \mathbb{Z}_q$ which satisfies $g^x = h$ is called the discrete logarithm of h with respect to g.
- DLP is hard in prime order subgroups of $\mathbb{F}_p^*$
- DLP is hard in some elliptic curve groups

Why ECC?

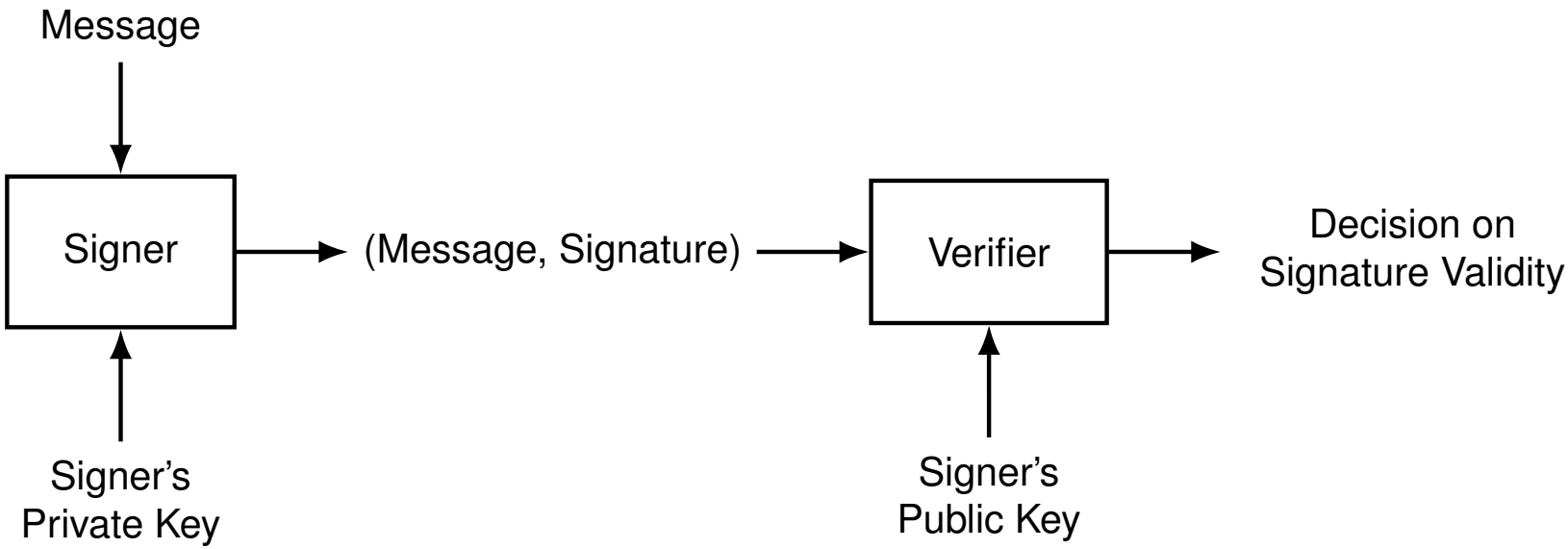
- Effective key length is a value n such that the best known attack takes $\mathcal{O}(2^n)$ time
- NIST recommended key lengths

Effective key length	**RSA Modulus** N	**Order** q **Subgroup of** $\mathbb{Z}_p^*$	**Elliptic curve group order** q
112	2048	p: 2048, q: 224	224
128	3072	p: 3072, q: 256	256
192	7680	p: 7680, q: 384	384
256	15360	p: 15360, q: 512	512

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 $g^s \cdot h^{-r} \stackrel{?}{=} I$
- 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 13.5 of Katz-Lindell

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 $H(I, m) \stackrel{?}{=} r$
- 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

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

Digital Signature Algorithm in $\mathbb{F}_p^*$

- Let g be the generator of a prime order cyclic subgroup of $\mathbb{F}_p^*$ of order q
  1. Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
  2. Let $F : \mathbb{F}_p^* \mapsto \mathbb{Z}_q$ be the function $F(x) = x \bmod q$.
  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

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

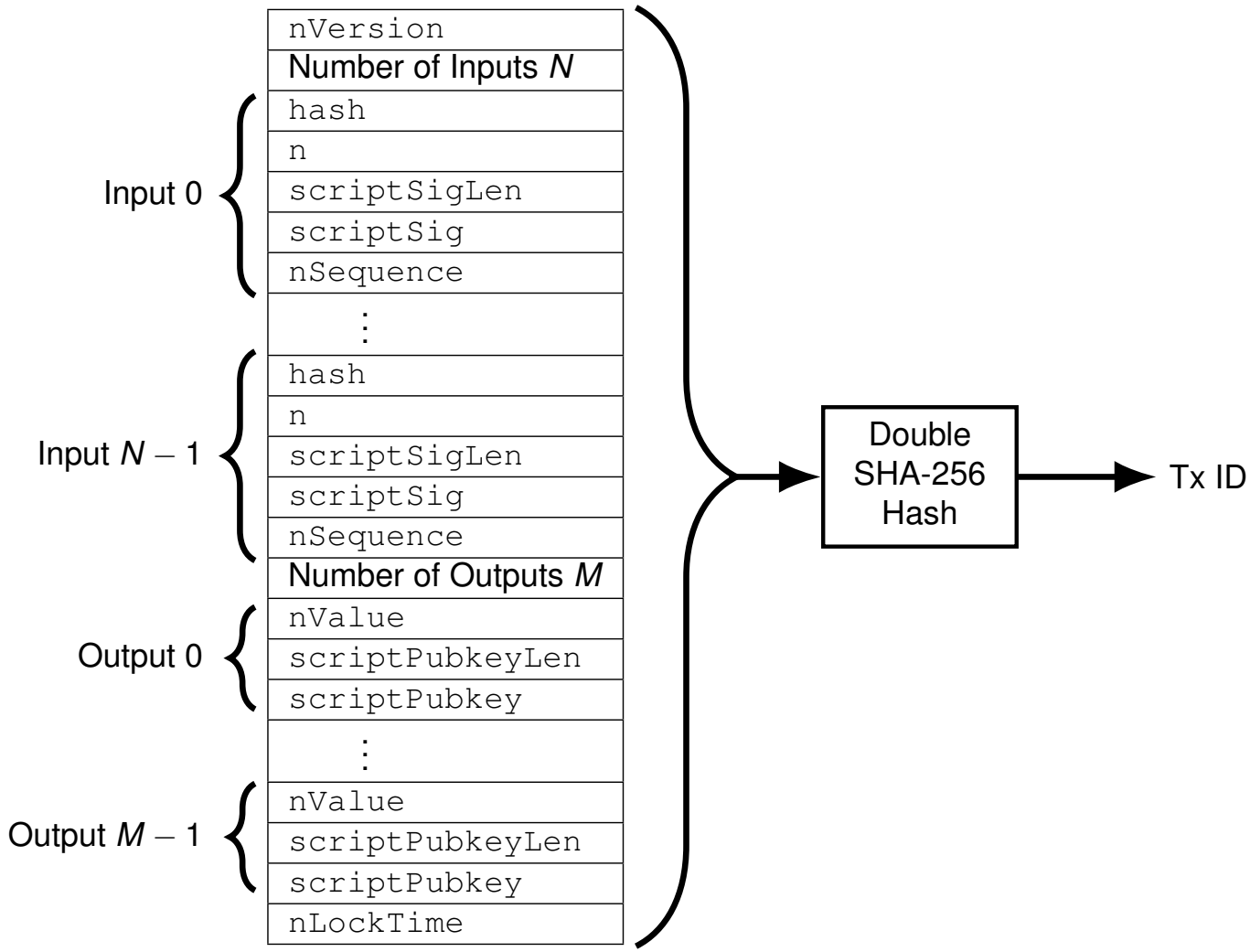
ECDSA in Bitcoin

- **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

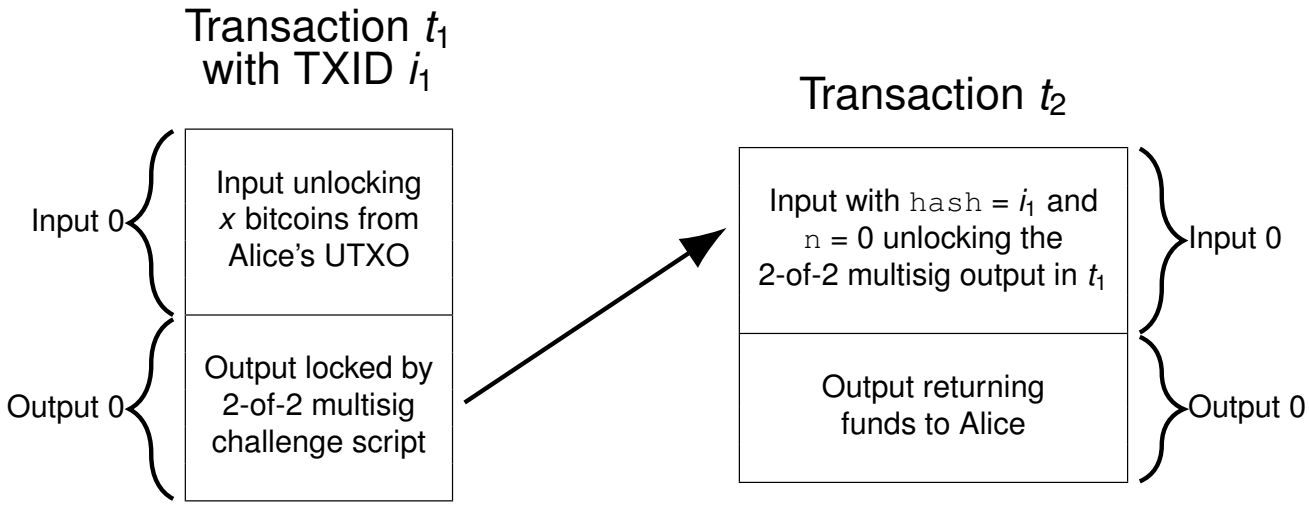
Regular Transaction

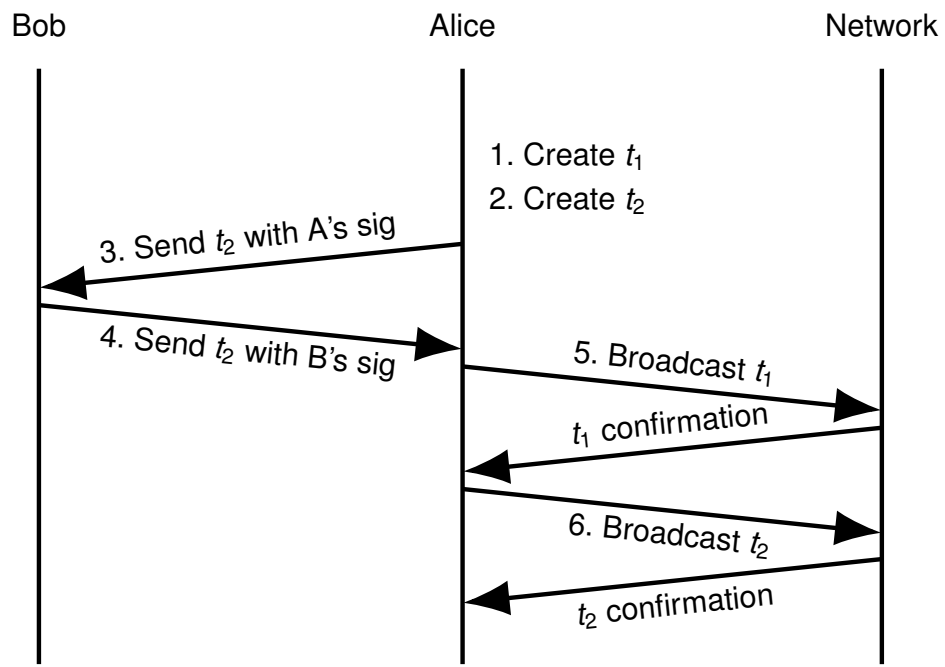


Refund Protocol

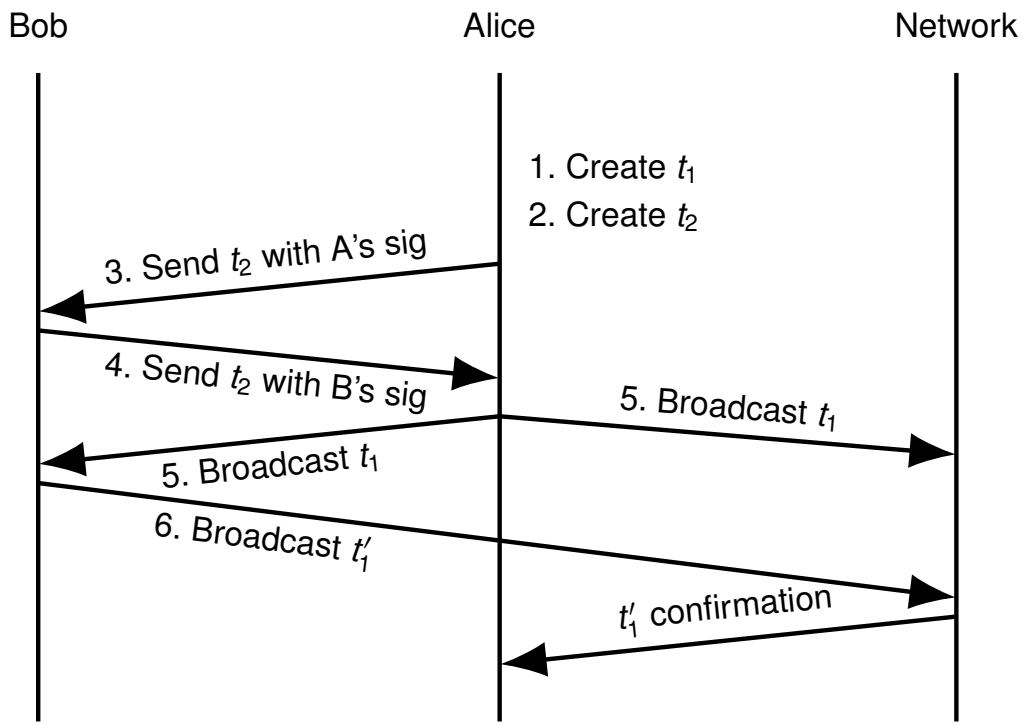
- Alice wants to teach Bob about transactions
- Bob does not own any bitcoins
- Alice decides to transfer some bitcoins to Bob
- Alice does not trust Bob
- She wants to ensure refund

Refund Protocol





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, 13.5 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`